CODES OVER $\mathbb{Z}_4$

A MAGMA Package[1]

by

Jaume Pernas, Jaume Pujol and Mercè Villanueva

Combinatoric, Coding and Security Group (CCSG)

Universitat Autònoma de Barcelona

Version 1.4

Barcelona
March 5, 2012

[1]This work has been partially supported by the Spanish MICINN under Grants PCI2006-A7-0616, MTM2006-03250, MTM2009-08435 and TIN2010-17358; and by the Catalan AGAUR under Grant 2009SGR1224.

Contents

Chapter 1

Preface

The *Combinatoric, Coding and Security Group* (CCSG) is a research group in the Department of Information and Communications Engineering (DEIC) at the Universitat Autònoma de Barcelona (UAB).

The research group CCSG has been uninterruptedly working since 1987 in several projects and research activities on Information Theory, Communications, Coding Theory, Source Coding, Cryptography, Electronic Voting, Network Coding, etc. The members of the group have been producing mainly results on optimal coding. Specifically, the research has been focused on uniformly-packed codes; perfect codes in the Hamming space; perfect codes in distance-regular graphs; the classification of optimal codes of a given length; and codes which are close to optimal codes by some properties, for example, Reed-Muller codes, Preparata codes, Kerdock codes and Hadamard codes.

Part of the research developed by CCSG deals with codes over $\mathbb{Z}_4$. Some members of CCSG have been developing this new package that expands the current functionality for codes over $\mathbb{Z}_4$ in MAGMA. MAGMA is a software package designed to solve computationally hard problems in algebra, number theory, geometry and combinatorics. A beta version of this new package for codes over $\mathbb{Z}_4$ and this manual with the description of all developed functions can be downloaded from the web page `http://ccsg.uab.es`. For any comment or further information about this package, you can send an e-mail to *support-ccsg@deic.uab.cat*.

The authors would like to thank Lorena Ronquillo and Bernat Gastón for their contributions developing part of this MAGMA package.

Chapter 2

Codes over $\mathbb{Z}_4$

2.1 Introduction

MAGMA currently supports the basic facilities for linear codes over integer residue rings and galois rings (see [3, Chapter 130]), including additional functionality for the special case of codes over $\mathbb{Z}_4$ (or equivalently, quaternary linear codes). A code over $\mathbb{Z}_4$ is a subgroup of $\mathbb{Z}_4^n$, so it is isomorphic to an abelian structure $\mathbb{Z}_2^\gamma \times \mathbb{Z}_4^\delta$ and we will say that it is of type $2^\gamma 4^\delta$, or simply that it has $2^{\gamma+2\delta}$ codewords. As general references on the available functions in MAGMA for codes over $\mathbb{Z}_4$, the reader is referred to [7, 16].

The functions described in this chapter expand the current functionality for codes over $\mathbb{Z}_4$ in MAGMA. Specifically, there are functions which give new constructions for some families of codes over $\mathbb{Z}_4$ and constructions to obtain new codes over $\mathbb{Z}_4$ from given codes over $\mathbb{Z}_4$ (Sections 2.2 and 2.3). Moreover, efficient functions for computing the rank and dimension of the kernel of any code over $\mathbb{Z}_4$ are also included (Section 2.4), as well as general functions to compute the coset leaders for a subcode in a code over $\mathbb{Z}_4$ (Section 2.5). Finally, there are also functions to compute the permutation automorphism group for Hadamard and extended perfect codes over $\mathbb{Z}_4$, and their cardinal (Section 2.6). As general references on these new functions, the reader is referred to [5, 6, 9, 12, 13, 14, 10, 11].

In this chapter the term "code" will refer to a code over $\mathbb{Z}_4$, unless otherwise specified.

2.2 Families of Codes over $\mathbb{Z}_4$

These functions give some constructions for some families of codes over $\mathbb{Z}_4$.

`HadamardCodeZ4(δ, m)`

Given an integer $m \geq 1$ and an integer δ such that $1 \leq \delta \leq \lfloor (m+1)/2 \rfloor$, return a Hadamard code over $\mathbb{Z}_4$ of length 2^{m-1} and type $2^\gamma 4^\delta$, where $\gamma = m + 1 - 2\delta$. Moreover, return a generator matrix with $\gamma + \delta$ rows constructed in a recursive way from the Plotkin and BQPlotkin constructions defined in Section 2.3.

