$\mathbb{Z}_2\mathbb{Z}_4$-ADDITIVE CODES

A MAGMA Package

by

Combinatorics and Coding Group (CCG)

Autonomous University of Barcelona

Version 1.7 beta

Barcelona
June 13, 2008

Contents

Preface

The research group CCG (Combinatorics and Coding Group) is one of the research groups in the dEIC (Department of Information and Communications Engineering) at the UAB (Universitat Autònoma de Barcelona) in Spain.

From 1987 the team CCG has been uninterruptedly working in several projects and research activities on Information Theory, Communications, Coding Theory, Source Coding, Teledetection, Cryptography, Electronic Voting, e-Auctions, Mobile Agents, etc.

The more important know-how of CCG is about algorithms for forward error correction (FEC), such as Golay codes, Hamming product codes, Reed-Solomon codes, Preparata and Preparata-like codes, (extended) nonlinear 1-perfect codes, $\mathbb{Z}_4$-linear codes, $\mathbb{Z}_2\mathbb{Z}_4$-linear codes, etc.; computations of the rank and the dimension of the kernel for nonlinear codes as binary 1-perfect codes, q-ary 1-perfect codes, Preparata codes, Hadamard codes, Kerdock codes, quaternary Reed-Muller codes, etc.; the existence and structural properties for 1-perfect codes, uniformly packed codes, completely regular codes, completely transitive codes, etc.

Currently, the research project where CCG is involved is supported by the Spanish MEC and the European FEDER under Grant MTM2006-03250 and also by the UAB under Grant PNL2006-13. Part of this research deals with $\mathbb{Z}_2\mathbb{Z}_4$-linear codes. There are no symbolic software to work with these codes, so the members of CCG has been developing a new package that supports the basic facilities for $\mathbb{Z}_2\mathbb{Z}_4$-additive codes. Specifically, this Magma package generalizes most of the known functions for codes over the ring $\mathbb{Z}_4$, which are subgroups of $\mathbb{Z}_4^n$, to $\mathbb{Z}_2\mathbb{Z}_4$-additive codes, which are subgroups of $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, maintaining all the functionality for codes over $\mathbb{Z}_4$ and adding new functions which, not only generalize the previous ones, but introduce new variants when it is needed. A beta version of this new package for $\mathbb{Z}_2\mathbb{Z}_4$-additive codes and this manual with the description of all functions can be downloaded from the web page `http://www.ccg.uab.es`. For any comment or further information about this package, you can send an e-mail to *support-ccg@deic.uab.cat*.

At present, the members of CCG are: Joaquim Borges, Cristina Fernández, Jaume Pujol, Josep Rifà and Mercè Villanueva, with the collaboration of some assistants, specially the software engineer Bernat Gaston. The group CCG collaborates with other national and international research groups and is open to new collaborations with other researchers.

Chapter 1

$\mathbb{Z}_2\mathbb{Z}_4$-Additive Codes

1.1 Introduction

MAGMA currently supports the basic facilities for linear codes over integer residue rings and galois rings (see [3, Chapter 119]), including cyclic codes and the complete weight enumerator calculation. Moreover, some functions are available for the special case of linear codes over $\mathbb{Z}_4$ (also called $\mathbb{Z}_4$-codes or quaternary linear codes), which are subgroups of $\mathbb{Z}_4^n$. This chapter describes functions which are applicable to codes that are subgroups of $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$.

MAGMA also supports functions for additive codes over a finite field, which are a generalization of the linear codes over a finite field (see [3, Chapter 120]) in a Hamming scheme. According to a more general definition of additive codes given by Delsarte in 1973 (see [4]), the additive codes are subgroups of the underlying abelian group in a translation association scheme. In the special case of a binary Hamming scheme, that is when the underlying abelian group is of size $2^{n_{bin}}$, the only structures for the abelian group are those of the form $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, with $\alpha + 2\beta = n_{bin}$. So, the codes that are subgroups of $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ are also called additive codes. In order to distinguish them from the additive codes over a finite field, we will call them $\mathbb{Z}_2\mathbb{Z}_4$-*additive codes*. Notice that when $\alpha = 0$, these codes are the quaternary linear codes, and when $\beta = 0$, they are the binary linear codes.

Let C be an $\mathbb{Z}_2\mathbb{Z}_4$-additive code. Since C is a subgroup of $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, it is isomorphic to an abelian structure like $\mathbb{Z}_2^\gamma \times \mathbb{Z}_4^\delta$. Therefore, we have that $|C| = 2^\gamma 4^\delta$ and the number of codewords of order two in C is $2^{\gamma+\delta}$. Let X (respectively Y) be the set of $\mathbb{Z}_2$ (respectively $\mathbb{Z}_4$) coordinate positions, so $|X| = \alpha$ and $|Y| = \beta$. Unless otherwise stated, the set X corresponds to the first α coordinates and Y corresponds to the last β coordinates. Call C_X (respectively C_Y) the punctured code of C by deleting the coordinates outside X (respectively Y). Let C_b be the $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode of C which contains all order two codewords and let κ be the dimension of $(C_b)_X$, which is a binary linear code. For the case $\alpha = 0$, we will write $\kappa = 0$. Considering all these parameters, we will say that C is of type $(\alpha, \beta; \gamma, \delta; \kappa)$. Notice that the quaternary linear code C_Y is of type $(0, \beta; \gamma_Y, \delta; 0)$, where $0 \leq \gamma_Y \leq \gamma$, and the binary linear code C_X is a code of length α and dimension γ_X, or equivalently of type $(\alpha, 0; \gamma_X, 0; \gamma_X)$, where $\kappa \leq \gamma_X \leq \gamma$.

Moreover, since the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes are subgroups of $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ and have an abelian structure like $\mathbb{Z}_2^\gamma \times \mathbb{Z}_4^\delta$, every codeword is uniquely expressible in the form

$$\sum_{i=1}^{\gamma} \lambda_i u_i + \sum_{j=\gamma+1}^{\gamma+\delta} \mu_j v_j,$$

where $\lambda_i \in \mathbb{Z}_2$ for $1 \leq i \leq \gamma$, $\mu_j \in \mathbb{Z}_4$ for $\gamma + 1 \leq j \leq \gamma + \delta$ and u_i, v_j are vectors in $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ of order two and order four, respectively. The vectors u_i, v_j give us a generator matrix of size $(\gamma + \delta) \times (\alpha + \beta)$ of the form

$$\left(\begin{array}{c|c} B_1 & 2B_3 \\ \hline B_2 & Q \end{array} \right), \tag{1.1}$$

where B_1, B_2, B_3 are matrices over $\mathbb{Z}_2$ of size $\gamma \times \alpha$, $\delta \times \alpha$ and $\gamma \times \beta$, respectively; and Q is a matrix over $\mathbb{Z}_4$ of size $\delta \times \beta$ with quaternary row vectors of order four.

