ÖAW Austrian Academy of Sciences

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

JKU JOHANNES KEPLER UNIVERSITY LINZ

# Computing coset leaders and leader codewords of binary codes

M. Borges-Quintana  M.A. Borges-Trenard

I. Márquez-Corbella  E. Martínez-Moro

DK-Report No. 2012-07  05 2012

A–4040 LINZ, ALTENBERGERSTRASSE 69, AUSTRIA

Supported by

Austrian Science Fund (FWF)  Upper Austria

FWF Der Wissenschaftsfonds.

LAND OBERÖSTERREICH

DK sponsors:

- **Johannes Kepler University Linz** (JKU)
- **Austrian Science Fund** (FWF)
- **Upper Austria**

# Computing coset leaders and leader codewords of binary codes

M. Borges-Quintana[†] M.A. Borges-Trenard[†]

I. Márquez-Corbella[‡]

E. Martínez-Moro[‡]

[†] Department of Mathematics, Faculty of Mathematics and Computer Science.

Universidad de Oriente, Santiago de Cuba, Cuba.

mijail@csd.uo.edu.cu, mborges@csd.uo.edu.cu

[‡] SINGACOM group. Universidad de Valladolid.

Castilla, Spain.

http://www.singacom.uva.es

imarquez@agt.uva.es, edgar@maf.uva.es

December, 2011

**Abstract**

We present an algorithm for computing the set of all coset leaders of a binary code $\mathscr{C} \subset \mathbb{F}_2^n$. The method is adapted from some of the techniques related to the computation of Gröbner representations associated with codes. The algorithm provides a Gröbner representation of the binary code and the set of coset leaders $\mathrm{CL}(\mathscr{C})$. Its efficiency stands on the fact that its complexity is linear on the number of elements of $\mathrm{CL}(\mathscr{C})$, which is smaller than exhaustive search in $\mathbb{F}_2^n$. It is also shown that the same algorithm could provide a set of codewords which form a test set for decoding by a gradient-like decoding algorithm.

**Keywords** Binary codes Cosets leaders Test set Gröbner representations.

# 1 Introduction

Section 4 of this work was developed during a stay of Borges-Quintana (October-November 2011) at RISC-Linz. Borges-Quintana would like to thank the invitation of Prof. Dr. Franz Winkler and the DK-Doctoral Program that partially founded the research stay. The RISC's sysadmin were also very helpful in giving me very useful hints for using the advantage of some powerful computers available in RISC, which were necessary for some of the computation in Section 4.

The error-correction problem in coding theory addresses given a received word recovering the codeword closest to it with respect to the Hamming distance. This previous statement is the usual formulation of the *Complete Decoding Problem* (CDP). The $t$-bounded distance decoding ($t$-BDD) algorithms determine a codeword (if such a word exists) which is at distance less or equal to $t$ to the received word. If $t$ is the covering radius of the code then the bounded distance decoding problem is the same as CDP. In the CDP of a linear code of length $n$, $\mathscr{C} \subset \mathbb{F}_q^n$ those errors that can be corrected are just those related to the coset leaders, which are vectors of smallest weight in the cosets $\mathbb{F}_q^n/\mathscr{C}$. When there is more than one leader in a coset there is more than one choice for the error. Therefore the following problem known as the *coset weights problem* (CWP) naturally arises:

**Input:** A binary $r \times n$ matrix $H$, a vector $\mathbf{s} \in \mathbb{F}_2^n$ and a non-negative integer $t$.

**Problem:** Does a binary vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight at most $t$ exist such that $H\mathbf{e} = s$?