A Hadamard code over $\mathbb{Z}_4$ of length 2^{m-1} is a code over $\mathbb{Z}_4$ such that, after the Gray map, give a binary (not necessarily linear) code with the same parameters as the binary Hadamard code of length 2^m.

`ExtendedPerfectCodeZ4(δ, m)`

Given an integer $m \geq 2$ and an integer δ such that $1 \leq \delta \leq \lfloor (m+1)/2 \rfloor$, return an extended perfect code over $\mathbb{Z}_4$ of length 2^{m-1}, such that its dual code is of type $2^\gamma 4^\delta$, where $\gamma = m + 1 - 2\delta$. Moreover, return a generator matrix constructed in a recursive way from the Plotkin and BQPlotkin constructions defined in Section 2.3.

An extended perfect code over $\mathbb{Z}_4$ of length 2^{m-1} is a code over $\mathbb{Z}_4$ such that, after the Gray map, give a binary (not necessarily linear) code with the same parameters as the binary extended perfect code of length 2^m.

Example H2E1

We compute codes over $\mathbb{Z}_4$ such that, after the Gray map, they are binary codes with the same parameters as some well-known families of binary linear codes.

First, we define a Hadamard code C over $\mathbb{Z}_4$ of length 8 and type $2^1 4^2$. The matrix Gc is the quaternary matrix used to generate C and obtained in a recursive way from Plotkin and BQPlotkin constructions.

```
> C, Gc := HadamardCodeZ4(2,4);
> C;
((8, 4^2 2^1)) Linear Code over IntegerRing(4)
Generator matrix:
[1 0 3 2 1 0 3 2]
[0 1 2 3 0 1 2 3]
[0 0 0 0 2 2 2 2]
> Gc;
[1 1 1 1 1 1 1 1]
[0 1 2 3 0 1 2 3]
[0 0 0 0 2 2 2 2]
> HasLinearGrayMapImage(C);
true [16, 5, 8] Linear Code over GF(2)
Generator matrix:
[1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 0]
[0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1]
[0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1]
```

```
[0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0]
[0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1]
Mapping from: CodeLinRng: C to [16, 5, 8] Linear Code over GF(2) given by a rule
```

Then, we define an extended perfect code D over $\mathbb{Z}_4$ of length 8, such that its dual code is of type $2^1 4^2$. The matrix Gd is the quaternary matrix which is used to generate D and obtained in a recursive way from Plotkin and BQPlotkin constructions. Note that the code D is the Kronecker dual code of C.

```
> D, Gd := ExtendedPerfectCodeZ4(2,4);
> D;
((8, 4^5 2^1)) Linear Code over IntegerRing(4)
Generator matrix:
[1 0 0 1 0 0 1 3]
[0 1 0 1 0 0 2 2]
[0 0 1 1 0 0 1 1]
[0 0 0 2 0 0 0 2]
[0 0 0 0 1 0 3 2]
[0 0 0 0 0 1 2 3]
> Gd;
[1 1 1 1 1 1 1 1]
[0 1 2 3 0 1 2 3]
[0 0 1 1 0 0 1 1]
[0 0 0 2 0 0 0 2]
[0 0 0 0 1 1 1 1]
[0 0 0 0 0 1 2 3]

> DualKroneckerZ4(C) eq D;
true
```

`ReedMullerCodeZ4(r, m)`

`ReedMullerCodeQRMZ4(r, m)`

Given an integer $m \geq 2$ and an integer r such that $0 \leq r \leq m$, return the r-th order Reed-Muller code over $\mathbb{Z}_4$ of length 2^m.

The binary image under the modulo 2 map is the binary linear r-th order Reed-Muller code of length 2^m. For $r = 1$ and $r = m - 2$, the function returns the quaternary linear Kerdock and Preparata code, respectively.

`ReedMullerCodesLRMZ4(r, m)`

Given an integer $m \geq 1$ and an integer r such that $0 \leq r \leq m$, return a set of r-th order Reed-Muller codes over $\mathbb{Z}_4$ of length 2^{m-1}.

The binary image under the Gray map of any of these codes is a binary (not necessarily linear) code with the same parameters as the binary linear r-th order Reed-Muller code of length 2^m. Note that for these codes neither the usual inclusion nor duality properties of the binary linear Reed-Muller family are satisfied.