In this chapter, the term "code" will refer to a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, unless otherwise specified. In order to be able to use the functions for quaternary linear codes, from now on the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes will be represented as quaternary linear codes changing the ones by twos in the first α binary coordinates, so they will be subgroups of $\mathbb{Z}_4^{\alpha+\beta}$. However, these codes are not equivalent to the quaternary linear codes, since the inner product defined in $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ gives us that the dual code of a $\mathbb{Z}_2\mathbb{Z}_4$-additive code is not equivalent to the dual code of the corresponding quaternary linear code.

Finally, as a general reference on $\mathbb{Z}_2\mathbb{Z}_4$-additive codes, the reader is referred to [2], where most of concepts on $\mathbb{Z}_2\mathbb{Z}_4$-additive codes implemented with the following functions are described in detail.

Note: (see [3, p. 3799]) Since the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes will be represented as quaternary linear codes, it is necessary to remember that for modules defined over rings with zero divisors, it is not possible to talk about the concept of dimension (the modules are not free). However, in Magma each code over such a ring has a unique generator matrix corresponding to the Howell form (see [5, 6]). The number of rows k in this generator matrix will be called the *pseudo-dimension* of the code. It should be noted that this pseudo-dimension is not invariant between equivalent codes, and so does not provide structural information like the dimension of a code over a finite field.

1.2 Construction of $\mathbb{Z}_2\mathbb{Z}_4$-Additive Codes

1.2.1 Construction of General $\mathbb{Z}_2\mathbb{Z}_4$-Additive Codes

`Z2Z4AdditiveCode(L : parameters)`

`Alpha`	RngIntElt	*Default*: 0
`OverZ2`	BoolElt	*Default*: `false`

Create a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C given as a record with two fields:

- `Alpha`: The length of the binary part of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code.
- `Code`: The quaternary linear code equal to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code, where the ones in the first `Alpha` coordinates are represented by twos.

The corresponding quaternary linear code of C is obtained using the function `LinearCode()` as a subspace of $V = \mathbb{Z}_4^n$ generated by L, where:

1. L is a sequence of elements of V,
2. or, L is a subspace of V,
3. or, L is a $m \times n$ matrix A over the ring $\mathbb{Z}_4$,
4. or, L is a quaternary linear code,
5. or, L is a binary linear code.

The parameter `Alpha` specifies the length of the binary part of the code. The default value is 0.

The parameter `OverZ2` specifies whether the first `Alpha` coordinates of each element in L are represented as elements in $\mathbb{Z}_2 = \{0, 1\}$ or, otherwise, they are represented as elements in $\{0, 2\} \subset \mathbb{Z}_4$. The default value is `false`. If `OverZ2` is `true` and `Alpha` > 0, then the first `Alpha` coordinates of each element in L must be 0 or 1. Otherwise, if `OverZ2` is `false` and `Alpha` > 0, then the first `Alpha` coordinates of each element in L must be 0 or 2.

Example H1E1

We define a $\mathbb{Z}_2\mathbb{Z}_4$-additive code by giving a sequence of elements of V, or equivalently a $m \times n$ matrix over $\mathbb{Z}_4$, and the length of the binary part of the code.

```
> Z4 := IntegerRing(4);

> V := RSpace(Z4, 5);
> C1 := Z2Z4AdditiveCode([V![2,2,1,1,3],V![0,2,1,2,1],
                          V![2,2,2,2,2],V![2,0,1,1,1]] : Alpha:=2);
> C1;
rec<Z2Z4Code |
    Code := ((5, 4^2 2^2)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 0 2]
    [0 2 0 0 2]
    [0 0 1 0 3]
    [0 0 0 1 0],
    Alpha := 2
    >

> A := Matrix(Z4, [[2,2,1,1,3],[0,2,1,2,1],[2,2,2,2,2],[2,0,1,1,1]]);
> C2 := Z2Z4AdditiveCode(A : Alpha:=2);
> C2;
rec<Z2Z4Code |
    Code := ((5, 4^2 2^2)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 0 2]
    [0 2 0 0 2]
    [0 0 1 0 3]
    [0 0 0 1 0],
    Alpha := 2
    >

> Z2Z4Equal(C1, C2);
true
```

Alternatively, we also define the same $\mathbb{Z}_2\mathbb{Z}_4$-additive codes by giving the first `Alpha` coordinates of each vector as elements in $\{0, 1\}$ instead of in $\{0, 2\}$, changing the twos by ones and adding the parameter `OverZ2`.

```
> C1b := Z2Z4AdditiveCode([V![1,1,1,1,3],V![0,1,1,2,1],V![1,1,2,2,2],
                           V![1,0,1,1,1]] : Alpha:=2, OverZ2:=true);

> Ab := Matrix(Z4, [[1,1,1,1,3],[0,1,1,2,1],[1,1,2,2,2],[1,0,1,1,1]]);
> C2b := Z2Z4AdditiveCode(Ab : Alpha:=2, OverZ2:=true);

> Z2Z4Equal(C1, C1b) and Z2Z4Equal(C2, C2b);
true
```

Example H1E2

Any quaternary linear code and any binary linear code is also a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, since the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes can be seen as a generalization of the quaternary linear codes, when $\alpha = 0$, and a generalization of the binary linear codes, when $\beta = 0$.

```
> B := RandomLinearCode(GF(2),7,3);
> B;
[7, 3, 1] Linear Code over GF(2)
Generator matrix:
[1 0 0 1 0 0 1]
[0 1 1 1 0 1 1]
[0 0 0 0 1 0 0]
> B_add := Z2Z4AdditiveCode(B);
> B_add;
rec<Z2Z4Code |
    Code := ((7, 4^0 2^3)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 2 0 0 2]
    [0 2 2 2 0 2 2]
    [0 0 0 0 2 0 0],
    Alpha := 7
    >

> Q := RandomLinearCode(IntegerRing(4),5,2);
> Q;
((5, 4^1 2^1)) Linear Code over IntegerRing(4)
Generator matrix:
[2 0 0 2 0]
[0 1 0 1 1]
> Q_add := Z2Z4AdditiveCode(Q);
> Q_add;
rec<Z2Z4Code |
    Code := ((5, 4^1 2^1)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 2 0]
    [0 1 0 1 1],
    Alpha := 0
    >
```

1.2.2 Some Trivial $\mathbb{Z}_2\mathbb{Z}_4$-Additive Codes

`Z2Z4AdditiveZeroCode(`α`,`β`)`

Given two positive integers α and β, return the $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; 0, 0; 0)$ consisting of only the zero codeword.

`Z2Z4AdditiveRepetitionCode(`α`,`β`)`

Given two positive integers α and β, return the $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; 1, 0; 1)$ generated by the vector $(1, \dots, 1|2, \dots, 2)$.