Recall that $H$ can be considered as the parity check matrix of a binary code and $\mathbf{s}$ the syndrome of a received word, thus the knowledge of $\mathbf{e}$ would solve the $t$-BDD problem. Unfortunatelly the hope of finding an efficient $t$-BDD algorithm is very bleak since it was proven that the CWP is NP-complete [4]. Thus the computation of all the coset leaders is also NP-complete. The study of the set of coset leaders is also related to the study of the set of minimal codewords, which have been used in the Maximum Likelihood Decoding Analysis [2] and which are also related to the minimal access structure of secret sharing schemes [13]. Furthermore, the computation of all coset leaders of a code allows to know more about its internal structure [7]. In connection to this item we will show that the algorithm presented here can be adapted to compute a set of codewords which is a test set [2], we call this set the set of leader codewords and we also show that the leader codewords are zeroneighbors; moreover, any optimal test set is a subset of the set of leader

codewords. In addition, the set of leader codewords can be used to compute all coset leaders corresponding to a given received word.

All problems mentioned before are considered to be hard computational problems (see for example [2, 4]) even if preprocesing is allowed [10]. However, taking into account the nature of the problem, to develop an algorithm for computing the set of all coset leaders of a binary code, in the vector space of $2^n$ vectors, by generating a number of vectors close to the cardinality of this set may be quite efficient. This is our purpose extending some results on Gröbner representations for binary codes. For previous results and applications of Gröbner representations for linear codes see [6, 7, 5], for a summary of the whole material we refer the reader to [8]. We extend some settings of previous work in order to obtain the set of all coset leaders keeping record of the additive structure in the cosets. The presentation of the paper is done in a "Gröbner bases"-free context.

The outline of the paper is as follows. In Section 2 we review some of the standard facts on binary linear codes and their Gröbner representation. Section 3 gives a concise presentation of the results in this paper. Definition 3.1 corresponds to the construction of List, the main object that is used in the algorithm proposed, while Theorem 3.1 guarantees that all coset leaders will belong to List. In this section it is presented the algorithm CLBC for computing the set of all coset leaders. At the end of the section we show an application to a binary linear code with 64 cosets and 118 coset leaders. Section 4 is devoted to show how the algorithm can be adapted to compute the set of leader codewords and some related results are shown. In Section 5 we discuss some complexity issues. Finally some conclusions are given which include further research.

# 2 Preliminaries

Let $\mathbb{F}_2$ be the finite field with 2 elements and $\mathbb{F}_2^n$ be the $\mathbb{F}_2$-vector space of dimension $n$. We will call the vectors in $\mathbb{F}_2^n$ *words*. A linear code $\mathscr{C}$ of dimension $k$ and length $n$ is the image of an injective linear mapping $L : \mathbb{F}_2^k \to \mathbb{F}_2^n$, where $k \leq n$, i.e. $\mathscr{C} = L(\mathbb{F}_2^k)$. From now on, we will use the term code to mean binary linear code. The elements in $\mathscr{C}$ are called *codewords*. For a word $\mathbf{y} \in \mathbb{F}_2^n$ the *support* of $\mathbf{y}$ is $\mathrm{supp}(\mathbf{y}) = \{i \in \{1,\ldots,n\} \mid \mathbf{y}_i \neq 0\}$ and *the Hamming weight of* $\mathbf{y}$ is the cardinal of $\mathrm{supp}(\mathbf{y})$ and is denoted by $\mathrm{wt}(\mathbf{y})$. The Hamming distance between two words $\mathbf{c}_1, \mathbf{c}_2$ is $\mathrm{d}(\mathbf{c}_1, \mathbf{c}_2) = \mathrm{wt}(\mathbf{c}_1 - \mathbf{c}_2)$ and the minimum distance $d$ of a code is the minimum weight among all the non-zero codewords. Let $t = \left[\frac{d-1}{2}\right]$, where $[\cdot]$ is the greatest integer function. It is well known that CDP has a unique solution

for those vectors in $B(\mathscr{C},t) = \{\mathbf{y} \in \mathbb{F}_2^n \mid \exists \mathbf{c} \in \mathscr{C} \text{ s.t. } d(\mathbf{c},\mathbf{y}) \leq t\}$. The parameter $t$ is called the error-correcting capability of the code.