`ReedMullerCodeRMZ4(s, r, m)`

Given an integer $m \geq 1$, an integer r such that $0 \leq r \leq m$, and an integer s such that $0 \leq s \leq \lfloor (m-1)/2 \rfloor$, return a r-th order Reed-Muller code over $\mathbb{Z}_4$ of length 2^{m-1}, denoted by $RM_s(r, m)$, as well as the generator matrix used in the recursive construction.

The binary image under the Gray map is a binary (not necessarily linear) code with the same parameters as the binary linear r-th order Reed-Muller code of length 2^m. Note that the inclusion and duality properties are also satisfied, that is, the code $RM_s(r-1, m)$ is a subcode of $RM_s(r, m)$, $r > 0$, and the code $RM_s(r, m)$ is the Kronecker dual code of $RM_s(m-r-1, m)$, $r < m$.

Example H2E2

We define $RM_1(1, 4)$ and $RM_1(2, 4)$. We can see that the former is a subcode of the latter. Note that $RM_1(1, 4)$ and $RM_1(2, 4)$ are the same as the ones given in Example H2E1 by `HadamardCodeZ4(2,4)` and `ExtendedPerfectCodeZ4(2,4)`, respectively.

```
> C1, G1 := ReedMullerCodeRMZ4(1,1,4);
> C2, G2 := ReedMullerCodeRMZ4(1,2,4);

> C1;
((8, 4^2 2^1)) Linear Code over IntegerRing(4)
Generator matrix:
[1 0 3 2 1 0 3 2]
[0 1 2 3 0 1 2 3]
[0 0 0 0 2 2 2 2]
> C2;
((8, 4^5 2^1)) Linear Code over IntegerRing(4)
Generator matrix:
[1 0 0 1 0 0 1 3]
[0 1 0 1 0 0 2 2]
[0 0 1 1 0 0 1 1]
[0 0 0 2 0 0 0 2]
[0 0 0 0 1 0 3 2]
[0 0 0 0 0 1 2 3]

> C1 subset C2;
true
> DualKroneckerZ4(C2) eq C1;
true
```

`ReedMullerCodesRMZ4(s, m)`

Given an integer $m \geq 1$, and an integer s such that $0 \leq s \leq \lfloor (m-1)/2 \rfloor$, return a sequence with the family of Reed-Muller codes over $\mathbb{Z}_4$ of length 2^{m-1}, that is, the codes $RM_s(r, m)$, for all $0 \leq r \leq m$.

The binary image of these codes under the Gray map gives a family of binary (not necessarily linear) codes with the same parameters as the binary linear Reed-Muller family of codes of length 2^m. Note that $RM_s(0, m) \subset RM_s(1, m) \subset \cdots \subset RM_s(m, m)$.

Example H2E3

We construct the family of Reed-Muller codes over $\mathbb{Z}_4$ of length 2^2 given by $s = 0$.

```
> F := ReedMullerCodesRMZ4(0,3);
> F;
[((4, 4^0 2^1)) Cyclic Linear Code over IntegerRing(4)
Generator matrix:
[2 2 2 2],
((4, 4^1 2^2)) Cyclic Linear Code over IntegerRing(4)
Generator matrix:
[1 1 1 1]
[0 2 0 2]
[0 0 2 2],
((4, 4^3 2^1)) Cyclic Linear Code over IntegerRing(4)
Generator matrix:
[1 0 0 1]
[0 1 0 1]
[0 0 1 1]
[0 0 0 2],
((4, 4^4 2^0)) Cyclic Linear Code over IntegerRing(4)
Generator matrix:
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]]

> F[1] subset F[2] and F[2] subset F[3] and F[3] subset F[4];
true
```

2.3 Constructing New Codes from Old

The functions described here produce a new code over $\mathbb{Z}_4$ by modifying in some way the codewords of some given codes over $\mathbb{Z}_4$.

`PlotkinSum(A, B)`

Given matrices A and B both over the same ring and with the same number of columns, return the P_{AB} matrix over the same ring of A and B, where

$$P_{AB} = \begin{pmatrix} A & A \\ 0 & B \end{pmatrix}.$$

`PlotkinSum(C, D)`

Given codes C and D both over the same ring and of the same length, construct the Plotkin sum of C and D. The Plotkin sum consists of all vectors of the form $(u|u+v)$, where $u \in C$ and $v \in D$.