`Z2Z4AdditiveUniverseCode(`α`,`β`)`

Given two positive integers α and β, return the generic $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; \alpha, \beta; \alpha)$ consisting of all possible codewords.

`RandomZ2Z4AdditiveCode([`α`,`β`,`γ`,`δ`,`κ`])`

Given a list of integers $[\alpha,\ \beta,\ \gamma,\ \delta,\ \kappa]$, return a random $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; \gamma, \delta; \kappa)$. Notice that there exists a $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; \gamma, \delta; \kappa)$ as long as $\kappa \leq min(\alpha, \gamma)$ and $\delta + \gamma \leq \beta + \kappa$. When the list has less than five integers, they correspond to the first ones in the above order and the rest are computed randomly.

Example H1E3

The zero $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; 0, 0; 0)$ is contained in every $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; \gamma, \delta; \kappa)$, and similarly every $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; \gamma, \delta; \kappa)$ is contained in the universe $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; \alpha, \beta; \alpha)$.

```
> a := 2; b := 3;
> U := Z2Z4AdditiveUniverseCode(a, b);
> Z := Z2Z4AdditiveZeroCode(a, b);
> R := RandomZ2Z4AdditiveCode([a, b]);
> Z2Z4Subset(Z, R) and Z2Z4Subset(R, U);
true
```

1.3 Invariants of a $\mathbb{Z}_2\mathbb{Z}_4$-Additive Code

1.3.1 Basic Numerical Invariants

`Z2Z4Length(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the length $n = \alpha + \beta$, and the list $[\alpha, \beta]$ with the number of coordinates over $\mathbb{Z}_2$ and the number of coordinates over $\mathbb{Z}_4$, respectively.

`Z2Z4BinaryLength(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the binary length $n_{bin} = \alpha + 2\beta$ of C, which corresponds to the length of the binary code $C_{bin} = \Phi(C)$, where Φ is the Gray map defined in Subsection 1.5.1.

`Z2Z4PseudoDimension(C)`

`Z2Z4NumberOfGenerators(C)`

`Z2Z4Ngens(C)`

The number of generators (which equals the pseudo-dimension k) of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C as a quaternary linear code.

`Z2Z4Type(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return a list with the parameters $[\alpha, \beta, \gamma, \delta, \kappa]$, that is, the type of the code.

`Z2Z4Cardinal(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the number of codewords belonging to C, that is $2^\gamma 4^\delta$.

`Z2Z4InformationRate(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the information rate of C, that is the ratio $(\gamma + 2\delta)/(\alpha + 2\beta)$.

Example H1E4

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, we compute its type, the number of codewords and its information rate.

```
> C := Z2Z4AdditiveCode([[2,2,1,1,3,1],[2,2,2,2,2,2],[0,0,1,1,1,3]] :
                                Alpha:=2);
> Z2Z4Type(C);
[ 2, 4, 2, 1, 1 ]
> Z2Z4Cardinal(C);
16
> Z2Z4InformationRate(C);
0.400000000000000000000000000000
```

1.3.2 The Code Space

`Z2Z4Generic(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the generic $\mathbb{Z}_2\mathbb{Z}_4$-additive code of type $(\alpha, \beta; \alpha, \beta; \alpha)$ in which C is contained.

`Z2Z4Name(C,i)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C and a positive integer i, return the i-th generator of C as a quaternary linear code, where the ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4Set(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return the set containing all its codewords, where the ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4Generators(C)`

The generators for the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C as a quaternary linear code, returned as a set of elements of C, where the ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4GeneratorMatrix(C)`

The unique generator matrix for the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C as a quaternary linear code, corresponding to the Howell form (see [5, 6]). The ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4OrderTwoGenerators(C)`

The γ generators of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, returned as a sequence of elements of C, where the ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4OrderFourGenerators(C)`

The δ generators of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, returned as a sequence of elements of C, where the ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4OrderTwoSubcodeGenerators(C)`

The $\gamma + \delta$ generators of the $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode C_b which contains all order two codewords of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, returned as a sequence of elements of C_b, where the ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4MinRowsGeneratorMatrix(C)`

A generator matrix of the form (1.1) for the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, with the minimum number of rows, that is with $\gamma + \delta$ rows: γ rows of order two and δ rows of order four. The ones in the first `Alpha` coordinates are represented by twos.

1.3.3 Conversion Functions

The $\mathbb{Z}_2\mathbb{Z}_4$-additive codes are internaly represented as quaternary linear codes, with their first `Alpha` coordinates represented as 0,2 over $\mathbb{Z}_4$ insted of 0,1 over $\mathbb{Z}_2$. This section provides functions to convert the outputs given as a sequence, set or matrix of vectors over $\mathbb{Z}_4^{\alpha+\beta}$ to a sequence or a set of tuples in the cartesian product set $\mathbb{Z}_2^{\alpha} \times \mathbb{Z}_4^{\beta}$, changing the twos over $\mathbb{Z}_4$ in the first `Alpha` coordinates to ones over $\mathbb{Z}_2$.

$\boxed{\texttt{FromZ4toZ2Z4(L,Alpha)}}$

Given a sequence L of vectors over $\mathbb{Z}_4^{\alpha+\beta}$ and an integer `Alpha`, return the conversion of these vectors to a sequence of tuples in the cartesian product set $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, changing the twos over $\mathbb{Z}_4$ in the first `Alpha` coordinates to ones over $\mathbb{Z}_2$.

$\boxed{\texttt{FromZ4toZ2Z4(S,Alpha)}}$

Given a set S of vectors over $\mathbb{Z}_4^{\alpha+\beta}$ and an integer `Alpha`, return the conversion of these vectors to a set of tuples in the cartesian product set $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, changing the twos over $\mathbb{Z}_4$ in the first `Alpha` coordinates to ones over $\mathbb{Z}_2$.

$\boxed{\texttt{FromZ4toZ2Z4(M,Alpha)}}$

Given a matrix M over $\mathbb{Z}_4$ with $\alpha+\beta$ columns and an integer `Alpha`, return the conversion from M to a sequence of tuples in the cartesian product set $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, changing the twos over $\mathbb{Z}_4$ in the first `Alpha` coordinates to ones over $\mathbb{Z}_2$.

1.3.4 The Dual Space

The inner product in $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ is defined uniquely by

$$\langle u, v\rangle = 2(\sum_{i=1}^{\alpha} u_i v_i) + \sum_{j=\alpha+1}^{\alpha+\beta} u_j v_j \in \mathbb{Z}_4, \tag{1.2}$$

where $u, v \in \mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$. Note that when $\alpha = 0$ the inner product is the usual one for vectors over $\mathbb{Z}_4$, and when $\beta = 0$ it is twice the usual one for vectors over $\mathbb{Z}_2$.