**Definition 2.1** *The words of minimum Hamming weight in the cosets of $\mathbb{F}_2^n/\mathscr{C}$ are called coset leaders.*

Cosets corresponding to elements in $B(\mathscr{C},t)$ have a unique leader, however, in general outside $B(\mathscr{C},t)$ there may be also cosets with a unique leader. Let $\mathrm{CL}(\mathscr{C})$ denote the set of coset leaders of the code $\mathscr{C}$ and $\mathrm{CL}(\mathbf{y})$ the subset of coset leaders corresponding to the coset $\mathscr{C} + \mathbf{y}$. Let $\mathbf{e}_i$ $i = 1,\ldots,n$ be the $i$-th vector of the canonical basis $\mathrm{X} = \{\mathbf{e}_1,\ldots,\mathbf{e}_n\} \subset \mathbb{F}_2^n$ and $\mathbf{0}$ the zero vector of $\mathbb{F}_2^n$. The following theorem gives some relations between cosets [12, Corollary 11.7.7].

**Theorem 2.1** *Given $\mathbf{w} \in \mathrm{CL}(\mathscr{C})$, such that $\mathbf{w} = \mathbf{w}_1 + \mathbf{e}_i$ for some $\mathbf{w}_1 \in \mathbb{F}_2^n$ and $i \in \mathrm{supp}(\mathbf{w})$. Then, $\mathbf{w}_1 \in \mathrm{CL}(\mathbf{w}_1)$.*

**Remark 2.1** *The subword $\mathbf{w}_1$ of $\mathbf{w}$ is called a descendant of $\mathbf{w}$ (see [12, p. 459, 460]).*

**Definition 2.2** *A Gröbner representation of $\mathbb{F}_2^n/\mathscr{C}$ [6, 8] is a pair $\mathrm{N},\phi$ where:*

1. *$\mathrm{N}$ is a transversal of $\mathbb{F}_2^n/\mathscr{C}$ such that $\mathbf{0} \in N$ and for each $\mathbf{n} \in \mathrm{N} \setminus \{\mathbf{0}\}$ there exists $\mathbf{e}_i$, $i \in 1,\ldots,n$, such that $\mathbf{n} = \mathbf{n}' + \mathbf{e}_i$ and $\mathbf{n}' \in \mathrm{N}$.*
2. *$\phi : \mathrm{N} \times \{\mathbf{e}_i\}_{i=1}^n \to \mathrm{N}$ is the function* Matphi *that maps each pair $(\mathbf{n},\mathbf{e}_i)$ to the element of $\mathrm{N}$ which belongs to the coset of $\mathbf{n} + \mathbf{e}_i$.*

# 3 Computing the set of coset leaders

A key point of the algorithms for computing Gröbner representations is the construction of an object we called List which is an ordered set of elements of $\mathbb{F}_2^n$ w.r.t. a linear order $\prec$ defined as follows: $\mathbf{w} \prec \mathbf{v}$ if $\mathrm{wt}(\mathbf{w}) < \mathrm{wt}(\mathbf{v})$ or $\mathrm{wt}(\mathbf{w}) = \mathrm{wt}(\mathbf{v})$ and $\mathbf{w} \prec_1 \mathbf{v}$, where $\prec_1$ is any admissible order on $\mathbb{N}^n$ (see [3, p. 167]). We will refer to such kind of orders as weight compatible orderings. In order to use $\prec_1$ on $\mathbb{F}_2^n$ it would be enough to consider $\mathbf{0}$ and $\mathbf{1}$ of $\mathbb{F}_2$ as the $\mathbf{0}$ and $\mathbf{1}$ of $\mathbb{N}$. Note that $\prec$ is a well-ordering on $\mathbb{F}_2^n$ (see [1, p. 277]) because $\mathbb{F}_2^n$ is a finite set. One can not expect admissibility of $\prec$ on $\mathbb{F}_2^n$ ($\mathbf{1}+\mathbf{1} = \mathbf{0}$ in $\mathbb{F}_2$), but we have the following property

$$\mathbf{v} \prec \mathbf{w} \text{ provided that } \mathrm{supp}(\mathbf{v}) \subset \mathrm{supp}(\mathbf{w}). \tag{1}$$