Note that the Plotkin sum is computed using generator matrices of C and D and the `PlotkinSum` function for matrices, that is, this function returns the code over $\mathbb{Z}_4$ generated by the matrix P_{AB} defined above, where A and B are generators matrices of C and D, respectively.

`QuaternaryPlotkinSum(A, B)`

Given two matrices A and B over $\mathbb{Z}_4$, both with the same number of columns, return the QP_{AB} matrix over $\mathbb{Z}_4$, where

$$QP_{AB} = \begin{pmatrix} A & A & A & A \\ 0 & B & 2B & 3B \end{pmatrix}.$$

`QuaternaryPlotkinSum(C, D)`

Given two codes C and D over $\mathbb{Z}_4$, both of the same length, construct the Quaternary Plotkin sum of C and D. The Quaternary Plotkin sum is a code over $\mathbb{Z}_4$ that consists of all vectors of the form $(u, u+v, u+2v, u+3v)$, where $u \in C$ and $v \in D$.

Note that the Quaternary Plotkin sum is computed using generator matrices of C and D and the `QuaternaryPlotkinSum` function for matrices, that is, this function returns the code over $\mathbb{Z}_4$ generated by the matrix QP_{AB} defined above, where A and B are generators matrices of C and D, respectively.

`BQPlotkinSum(A, B, C)`

Given three matrices A, B, and C over $\mathbb{Z}_4$, all with the same number of columns, return the BQP_{ABC} matrix over $\mathbb{Z}_4$, where

$$BQP_{ABC} = \begin{pmatrix} A & A & A & A \\ 0 & B' & 2B' & 3B' \\ 0 & 0 & \hat{B} & \hat{B} \\ 0 & 0 & 0 & C \end{pmatrix},$$

B' is obtained from B replacing the twos with ones in the rows of order two, and $\hat{B}$ is obtained from B removing the rows of order two.

`BQPlotkinSum(D, E, F)`

Given three codes D, E and F over $\mathbb{Z}_4$, all of the same length, construct the BQ Plotkin sum of D, E and F. Let Ge be a generator matrix of the code E of type $2^\gamma 4^\delta$. The code E' over $\mathbb{Z}_4$ is obtained from E replacing the twos with ones in the γ rows of order two of Ge, and the code $\hat{E}$ over $\mathbb{Z}_4$ is obtained from E removing the γ rows of order two of Ge.
The BQ Plotkin sum is a code over $\mathbb{Z}_4$ that consists of all vectors of the form $(u, u+v', u+2v'+\hat{v}, u+3v'+\hat{v}+z)$, where $u \in Gd$, $v' \in Ge'$ $\hat{v} \in \hat{G}e$, and $z \in Gf$, where Gd, Ge', $\hat{G}e$ and Gf are generators matrices of D, E', $\hat{E}$ and F, respectively.

Note that the BQPlotkin sum is computed using generator matrices of D, E and F and the `BQPlotkinSum` function for matrices. However, this function does not necessarily return the same code over $\mathbb{Z}_4$ generated by the matrix BQP_{ABC} defined above, where A, B and C are generators matrices of D, E and F, respectively, as shown in Example H2E4.

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

Given four matrices A, B, C, and D over $\mathbb{Z}_4$, all with the same number of columns, return the DP_{ABC} matrix over $\mathbb{Z}_4$, where

$$DP_{ABCD} = \begin{pmatrix} A & A & A & A \\ 0 & B & 2B & 3B \\ 0 & 0 & C & C \\ 0 & 0 & 0 & D \end{pmatrix}.$$

`DoublePlotkinSum(E, F, G, H)`

Given four codes E, F, G and H over $\mathbb{Z}_4$, all of the same length, construct the Double Plotkin sum of E, F, G and H. The Double Plotkin sum is a code over $\mathbb{Z}_4$ that consists of all vectors of the form $(u, u+v, u+2v+z, u+3v+z+t)$, where $u \in E$, $v \in F$, $z \in G$ and $t \in H$.

Note that the Double Plotkin sum is computed using generator matrices of E, F, G and H and the `DoublePlotkinSum` function for matrices, that is, this function returns the code over $\mathbb{Z}_4$ generated by the matrix DP_{ABCD} defined above, where A, B, C and D are generators matrices of E, F, G and H, respectively.