The *additive orthogonal code* of a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, denoted by $C^\perp$, is defined in the standard way

$$C^\perp = \{v \in \mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta \mid \langle u, v\rangle = 0 \text{ for all } u \in C\}.$$

We will also call $C^\perp$ the *additive dual code* of C. Note that the additive dual code $C^\perp$ is also a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, that is a subgroup of $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$.

$\boxed{\texttt{Z2Z4Dual(C)}}$

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return the additive dual code of C.

$\boxed{\texttt{Z2Z4DualType(C)}}$

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return a list with the parameters $[\alpha, \beta, \gamma', \delta', \kappa']$, that is the type of the additive dual code of C. If C is of type $(\alpha, \beta; \gamma, \delta; \kappa)$, then its additive dual code is of type $(\alpha, \beta; \alpha + \gamma - 2\kappa, \beta - \gamma - \delta + \kappa; \alpha - \kappa)$.

`Z2Z4ParityCheckMatrix(C)`

The unique parity-check matrix for the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, that is the unique generator matrix for the additive dual code of C as a quaternary linear code, corresponding to the Howell form (see [5, 6]). The ones in the first `Alpha` coordinates are represented by twos.

`Z2Z4MinRowsParityCheckMatrix(C)`

A parity-check matrix of the form (1.1) for the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, that is a generator matrix for the additive dual code of C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, with the minimum number of rows, that is with $\gamma+\delta$ rows: γ rows of order two and δ rows of order four. The ones in the first `Alpha` coordinates are represented by twos.

1.4 The Standard Form

A $\mathbb{Z}_2\mathbb{Z}_4$-additive code is in *standard form* if its generator matrix is of the form:

$$\left(\begin{array}{cc|ccc} I_\kappa & T_b & 2T_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 2T_1 & 2I_{\gamma-\kappa} & \mathbf{0} \\ \hline \mathbf{0} & S_b & S_q & R & I_\delta \end{array}\right),$$

where T_b, T_1, T_2, R, S_b are matrices over $\mathbb{Z}_2$ and S_q is a matrix over $\mathbb{Z}_4$. Any $\mathbb{Z}_2\mathbb{Z}_4$-additive code C is permutation-equivalent to a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C_{SF} which is in standard form (see [1, 2]). Two $\mathbb{Z}_2\mathbb{Z}_4$-additive codes which differ only by a coordinate permutation are said to be *permutation-equivalent.*

`Z2Z4StandardForm(C)`

Given any $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return a permutation-equivalent $\mathbb{Z}_2\mathbb{Z}_4$-additive code C_{SF} in standard form, together with the corresponding isomorphism from C to C_{SF} and the generator matrix in standard form.

Example H1E5

We compute the standard form of a certain $\mathbb{Z}_2\mathbb{Z}_4$-additive code. Note that the number of rows in the general matrix of the standard code may be less than the number of rows in the matrix of the original code.

```
> C := Z2Z4AdditiveCode([[2,0,2,1,1,1,2,1],[0,0,0,2,0,1,1,1],
                         [0,0,0,0,2,1,3,1],[0,0,0,0,0,2,2,0],
                         [0,0,0,0,0,0,0,2]] : Alpha:=3);
> C;
rec<Z2Z4Code |
    Code := ((8, 4^2 2^1)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 2 1 1 1 2 1]
    [0 0 0 2 0 1 1 1]
    [0 0 0 0 2 1 3 1]
    [0 0 0 0 0 2 2 0]
    [0 0 0 0 0 0 0 2],
    Alpha := 3
    >

> C_SF,f,G_SF:=Z2Z4StandardForm(C);
> C_SF;
rec<recformat<Code: CodeLinRng, Alpha: RngIntElt> |
    Code := ((8, 4^2 2^1)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 2 1 1 1 1 2]
    [0 0 0 2 0 1 1 1]
    [0 0 0 0 2 1 1 3]
    [0 0 0 0 0 2 0 2]
    [0 0 0 0 0 0 2 0],
    Alpha := 3
    >
> G_SF;
[0 0 0 2 2 2 0 0]
[2 0 2 3 3 1 1 0]
[2 0 2 1 3 2 0 1]
> f;
Mapping from: ((8, 4^2 2^1)) Linear Code over IntegerRing(4) to ((8, 4^2 2^1))
Linear Code over IntegerRing(4) given by a rule
> c:=Z2Z4Random(C);
> f(c) in C_SF‘Code;
true
```

1.5 Derived Binary and Quaternary Codes

1.5.1 The Gray Map

The $\mathbb{Z}_2\mathbb{Z}_4$-additive codes are subgroups C of $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ and the corresponding binary codes are $C_{bin} = \Phi(C)$, where Φ is the following extension of the usual Gray map: $\Phi : \mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta \longrightarrow \mathbb{Z}_2^{n_{bin}}$, where $n_{bin} = \alpha + 2\beta$, given by

$$\Phi(x, y) = (x, \phi(y_1), \ldots, \phi(y_\beta)) \quad \forall x \in \mathbb{Z}_2^\alpha,\ \forall y = (y_1, \ldots, y_\beta) \in \mathbb{Z}_4^\beta;$$

where $\phi : \mathbb{Z}_4 \longrightarrow \mathbb{Z}_2^2$ is the usual Gray map, that is,

$$\phi(0) = (0, 0),\ \phi(1) = (0, 1),\ \phi(2) = (1, 1),\ \phi(3) = (1, 0).$$

This Gray map is an isometry which transforms Lee distances defined in codes C over $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ to Hamming distances defined in the binary codes

$C_{bin} = \Phi(C)$ (see Subsection 1.6.3). Note that the length of the binary code C_{bin} is $n_{bin} = \alpha + 2\beta$.

`Z2Z4GrayMap(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return the Gray map for C. This is the map Φ from C to $\Phi(C)$, as defined above.

`Z2Z4GrayMapImage(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return the image of C under the Gray map as a sequence of vectors in $\mathbb{Z}_2^{\alpha+2\beta}$. As the resulting image may not be a binary linear code, a sequence of vectors is returned rather than a code.

`HasZ2Z4LinearGrayMapImage(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return `true` if and only if the image of C under the Gray map is a binary linear code. If so, the function also returns the image B as a binary linear code, together with the bijection $\Phi : C \to B$.