**Definition 3.1 (Construction of** List**)** . *Let* List *be the ordered structure given by the following axioms*

1. $\mathbf{0} \in$ List.
2. *If* $\mathbf{v} \in$ List *is such that* $\mathrm{wt}(\mathbf{v}) = \mathrm{wt}(\mathrm{N}(\mathbf{v}))$ *(where* $\mathrm{N}(\mathbf{v}) = \min_{\prec}\{\mathbf{w} \mid \mathbf{w} \in \mathrm{List} \cap (\mathscr{C} + \mathbf{v})\}$*), then* $\{\mathbf{v} + \mathbf{e}_i : i \notin \mathrm{supp}(\mathbf{v}), i \in \{1, \ldots, n\}\} \subset$ List.

**Remark 3.1** *As a possible set* N *of a Gröbner representation we can take the subset of* List *given as* $\{\mathrm{N}(\mathbf{v}) \mid \mathbf{v} \in \mathrm{List}\}$.

**Remark 3.2** *We start* List *with* $\mathbf{0}$ *and we will see that, whenever condition 2. holds for* $\mathbf{v}$*, the vector* $\mathbf{v}$ *will be a coset leader of* $\mathscr{C} + \mathbf{v}$*. Then, we insert* $\mathbf{v} + \mathbf{e}_i$ *to* List*, for* $i = 1, \ldots, n$ *and* $i \notin \mathrm{supp}(\mathbf{v})$ *(see Theorem 2.1). It is not necessary to introduce* $\mathbf{v} + \mathbf{e}_i$*, for* $i \in \mathrm{supp}(\mathbf{v})$*, because in this case* $\mathbf{v} + \mathbf{e}_i \prec \mathbf{v}$ *and then* $\mathbf{v} + \mathbf{e}_i$ *has been already considered in* List *since it is an ordered structure.*

Next theorem states that List in Definition 3.1 includes the set of coset leaders.

**Theorem 3.1** *Let* $\mathbf{w} \in \mathrm{CL}(\mathscr{C})$*, then* $\mathbf{w} \in$ List.

*Proof.* We will proceed by Noetherian Induction on the words of $\mathbb{F}_2^n$ with the ordering $\prec$ (see[1, p. 277]). Then $\mathbf{0} \in \mathrm{CL}(\mathscr{C})$ and by definition it belongs to List, let be $\mathbf{w} \in \mathrm{CL}(\mathscr{C}) \setminus \{\mathbf{0}\}$. Assume the property valid for any word less than $\mathbf{w}$ with respect to $\prec$, i.e. $\mathbf{u} \in$ List provided $\mathbf{u} \in \mathrm{CL}(\mathscr{C})$ and $\mathbf{u} \prec \mathbf{w}$. Taking $i \in \mathrm{supp}(\mathbf{w})$, we write $\mathbf{w} = \mathbf{u} + \mathbf{e}_i$ where $\mathbf{u} \in \mathbb{F}_2^n$, then by Theorem 2.1, $\mathbf{u} \in \mathrm{CL}(\mathscr{C})$. In addition, $|\mathrm{supp}(\mathbf{u})| = |\mathrm{supp}(\mathbf{w})| - 1$, thus $\mathbf{u} \prec \mathbf{w}$. Therefore, by applying the induction principle we have $\mathbf{u} \in$ List. If $\mathbf{u}$ is a coset leader belonging to List it is clear that $\mathrm{wt}(\mathbf{u}) = \mathrm{wt}(\mathrm{N}(\mathbf{u}))$. As a consequence, by *2.* in Definition 3.1, $\mathbf{w} = \mathbf{u} + \mathbf{e}_i \in$ List. □