`DualKronecker(C)`

Given a code C over $\mathbb{Z}_4$ of length 2^m, return its Kronecker dual code. The Kronecker dual code of C is $C_{\otimes}^{\perp} = \{x \in \mathbb{Z}_4^{2^m} : x \cdot K_{2^m} \cdot y^t = 0, \forall y \in C\}$, where $K_{2^m} = \otimes_{j=1}^{m} K_2$, $K_2 = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$ and $\otimes$ denotes the Kronecker product of matrices. Equivalently, K_{2^m} is a quaternary matrix of length 2^m with the vector $(1, 3, 3, 1, 3, 1, 1, 3, \ldots)$ in the main diagonal and zeros elsewhere.

Example H2E4

In this example, we show that the codes over $\mathbb{Z}_4$ constructed from the `BQPlotkinSum` function for matrices are not necessarily the same as the ones constructed from the `BQPlotkinSum` function for codes.

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

> Ga := Matrix(Z4,1,2,[1,1]);
> Gb := Matrix(Z4,2,2,[1,2,0,2]);
> Gc := Matrix(Z4,1,2,[2,2]);

> Ca := LinearCode(Ga);
> Cb := LinearCode(Gb);
> Cc := LinearCode(Gc);

> C := LinearCode(BQPlotkinSum(Ga,Gb,Gc));
> D := BQPlotkinSum(Ca,Cb,Cc);
> C eq D;
false
```

Example H2E5

```
> Ga := GeneratorMatrix(ReedMullerCodeRMZ4(1,2,3));
> Gb := GeneratorMatrix(ReedMullerCodeRMZ4(1,1,3));
```

```
> Gc := GeneratorMatrix(ReedMullerCodeRMZ4(1,0,3));

> C := ReedMullerCodeRMZ4(1,2,4));
> Cp := LinearCode(PlotkinSum(Ga, Gb));
> C eq Cp;
true

> D := ReedMullerCodeRMZ4(2,2,5);
> Dp := LinearCode(BQPlotkinSum(Ga, Gb, Gc));
> D eq Dp;
true
```

2.4 Invariants of Codes over $\mathbb{Z}_4$

`MinRowsGeneratorMatrix(C)`

A generator matrix for the code C over $\mathbb{Z}_4$ of length n and type $2^\gamma 4^\delta$, with the minimum number of rows, that is with $\gamma + \delta$ rows: γ rows of order two and δ rows of order four. It also returns the parameters γ and δ.

`SpanZ2CodeZ4(C)`

Given a code C over $\mathbb{Z}_4$ of length n, return $S_C = \Phi^{-1}(S_{bin})$ as a code over $\mathbb{Z}_4$, and the linear span of C_{bin}, $S_{bin} = \langle C_{bin} \rangle$, as a binary linear code of length $2n$, where $C_{bin} = \Phi(C)$ and Φ is the Gray map.

`KernelZ2CodeZ4(C)`

Given a code C over $\mathbb{Z}_4$ of length n, return its kernel K_C as a subcode over $\mathbb{Z}_4$ of C, and $K_{bin} = \Phi(K_C)$ as a binary linear subcode of C_{bin} of length $2n$, where $C_{bin} = \Phi(C)$ and Φ is the Gray map.

The kernel K_C contains the codewords v such that $2v*u \in C$ for all $u \in C$, where $*$ denotes the component-wise product. Equivalently, the kernel $K_{bin} = \Phi(K_C)$ contains the codewords $c \in C_{bin}$ such that $c + C_{bin} = C_{bin}$, where $C_{bin} = \Phi(C)$ and Φ is the Gray map.

`KernelCosetLeaders(C)`

Given a code C over $\mathbb{Z}_4$ of length n, return the coset leaders $[c_1, \ldots, c_t]$ as a sequence of codewords of C, such that $C = K_C \cup \bigcup_{i=1}^{t} \left(K_C + c_i\right)$, where K_C is the kernel of C as a subcode over $\mathbb{Z}_4$. It also returns the coset leaders of the corresponding binary code $C_{bin} = \Phi(C)$ as a sequence of binary codewords $[\Phi(c_1), \ldots, \Phi(c_t)]$, such that $C_{bin} = K_{bin} \cup \bigcup_{i=1}^{t} \left(K_{bin} + \Phi(c_i)\right)$, where $K_{bin} = \Phi(K_C)$ and Φ is the Gray map.

`DimensionOfSpanZ2(C)`

`RankZ2(C)`

Given a code C over $\mathbb{Z}_4$, return the dimension of the linear span of C_{bin}, that is, the dimension of $\langle C_{bin} \rangle$, where $C_{bin} = \Phi(C)$ and Φ is the Gray map.

`DimensionOfKernelZ2(C)`

Given a code C over $\mathbb{Z}_4$, return the dimension of the Gray map image of its kernel K_C over $\mathbb{Z}_4$, that is the dimension of $K_{bin} = \Phi(K_C)$, where Φ is the Gray map. Note that K_{bin} is always a binary linear code.

Example H2E6