Example H1E6

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, we compute its image under the Gray map. This image is not always a binary linear code.

```
> C := Z2Z4AdditiveCode([[2,0,0,2],[0,1,0,3],[0,0,1,3]] : Alpha:=1);
> C;
rec<Z2Z4Code |
    Code := ((4, 4^2 2^1)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 2]
    [0 1 0 3]
    [0 0 1 3],
    Alpha := 1
    >
> HasZ2Z4LinearGrayMapImage(C);
false
> Cb := Z2Z4GrayMapImage(C);
> #Cb;
32

> D := Z2Z4AdditiveCode([[2,2,1,1,3,1],[2,2,2,2,2,2],[0,0,1,1,1,3]] :
                          Alpha:=2);
> D;
rec<Z2Z4Code |
    Code := ((6, 4^1 2^2)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 2 0 0 0 0]
    [0 0 1 1 1 3]
    [0 0 0 0 2 2],
    Alpha := 2
    >
> f := Z2Z4GrayMap(D);
> D`Code.1;
(2 2 0 0 0 0)
> f(D`Code.1);
(1 1 0 0 0 0 0 0 0 0)
> D`Code.2;
(0 0 1 1 1 3)
> f(D`Code.2);
(0 0 0 1 0 1 0 1 1 0)
> D`Code.3;
(0 0 0 0 2 2)
> f(D`Code.3);
(0 0 0 0 0 0 1 1 1 1)
> l, B, f:=HasZ2Z4LinearGrayMapImage(D);
> l;
true
> B;
(10, 16, 2) Linear Code over IntegerRing(2)
Generator matrix:
[1 1 0 0 0 0 0 0 0 0]
[0 0 1 0 1 0 0 1 1 0]
[0 0 0 1 0 1 0 1 1 0]
[0 0 0 0 0 0 1 1 1 1]
> f(D`Code.1) in B;
true
> f(D`Code.2) in B;
true
> f(D`Code.3) in B;
true
> Length(B) eq Z2Z4BinaryLength(D);
true
```

1.5.2 From Linear to $\mathbb{Z}_2\mathbb{Z}_4$-Additive Codes

`Z2Z4AdditiveFromBinaryLinearCode(C)`

Given a binary linear code C of length n, return the same code as a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, so with $\alpha = n$ and $\beta = 0$.

`Z2Z4AdditiveFromQuaternaryLinearCode(C)`

Given a quaternary linear code C of length n, return the same code as a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, so with $\alpha = 0$ and $\beta = n$.

Example H1E7

We convert a binary linear code and a quaternary linear code to a $\mathbb{Z}_2\mathbb{Z}_4$-additive code.

```
> B := RandomLinearCode(GF(2),7,3);
> Q := RandomLinearCode(IntegerRing(4),5,2);

> B_add := Z2Z4AdditiveFromBinaryLinearCode(B);
> B_add;
rec<Z2Z4Code |
    Code := ((7, 4^0 2^3)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 2 0 0 2 2]
    [0 2 0 0 0 0 0]
    [0 0 0 2 0 2 0],
    Alpha := 7
    >

> Q_add:=Z2Z4AdditiveFromQuaternaryLinearCode(Q);
> Q_add;
rec<Z2Z4Code |
    Code := ((5, 4^2 2^0)) Linear Code over IntegerRing(4)
    Generator matrix:
    [1 0 0 3 3]
    [0 1 0 3 3],
    Alpha := 0
    >

> IsZ2Z4AdditiveCode(B_add) and IsZ2Z4AdditiveCode(Q_add);
true
```

1.5.3 Subcodes C_X and C_Y

`Z2Z4LinearBinaryCode(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the binary linear code C_X of length α which is the punctured code of C by deleting the coordinates outside X, where X is the set of the first α coordinates.

`Z2Z4LinearQuaternaryCode(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the quaternary linear code C_Y of length β which is the punctured code of C by deleting the coordinates outside Y, where Y is the set of the last β coordinates.

Example H1E8

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, we compute the corresponding binary linear code C_X and the corresponding quaternary linear code C_Y.

```
> C := Z2Z4AdditiveCode([[2,0,0,2],[0,1,0,3],[0,0,1,3]] : Alpha:=1);
> CX := Z2Z4LinearBinaryCode(C);
> CX;
```

```
(1, 2, 1) Cyclic Linear Code over IntegerRing(2)
Generator matrix:
[1]
> CY := Z2Z4LinearQuaternaryCode(C);
> CY;
((3, 4^2 2^1)) Cyclic Linear Code over IntegerRing(4)
Generator matrix:
[1 0 1]
[0 1 1]
[0 0 2]
> _,n := Z2Z4Length(C);
> alpha := n[1]; beta := n[2];
> alpha; beta;
1
3
> Length(CX) eq alpha and Length(CY) eq beta;
true
```

1.6 Operations on Codewords

A $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha,\beta;\gamma,\delta;\kappa)$ is represented in MAGMA as a record with two fields: `Alpha`, the length of the binary part of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code; and `Code`, the quaternary linear code (or equivalently, the subspace of $V=\mathbb{Z}_4^{\alpha+\beta}$) equal to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code, where the ones in the first `Alpha` coordinates are represented by twos (see Subsection 1.2.1).

In this section, in order to use the same functions as for quaternary linear codes, notice that we will write `C‘Code` instead of `C`. Also notice that all codewords, which are elements in $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, are represented as elements in $V=\mathbb{Z}_4^{\alpha+\beta}$ by changing the ones in the first `Alpha` coordinates by twos.

1.6.1 Construction of a Codeword

`C‘Code ! [` $a_1,\ldots,a_n$ `]`

`elt< C‘Code |` $a_1,\ldots,a_n$ `>`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, which is represented in MAGMA as a subspace of $V=\mathbb{Z}_4^n$ $(n=\alpha+\beta)$, and elements $a_1,\ldots,a_n$ belonging to $\mathbb{Z}_4$, construct the codeword $(a_1,\ldots,a_n)$ of C. It is checked that the vector $(a_1,\ldots,a_n)$ is an element of C.

`C‘Code ! u`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, which is represented in Magma as a subspace of $V = \mathbb{Z}_4^{\alpha+\beta}$, and an element u belonging to V, create the codeword of C corresponding to u. The function will fail if u does not belong to C.

`C‘Code ! 0`

The zero word of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C.

`Z2Z4Random(C)`

A random codeword of a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, where the ones in the first `Alpha` coordinates are represented by twos.

Example H1E9

We create some elements of a $\mathbb{Z}_2\mathbb{Z}_4$-additive code.

```
> C := Z2Z4AdditiveCode([[2,0,0,2],[0,1,0,3],[0,0,1,3]] : Alpha:=1);
> C‘Code![2,0,0,2];
(2 0 0 2)
> elt<C‘Code | 2,1,1,0>;
(2 1 1 0)
> Z2Z4Random(C);
(2 0 1 1)
```

If the given vector does not lie in the given $\mathbb{Z}_2\mathbb{Z}_4$-additive code, then an error will result.

```
> C‘Code![2,1,0,0];

>> C‘Code![2,1,0,0];
         ^
Runtime error in ’!’: Result is not in the given structure
```

1.6.2 Operations on Codewords and Vectors

`u + v`

Sum of the codewords u and v, where u and v belong to the same $\mathbb{Z}_2\mathbb{Z}_4$-additive code C.

`- u`

Additive inverse of the codeword u belonging to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C.