**Algorithm 1 (CLBC)**

**Input:** *H: A parity check matrix of a binary code* $\mathscr{C}$*.*

**Output:** $\mathrm{CL}(\mathscr{C}), \phi$*: The set of all coset leaders and the function* Matphi*.*

*1:* List $\leftarrow [\mathbf{0}]$, N $\leftarrow \emptyset$, $r \leftarrow 0$, $\mathrm{CL}(\mathscr{C}) \leftarrow []$
*2:* ***while*** List $\neq \emptyset$ ***do***
*3:* $\quad \tau \leftarrow$ NextTerm[List], $\mathbf{s} \leftarrow \tau H$
*4:* $\quad j \leftarrow$ Member$[\mathbf{s}, \{\mathbf{s}_1, \ldots, \mathbf{s}_r\}]$

*5:* ***if*** $j \neq$ false ***then***
*6:* **for** $k \in \mathrm{supp}(\tau)$ *such that* $\tau = \tau' + \mathbf{e}_k$ *with* $\tau' \in \mathrm{N}$ **do** $\phi(\tau', \mathbf{e}_k) \leftarrow \tau_j$
*7:* ***if*** $\mathrm{wt}(\tau) = \mathrm{wt}(\tau_j)$ ***then***
*8:* $\mathrm{CL}(\mathscr{C})[\tau_j] \leftarrow \mathrm{CL}(\mathscr{C})[\tau_j] \cup \{\tau\}$
*9:* $\mathrm{List} \leftarrow \mathrm{InsertNext}[\tau, \mathrm{List}]$
*10:* ***end if***
*11:* ***else***
*12:* $r \leftarrow r+1$, $\mathbf{s}_r \leftarrow \mathbf{s}$, $\tau_r \leftarrow \tau$, $\mathrm{N} \leftarrow \mathrm{N} \cup \{\tau_r\}$
*13:* $\mathrm{CL}(\mathscr{C})[\tau_r] \leftarrow \mathrm{CL}(\mathscr{C})[\tau_r] \cup \{\tau\}$
*14:* $\mathrm{List} \leftarrow \mathrm{InsertNext}[\tau_r, \mathrm{List}]$
*15:* ***for*** $k \in \mathrm{supp}(\tau_r)$ *such that* $\tau_r = \tau' + \mathbf{e}_k$ *with* $\tau' \in \mathrm{N}$ ***do***
*16:* $\phi(\tau', \mathbf{e}_k) \leftarrow \tau_r$
*17:* $\phi(\tau_r, \mathbf{e}_k) \leftarrow \tau'$
*18:* ***end for***
*19:* ***end if***
*20:* ***end while***
*21:* ***return*** $\mathrm{CL}(\mathscr{C})$, $\phi$

Where

1. InsertNext$[\tau, \mathrm{List}]$ Inserts all the sums $\tau + \mathbf{e}_k$ in List, where $k \notin \mathrm{supp}(\tau)$, and keeps List in increasing order w.r.t. the ordering $\prec$.

2. NextTerm[List] returns the first element from List and deletes it from this set.

3. Member$[obj, G]$ returns the position $j$ of $obj$ in $G$ if $obj \in G$ and false otherwise.

**Theorem 3.2** CLBC *computes the set of coset leaders of a given binary code and its corresponding* Matphi*.*

*Proof.* Note that when an element $\tau$ is taken out from List by NextTerm the elements to be inserted in List by InsertNext are of the form $\tau + \mathbf{e}_k$, where $k \notin \mathrm{supp}(\tau)$. As a consequence, $\tau + \mathbf{e}_k \succ \tau$. Then all elements generated by CLBC in List, after $\tau$ is taken out, are greater than $\tau$. Therefore, when $\tau$ is the first element in List in Step 3, all elements of List that shall be analyzed by CLBC are greater than $\tau$.

Let us prove that the procedure generates List according to Definition 3.1. First, by Step 1, $\mathbf{0} \in$ List. Let $\tau =$ NextTerm[List] in Step 3, in this step the

syndrome $\mathbf{s}$ of $\tau$ is computed. Thus we have two cases regarding the result of Step 4, namely

1. Assume $j =$ "false", thus Steps 5 and 11 guaranty us to find $\mathrm{N}(\tau)$ as the least element in List having the same syndrome as $\tau$. When we are in this case $\mathrm{N}(\tau) = \tau$, see in Step 12 the construction of N and in Step 14 it is performed *2.* of Definition 3.1.