```
> C := ReedMullerCodeRMZ4(0,3,5);

> DimensionOfKernelZ2(C);
20
> DimensionOfSpanZ2(C);
27

> K, Kb := KernelZ2CodeZ4(C);
> S, Sb := SpanZ2CodeZ4(C);

> K subset C;
true
> C subset S;
true

> Dimension(Kb) eq DimensionOfKernelZ2(C);
true
> Dimension(Sb) eq DimensionOfSpanZ2(C);
true
```

2.5 Coset Leaders

`CosetLeaders(C)`

Given a code C over $\mathbb{Z}_4$ of length n, with ambient space $V = \mathbb{Z}_4^n$, return a set of coset leaders (not necessarily of minimal weight in their cosets) for C in V as an indexed set of vectors from V. The set of coset leaders $\{c_0, c_1, \ldots, c_t\}$ satisfies that c_0 is the zero codeword, and $V = \bigcup_{i=0}^{t} \big(C + c_i\big)$. Note that this function is only applicable when V and C are small.

`CosetLeaders(C,S)`

Given a code C over $\mathbb{Z}_4$ of length n, and a subcode S over $\mathbb{Z}_4$ of C, return a set of coset leaders (not necessarily of minimal weight in their cosets) for S in C as an indexed set of codewords from C. The set of coset leaders $\{c_0, c_1, \ldots, c_t\}$ satisfies that c_0 is the zero codeword, and $C = \bigcup_{i=0}^{t} \big(S + c_i\big)$. Note that this function is only applicable when S and C are small.

Example H2E7

```
> C := LinearCode<Integers(4),4 | [[1,0,0,3],[0,1,1,3]]>;
> L := CosetLeaders(C);
> Set(RSpace(Integers(4),4)) eq {v+ci : v in Set(C), ci in L};
true

> K := KernelZ2CodeZ4(C);
> L := CosetLeaders(C,K);
> {C!0} join Set(KernelCosetLeadersZ4(C)) eq L;
true
> Set(C) eq {v+ci : v in Set(K), ci in L};
true
```

2.6 Automorphism Groups

`PermutationGroupHadamardCodeZ4(δ, m)`

`PAutHadamardCodeZ4(δ, m)`

Given an integer $m \geq 1$ and an integer δ such that $1 \leq \delta \leq \lfloor (m+1)/2 \rfloor$, return the permutation group G of a Hadamard code over $\mathbb{Z}_4$ of length 2^{m-1} and type $2^\gamma 4^\delta$, where $\gamma = m + 1 - 2\delta$. The group G contains all permutation-action permutations which preserve the code. Thus only permutation of coordinates is allowed, and the degree of G is always 2^{m-1}. Moreover, return the generator matrix with $\gamma + \delta$ rows used to generate the code and constructed in a recursive way from the Plotkin and BQPlotkin constructions defined in Section 2.3.

`CardinalPermutationGroupHadamardCodeZ4(δ, m)`

`CardinalPAutHadamardCodeZ4(δ, m)`

Given an integer $m \geq 1$ and an integer δ such that $1 \leq \delta \leq \lfloor (m+1)/2 \rfloor$, return the cardinal of the permutation group G of a Hadamard code over $\mathbb{Z}_4$ of length 2^{m-1} and type $2^\gamma 4^\delta$, where $\gamma = m + 1 - 2\delta$. The group G contains all permutation-action permutations which preserve the code.

`PermutationGroupExtendedPerfectCodeZ4(δ, m)`

`PAutExtendedPerfectCodeZ4(δ, m)`

Given an integer $m \geq 2$ and an integer δ such that $1 \leq \delta \leq \lfloor (m+1)/2 \rfloor$, return the permutation group G of an extended perfect code over $\mathbb{Z}_4$ of length 2^{m-1}, such that its dual code is of type $2^\gamma 4^\delta$, where $\gamma = m + 1 - 2\delta$. The group G contains all permutation-action permutations which preserve the code. Thus only permutation of coordinates is allowed, and the degree of G is always 2^{m-1}. Moreover, return the generator matrix with $\gamma + \delta$ rows used to generate the code and constructed in a recursive way from the Plotkin and BQPlotkin constructions defined in Section 2.3.

`CardinalPermutationGroupExtendedPerfectCodeZ4(δ, m)`

`CardinalPAutExtendedPerfectCodeZ4(δ, m)`

Given an integer $m \geq 2$ and an integer δ such that $1 \leq \delta \leq \lfloor (m+1)/2 \rfloor$, return the permutation group G of an extended perfect code over $\mathbb{Z}_4$ of length 2^{m-1}, such that its dual code is of type $2^\gamma 4^\delta$, where $\gamma = m + 1 - 2\delta$. The group G contains all permutation-action permutations which preserve the code.

Example H2E8