`u - v`

Difference of the codewords u and v, where u and v belong to the same $\mathbb{Z}_2\mathbb{Z}_4$-additive code C.

`a * u`

Given an element a belonging to $\mathbb{Z}_4$, and a codeword u belonging to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return the codeword $a \cdot u$.

`Normalize(u)`

Given an element u belonging to a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, which is represented in MAGMA as a subspace of $V = \mathbb{Z}_4^{\alpha+\beta}$, return the normalization of u, which is the unique vector v such that $v = a \cdot u$ for some scalar a in $\mathbb{Z}_4$ such that the first non-zero entry of v is the canonical associate in $\mathbb{Z}_4$ of the first non-zero entry of u (v is zero if u is zero).

`Z2Z4InnerProduct(u, v, α)`

The inner product $\langle u, v\rangle$ in $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$ defined in Subsection 1.3.4. The vectors u and v are represented as vectors in $V = \mathbb{Z}_4^{\alpha+\beta}$, that is the parent vector space of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C.

`Support(w)`

Given a codeword w belonging to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, which is represented in MAGMA as a subspace of $V = \mathbb{Z}_4^{\alpha+\beta}$, return its support as a subset of the integer set $\{1, \dots, \alpha + \beta\}$. The support of w consists of the coordinates at which w has non-zero entries.

`Z2Z4Coordinates(C, u)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C and a codeword u of C, return the coordinates of u with respect to C. The coordinates of u are returned as a sequence $Q = [a_1, \dots, a_k]$ of elements from $\mathbb{Z}_4$ so that $u = a_1 \cdot C\text{`}Code.1 + \cdots + a_k \cdot C\text{`}Code.k$.

`Rotate(u, k)`

Given a vector u, return the vector obtained from u by cyclically shifting its components to the right by k coordinate positions.

`Rotate(~u, k)`

Given a vector u, destructively rotate u by k coordinate positions.

`Parent(w)`

Given a codeword w belonging to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, return the ambient space $V = \mathbb{Z}_4^{\alpha+\beta}$ of C.

Example H1E10

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, we explore various operations on its codewords.

```
> C := Z2Z4AdditiveCode([[2,0,0,2],[0,1,0,3],[0,0,1,3]] : Alpha:=1);
> u := C`Code.1;
> v := C`Code.2;
> u; v;
```

```
(2 0 0 2)
(0 1 0 3)
> u+v;
(2 1 0 1)
> 2*v;
(0 2 0 2)
> u+v in C`Code;
true
> Z2Z4InnerProduct(u, v, 1);
2
> Support(u);
{ 1, 4 }
> Z2Z4Coordinates(C, u+2*v);
[ 1, 2, 0 ]
> Parent(u);
Full RSpace of degree 4 over IntegerRing(4)
```

1.6.3 Distance and Weight

`Weight(v)`

The Hamming weight of the codeword v, i.e., the number of non-zero components of v.

`Distance(u, v)`

The Hamming distance between the codewords u and v, where u and v belong to the same $\mathbb{Z}_2\mathbb{Z}_4$-additive code C. This is defined to be the Hamming weight of $u - v$.

`Z2Z4LeeWeight(v, α)`

The Lee weight of the codeword v, i.e., the Hamming weight of the binary part (that is, the first α coordinates) of v plus the Lee weight of the quaternary part (that is, the rest of the coordinates) of v (see [2]). Equivalently, it corresponds to the number of non-zero components of $\Phi(v)$, where Φ is the Gray map defined in Subsection 1.5.1.

`Z2Z4LeeDistance(u, v, α)`

The Lee distance between the codewords u and v, where u and v belong to the same $\mathbb{Z}_2\mathbb{Z}_4$-additive code C. This is defined to be the Lee weight of $u - v$.

Example H1E11

We calculate the Hamming weight and distance of some vectors in $V = \mathbb{Z}_4^{\alpha+\beta}$, as well as the Lee weight and distance of these vectors. Notice that when $\alpha = 0$, the functions for the Lee weight and distance return the same as that the corresponding functions for vectors over $\mathbb{Z}_4$.

```
> V := RSpace(IntegerRing(4),4);
> u := V![2,1,2,3];
> v := V![0,0,2,1];
> Distance(u, v);
3
> Distance(u, v) eq Weight(u-v);
true
> Z2Z4LeeDistance(u, v, 1);
4
> Z2Z4LeeDistance(u, v, 1) eq Z2Z4LeeWeight(u-v, 1);
true
> Z2Z4LeeDistance(u, v, 0) eq LeeDistance(u, v);
true
> Z2Z4LeeWeight(u, 0) eq LeeWeight(u);
true
```

1.6.4 Accessing Components of a Codeword

`u[i]`

Given a codeword u belonging to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, which is represented in MAGMA as a subspace of $V = \mathbb{Z}_4^{\alpha+\beta}$, return the i-th component of u (as an element of $\mathbb{Z}_4$).

`u[i] := x;`

Given an element u belonging to a $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode C of the full $\mathbb{Z}_4$-space $V = \mathbb{Z}_4^{\alpha+\beta}$, a positive integer i, $1 \leq i \leq \alpha + \beta$, and an element x of $\mathbb{Z}_4$, this function returns a vector in V which is u with its i-th component redefined to be x.

1.7 Boolean Predicates

Again notice that sometimes in order to use the same functions as for quaternary linear codes, we will write `C‘Code` instead of `C`. Also, all codewords, which are elements in $\mathbb{Z}_2^\alpha \times \mathbb{Z}_4^\beta$, are represented as elements in $V = \mathbb{Z}_4^{\alpha+\beta}$ by changing the ones in the first `Alpha` coordinates by twos.

1.7.1 Membership and Equality

`u in C‘Code`

Return `true` if and only if the vector u of $V = \mathbb{Z}_4^{\alpha+\beta}$ belongs to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$.

`u notin C‘Code`

Return `true` if and only if the vector u of $V = \mathbb{Z}_4^{\alpha+\beta}$ does not belong to the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$.

`Z2Z4Subset(C, D)`

Return `true` if and only if the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C is a subcode of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code D.

`Z2Z4NotSubset(C, D)`

Return `true` if and only if the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C is not a subcode of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code D.

`Z2Z4Equal(C, D)`

Return `true` if and only if the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes C and D are equal.

`Z2Z4NotEqual(C, D)`

Return `true` if and only if the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes C and D are not equal.

`IsZero(u)`

Return `true` if and only if the codeword u is the zero vector.

1.7.2 Properties of $\mathbb{Z}_2\mathbb{Z}_4$-Additive Codes

`IsZ2Z4AdditiveCode(C)`

Return `true` if and only if C is a $\mathbb{Z}_2\mathbb{Z}_4$-additive code.

`IsZ2Z4SelfDual(C)`

Return `true` if and only if the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C is additive self-dual, i.e., C equals the additive dual of C.