2. On the other hand, assume $j \neq$ "false" (an element $\mathrm{N}(\tau) = \tau_j$ has been already computed), if we have $\mathrm{wt}(\tau) = \mathrm{wt}(\tau_j)$, in Step 9 it is performed *2.* of Definition 3.1.

Therefore CLBC constructs the object List following Definition 3.1. By Theorem 3.1, the set of coset leaders is a subset of List. Then Steps 8 and 13 assure the computation of this set. The procedure computes the Matphi structure as a direct consequence of Steps 6, 16, 17. The reason for including Step 17 is that in this case

$$\tau_r + \mathbf{e}_k = \tau' \prec \tau_r$$

and those elements $\tau_r + \mathbf{e}_k$, where $k \in \mathrm{supp}(\tau_r)$, are not inserted in List since $\tau'$ has been already considered when $\tau_r$ is computed as a new element of N. In addition, since (1), all subwords of $\tau_r$ are also least elements of their cosets according to $\prec$, so $\tau' = \mathrm{N}(\tau')$ (note $\tau'$ is a coset leader and by Theorem 3.1 it belongs to List).

We have proved that CLBC guarantees the outputs we are claiming. Termination is a consequence of the fact that the cardinal of the set of elements belonging to List is less than $n|\mathrm{CL}(\mathscr{C})|$. Then, to a certain extent (when the set of coset leaders has been computed) no more elements are inserted in List in Steps 9 and 14. Therefore, the list get empty, and by Step 2, CLBC terminates.□

**Remark 3.3** *In Steps from 14 to 17, the pairs $(\tau_r, \mathbf{e}_k)$, where $k \in \mathrm{supp}(\tau_r)$, are not necessary to compute the coset leaders but for computing the structure* Matphi *and could be removed if we are only interested in the coset leaders.*

**Example 3.1** *Define the $[n = 10, k = 4, t = 1]$ code with parity check matrix $H$*

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

*We use GAP 4.12 [11] and the GAP's package GUAVA 3.10 for Coding Theory. We have built in this framework a collection of programs we call GBLA_LC "Gröbner Bases by Linear Algebra and Linear Codes" [9]. In particular, we have run the function "*CLBC*" of GBLA_LC (Coset Leaders of Binary Codes), it gives a list of three objects as an output, the first one is the set of coset leaders, the second one the function* Matphi*, and the third one the error correcting capability of the code. The complete set of coset leaders is the list below with 64 components and each component corresponds to a coset with its cosets leaders, the set* N *is composed by the first elements of each component. We have indicated with arrows some places in the list below, which we are going to use during the example. The elements in the list* $\mathrm{CL}(\mathscr{C})$ *are*