```
> C := HadamardCodeZ4(2,4);
> PAut := PAutHadamardCodeZ4(2,4);
> #[ Set(C^p) eq Set(C) : p in PAut] eq #PAut;
true
> #PAut eq CardinalPAutHadamardCodeZ4(2,4);
true

> PAutHadamardCodeZ4(2,4) eq PAutExtendedPerfectCodeZ4(2,4);
true
```

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-Córdoba, J. Pujol, J. Rifà and M. Villanueva, "$\mathbb{Z}_2\mathbb{Z}_4$-linear codes: generator matrices and duality," *Designs, Codes and Cryptography*, vol 54, no. 2, pp. 167-179, 2010.

[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] C. Fernández-Córdoba, J. Pujol, and M. Villanueva, "On rank and kernel of $\mathbb{Z}_4$-linear codes," *Lecture Notes in Computer Science*, n. 5228, 2008.

[6] C. Fernández-Córdoba, J. Pujol, and M. Villanueva, "$\mathbb{Z}_2\mathbb{Z}_4$-linear codes: rank and kernel," *Designs, Codes and Cryptography*, vol 56, no. 1, pp. 43-59, 2010.

[7] A.R. Hammons, P.V. Kumar, A.R. Calderbank, N.J.A. Sloane and P. Solé, "The $\mathbb{Z}_4$-linearity of kerdock, preparata, goethals and related codes," *IEEE Trans. on Information Theory*, vol. 40, pp. 301-319, 1994.

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

[9] D. S. Krotov, "$\mathbb{Z}_4$-linear Hadamard and extended perfect codes," *Proc. of the International Workshop on Coding and Cryptography*, Paris (France), Jan. 8-12, pp. 329–334, 2001.

[10] J. Pernas, J. Pujol, and M. Villanueva, "On the Permutation Automorphism Group of Quaternary Linear Hadamard Codes," *Proc. of 3rd International Castle Meeting on Coding Theory and Applications*, Cardona (Spain), Sep. 11-15, pp. 213–218, 2011.

[11] J. Pernas, J. Pujol, and M. Villanueva, "Characterization of the Automorphism Group of Quaternary Linear Hadamard Codes," accepted for publication on *Designs, Codes and Cryptography*, 2012.

[12] J. Pujol, J. Rifà, F. I. Solov'eva, "Quaternary Plotkin constructions and Quaternary Reed-Muller codes," *Lecture Notes in Computer Science* n. 4851, pp. 148-157, 2007.

[13] J. Pujol, J. Rifà and F. I. Solov'eva "Construction of $\mathbb{Z}_4$-linear Reed-Muller codes," *IEEE Trans. on Information Theory*, vol. 55, no. 1, pp. 99-104, 2009.

[14] F. I. Solov'eva "On $\mathbb{Z}_4$-linear codes with parameters of Reed-Muller codes," *Problems of Information Transmission*, vol. 43, no. 1, pp. 26-32, 2007.

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

[16] Z.-X. Wan, *Quaternary Codes*, World Scientific, 1997.