`IsZ2Z4SelfOrthogonal(C)`

Return `true` if and only if the $\mathbb{Z}_2\mathbb{Z}_4$-additive code C is additive self-orthogonal, that is, return whether C is contained in the additive dual of C.

Example H1E12

We consider a $\mathbb{Z}_2\mathbb{Z}_4$-additive code and examine some of its properties.

```
> C:=Z2Z4AdditiveCode([[2,2,2,0,0],[0,0,0,2,0],[0,2,1,0,1]] : Alpha:=2);
> C;
rec<Z2Z4Code |
    Code := ((5, 4^1 2^2)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 1 0 3]
    [0 2 1 0 1]
    [0 0 2 0 2]
    [0 0 0 2 0],
    Alpha := 2
    >
> IsZ2Z4SelfOrthogonal(C);
true
> IsZ2Z4SelfDual(C);
true
> Z2Z4Equal(C,Z2Z4Dual(C));
true
```

1.8 Constructing New Codes from Old

1.8.1 Construction of Subcodes

`Z2Z4OrderTwoSubcode(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$, return the $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode C_b which contains all order two codewords of C.

`Z2Z4Subcode(C, t1, t2)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$ and two integers, $t1$ and $t2$, such that $1 \leq t1 \leq \gamma$ and $1 \leq t2 \leq \delta$, return a $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode of C of type $(\alpha, \beta; t1, t2; \kappa')$, where $\kappa' \leq \kappa$. This $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode is generated by the first $t1$ rows of order two and the first $t2$ rows of order four in the generator matrix given by the function `Z2Z4MinRowsGeneratorMatrix(C)`.

`Z2Z4Subcode(C, S1, S2)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C of type $(\alpha, \beta; \gamma, \delta; \kappa)$ and two sets of integers, $S1$ and $S2$, such that each of their elements lies in the range $[1, \gamma]$ and $[1, \delta]$, respectively, return a $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode of C of type $(\alpha, \beta; |S1|, |S2|; \kappa')$, where $\kappa' \leq \kappa$. This $\mathbb{Z}_2\mathbb{Z}_4$-additive subcode is generated by the rows of order two and four whose positions appear in $S1$ and $S2$, respectively, in the generator matrix given by the function `Z2Z4MinRowsGeneratorMatrix(C)`.

Example H1E13

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code, we compute some subcodes of this code.

```
> C := RandomZ2Z4AdditiveCode([2,4,2,3]);
> C;
rec<Z2Z4Code |
    Code := ((6, 4^3 2^2)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 0 0 0]
    [0 2 0 0 0 1]
    [0 0 1 0 1 0]
    [0 0 0 1 0 0]
    [0 0 0 0 2 0]
    [0 0 0 0 0 2],
    Alpha := 2
    >
> Z2Z4Type(C);
[ 2, 4, 2, 3, 1 ]
> Z2Z4MinRowsGeneratorMatrix(C);
[0 0 0 0 2 0]
[2 0 0 0 0 0]
[0 0 1 0 1 0]
[0 0 0 1 0 0]
[0 2 0 0 0 1]

> C1:=Z2Z4Subcode(C,2,1);
> C1;
rec<Z2Z4Code |
    Code := ((6, 4^1 2^2)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 0 0 0]
    [0 0 1 0 1 0]
    [0 0 0 0 2 0],
    Alpha := 2
    >
> Z2Z4Subset(C1, C);
true
> Z2Z4Type(C1);
```

```
[ 2, 4, 2, 1, 1 ]
> Z2Z4MinRowsGeneratorMatrix(C1);
[0 0 2 0 0 0]
[2 0 0 0 0 0]
[0 0 1 0 1 0]

> C2 := Z2Z4Subcode(C,{2},{1,3});
> C2;
rec<Z2Z4Code |
    Code := ((6, 4^2 2^1)) Linear Code over IntegerRing(4)
    Generator matrix:
    [2 0 0 0 0 0]
    [0 2 0 0 0 1]
    [0 0 1 0 1 0]
    [0 0 0 0 0 2],
    Alpha := 2
    >
> Z2Z4MinRowsGeneratorMatrix(C2);
[2 0 0 0 0 0]
[0 0 1 0 1 0]
[0 2 0 0 0 1]
```

1.8.2 Sum and Intersection

`Z2Z4Sum(C, D)`

The (vector space) sum of the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes C and D, where C and D have the same parameters α and β, hence also the same length.

`Z2Z4Intersection(C, D)`

The intersection of the $\mathbb{Z}_2\mathbb{Z}_4$-additive codes C and D, where C and D have the same parameters α and β, hence also the same length.

Example H1E14

We verify some simple results from the sum and intersection of $\mathbb{Z}_2\mathbb{Z}_4$-additive subcodes.

```
> C:=Z2Z4AdditiveCode([[2,0,0,2],[0,1,0,3],[0,0,1,3]] : Alpha:=1);

> C1:=Z2Z4Subcode(C,1,0);
> C2:=Z2Z4Subcode(C,0,1);
> C3:=Z2Z4Subcode(C,1,1);

> Z2Z4Equal(Z2Z4Sum(C1,C2), C3);
true
> Z2Z4Equal(Z2Z4Intersection(C1,C3), C1);
true
```

1.8.3 Standard Constructions

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, which is represented in Magma as a subspace of $V = \mathbb{Z}_4^{\alpha+\beta}$, any codeword u belonging to C can be written as $u = (u_1|u_2)$, where $u_1 \in \mathbb{Z}_4^\alpha$ and $u_2 \in \mathbb{Z}_4^\beta$.

`Z2Z4DirectSum(C, D)`

Given $\mathbb{Z}_2\mathbb{Z}_4$-additive codes C and D, construct the direct sum of C and D. The direct sum is a $\mathbb{Z}_2\mathbb{Z}_4$-additive code that consists of all vectors of the form $(u_1, v_1|u_2, v_2)$, where $(u_1|u_2) \in C$ and $(v_1|v_2) \in D$.

`Z2Z4Concatenation(C, D)`

Given $\mathbb{Z}_2\mathbb{Z}_4$-additive codes C and D, return the concatenation of C and D. If $G_c = (A_\alpha|A_\beta)$ and $G_d = (B_\alpha|B_\beta)$ are the generator matrices of C and D, respectively, the concatenation of C and D is the $\mathbb{Z}_2\mathbb{Z}_4$-additive code with generator matrix whose rows consist of each row $(a_\alpha|a_\beta)$ of G_c concatenated with each row $(b_\alpha|b_\beta)$ of G_d in the following way: $(a_\alpha, b_\alpha|a_\beta, b_\beta)$.

`Z2Z4PlotkinSum(C, D)`

Given $\mathbb{Z}_2\mathbb{Z}_4$-additive codes C and D both with the same parameters α and β, hence also the same length, construct the Plotkin sum of C and D. The Plotkin sum is a $\mathbb{Z}_2\mathbb{Z}_4$-additive code that consists of all vectors of the form $(u_\alpha, u_\alpha + v_\alpha|u_\beta, u_\beta + v_\beta)$, where $(u_\alpha|u_\beta) \in C$ and $(v_\alpha|v_\beta) \in D$.

`Z2Z4QuaternaryPlotkinSum(A, B)`

Given $\mathbb{Z}_2\mathbb{Z}_4$-additive codes A and B both with the same parameters $\alpha = 0$ and β, hence also the same length, construct the Quaternary Plotkin sum of A and B. The Quaternary Plotkin sum is a $\mathbb{Z}_2\mathbb{Z}_4$-additive code that consists of all vectors of the form $(u, u + v, u + 2v, u + 3v)$, where $u \in A$ and $v \in B$.

`Z2Z4DoublePlotkinSum(A, B, C, D)`

Given $\mathbb{Z}_2\mathbb{Z}_4$-additive codes A, B, C and D all with the same parameters $\alpha = 0$ and β, hence also the same length, construct the Double Plotkin sum of A, B, C and D. The Double Plotkin sum is a $\mathbb{Z}_2\mathbb{Z}_4$-additive code that consists of all vectors of the form $(u, u+v, u+2v+z, u+3v+z+t)$, where $u \in A$, $v \in B$, $z \in C$ and $t \in D$.

`Z2Z4BQPlotkinSum(A, B, C)`

Given $\mathbb{Z}_2\mathbb{Z}_4$-additive codes A, B and C all with the same parameters $\alpha = 0$ and β, hence also the same length, construct the BQ Plotkin sum of A, B and C. Let G_B be a generator matrix of the $\mathbb{Z}_2\mathbb{Z}_4$-additive code B of type $(0, \beta; \gamma, \delta; \kappa)$. The $\mathbb{Z}_2\mathbb{Z}_4$-additive code B' is obteained from B switching twos by ones in the γ rows of order two of G_B and the $\mathbb{Z}_2\mathbb{Z}_4$-additive code $\hat{B}$ is obtained from B removing the γ rows of order two of G_B.

The BQ Plotkin sum is a $\mathbb{Z}_2\mathbb{Z}_4$-additive code that consists of all vectors of the form $(u, u + v', u + 2v' + \hat{v}, u + 3v' + \hat{v} + z)$, where $u \in A$, $v' \in B'$ $\hat{v} \in \hat{B}$, and $z \in C$.

`Z2Z4PunctureCode(C, i)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C and an integer i, $1 \leq i \leq \alpha + \beta$, construct a new $\mathbb{Z}_2\mathbb{Z}_4$-additive code C' by deleting the i-th coordinate from each codeword of C.

`Z2Z4PunctureCode(C, S)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C and a set S of distinct integers $\{i_1, \ldots, i_r\}$ each of which lies in the range $[1, \alpha + \beta]$, construct a new $\mathbb{Z}_2\mathbb{Z}_4$-additive code C' by deleting the components $i_1, \ldots, i_r$ from each codeword of C.

`Z2Z4ShortenCode(C, i)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C and an integer i, $1 \leq i \leq \alpha + \beta$, construct a new $\mathbb{Z}_2\mathbb{Z}_4$-additive code from C by selecting only those codewords of C having a zero as their i-th component and deleting the i-th component from these codewords.

`Z2Z4ShortenCode(C, S)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C and a set S of distinct integers $\{i_1, \ldots, i_r\}$ each of which lies in the range $[1, \alpha + \beta]$, construct a new $\mathbb{Z}_2\mathbb{Z}_4$-additive code from C by selecting only those codewords of C having zeros in each of the coordinate positions $i_1, \ldots, i_r$, and deleting these components.

`Z2Z4ExtendCode(C)`

Given a $\mathbb{Z}_2\mathbb{Z}_4$-additive code C form a new $\mathbb{Z}_2\mathbb{Z}_4$-additive code C' from C by adding the appropriate extra binary coordinate to each codeword v of C such that $\Phi(v)$ has even Hamming weight, where Φ is the Gray map defined in Subsection 1.5.1.

`Z2Z4Kernel(C)`

Given a code $\mathbb{Z}_2\mathbb{Z}_4$-additive code C, returns its kernel as a lineal $\mathbb{Z}_2\mathbb{Z}_4$-additive code. The kernel of C are the codewords v such that $\phi(v)$ + C_b = C_b, where $C_b = \Phi(C)$ is the corresponding binary code and Φ is the Gray map defined in Subsection 1.5.1.

Example H1E15

We combine $\mathbb{Z}_2\mathbb{Z}_4$-additive codes in diferent ways and look at the parameters α and β of the new $\mathbb{Z}_2\mathbb{Z}_4$-additive codes.

```
> C1:=RandomZ2Z4AdditiveCode([2,3]);
> C2:=RandomZ2Z4AdditiveCode([1,4]);
> Z2Z4Length(C1);
5 [ 2, 3 ]
> Z2Z4Length(C2);
5 [ 1, 4 ]

> C3:=Z2Z4DirectSum(C1,C2);
> Z2Z4Length(C3);
10 [ 3, 7 ]
> C4:=Z2Z4Concatenation(C1,C2);
> Z2Z4Length(C4);
10 [ 3, 7 ]
> Z2Z4Subset(C4,C3);
true

> C5:=Z2Z4PunctureCode(C1,3);
> Z2Z4Length(C5);
4 [ 2, 2 ]
```

Bibliography

[1] J. Borges, C. Fernández, J. Pujol, J. Rifà and M. Villanueva, "On $\mathbb{Z}_2\mathbb{Z}_4$-linear codes and duality", *V Jornadas de Matemática Discreta y Algorítmica*, Soria (Spain), July 11-14, pp. 171-177, 2006.

[2] J. Borges, C. Fernández, J. Pujol, J. Rifà and M. Villanueva, "$\mathbb{Z}_2\mathbb{Z}_4$-linear codes: generator matrices and duality", submitted to *IEEE Trans. on Information Theory*, 2007. arXiv:0710.1149

[3] J. J. Cannon and W. Bosma (Eds.) *Handbook of* MAGMA *Functions*, Edition 2.13, 4350 pages, 2006.

[4] P. Delsarte, "An algebraic approach to the association schemes of coding theory", *Philips Research Rep. Suppl.*, vol. 10, 1973.

[5] J. A. Howell, "Spans in the module $\mathbb{Z}_m^s$". *Linear and Multilinear Algebra*, 19, pp. 67-77, 1986.

[6] A. Storjohann and T. Mulders, "Fast algorithms for linear algebra modulo N", *Lecture Notes In Computer Science*, vol. 1461, pp. 139-150, 1998.