$$
\begin{array}{l}
[[1],[e_1],[e_2],[e_3],[e_4],[e_5],[e_6],[e_7],[e_8],[e_9],[e_{10}],\\
[e_1+e_2,\rightarrow e_5+e_6\leftarrow],[e_1+e_3,e_5+e_7],[e_1+e_4,e_5+e_8],\\
[e_1+e_5,e_2+e_6,e_3+e_7,e_4+e_8],[e_1+e_6,e_2+e_5],[e_1+e_7,e_3+e_5],\\
[e_1+e_8,\rightarrow e_4+e_5\leftarrow],[e_1+e_9],[e_1+e_{10}],[e_2+e_3,e_6+e_7],\\
[e_2+e_4,e_6+e_8],[e_2+e_7,e_3+e_6],[e_2+e_8,\rightarrow e_4+e_6\leftarrow],[e_2+e_9],[e_2+e_{10}],\\
[e_3+e_4,e_7+e_8],[e_3+e_8,e_4+e_7],[e_3+e_9],[e_3+e_{10}],\\
[e_4+e_9],[e_4+e_{10}],[e_5+e_9],[e_5+e_{10}],[e_6+e_9],[e_6+e_{10}],\\
[e_7+e_9],[e_7+e_{10}],[e_8+e_9],[e_8+e_{10}],[e_9+e_{10}],\\
[e_1+e_2+e_3,e_1+e_6+e_7,e_2+e_5+e_7,e_3+e_5+e_6],\\
\rightarrow[e_1+e_2+e_4,e_1+e_6+e_8,e_2+e_5+e_8,e_4+e_5+e_6]\leftarrow,\\
[e_1+e_2+e_7,e_1+e_3+e_6,e_2+e_3+e_5,e_5+e_6+e_7],\\
[e_1+e_2+e_8,e_1+e_4+e_6,e_2+e_4+e_5,e_5+e_6+e_8],\\
[e_1+e_2+e_9,e_5+e_6+e_9],[e_1+e_2+e_{10},e_5+e_6+e_{10}],\\
[e_1+e_3+e_4,e_1+e_7+e_8,e_3+e_5+e_8,e_4+e_5+e_7],\\
[e_1+e_3+e_8,e_1+e_4+e_7,e_3+e_4+e_5,e_5+e_7+e_8],\\
[e_1+e_3+e_9,e_5+e_7+e_9],[e_1+e_3+e_{10},e_5+e_7+e_{10}],[e_1+e_4+e_9,e_5+e_8+e_9],\\
[e_1+e_4+e_{10},e_5+e_8+e_{10}],[e_1+e_5+e_9,e_2+e_6+e_9,e_3+e_7+e_9,e_4+e_8+e_9],\\
[e_1+e_5+e_{10},e_2+e_6+e_{10},e_3+e_7+e_{10},e_4+e_8+e_{10}],[e_1+e_6+e_9,e_2+e_5+e_9],\\
[e_1+e_6+e_{10},e_2+e_5+e_{10}],[e_1+e_7+e_9,e_3+e_5+e_9],[e_1+e_7+e_{10},e_3+e_5+e_{10}],\\
[e_1+e_8+e_9,e_4+e_5+e_9],[e_1+e_8+e_{10},e_4+e_5+e_{10}],[e_1+e_9+e_{10}],\\
[e_2+e_3+e_8,e_2+e_4+e_7,e_3+e_4+e_6,e_6+e_7+e_8],[e_5+e_9+e_{10}]]
\end{array}
$$

*Note that* $\mathbf{y}=\mathbf{e}_4+\mathbf{e}_5+\mathbf{e}_6$ *in* $\mathrm{CL}(\mathscr{C})_{43}=[e_1+e_2+e_4,e_1+e_6+e_8,e_2+e_5+e_8,e_4+e_5+e_6]$ *(pointed by arrows) is a coset leader such that no descendant of* $\mathbf{y}$ *(see Remark 2.1) is in* N *(see the previous elements pointed by arrows). This*

*shows the importance of considering all coset leaders in 3. of Definition 3.1 and not only the coset leaders belonging to* N *like in previous works.*

*The algorithm could be adapted without incrementing the complexity to get more information like the Covering radius and the Newton radius. In this case by analyzing the last element* $\mathrm{CL}(\mathscr{C})_{64} = [e_5 + e_9 + e_{10}]$ *we have a coset of highest weight which also contains only one leader; therefore, the Covering radius and the Newton radius are equal to 3. Moreover, for computing these parameters the algorithm does not need to run until the very end.*

*As we state before, the set* N *would be enough to compute the Weight Distribution of the Coset Leaders* $\mathrm{WDCL} = (\alpha_0, \ldots, \alpha_n)$ *where* $\alpha_i$ *is the number of cosets with coset leaders of weight i,* $i = 1, \ldots, n$ *of the code. We can provide also an object which gives more information about the structure of the code, that would be the numbers of coset leaders in each coset (*$\#(\mathrm{CL})$*).*

$$\mathrm{WDCL} = [1, 10, 30, 23, 0, 0, 0, 0, 0, 0, 0].$$

$$\begin{aligned}\#(\mathrm{CL}) = [&1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,2,2,4,1,1,1,1,2,2,2,2,2,2,4,2,2,2,4,4,\\ &1,1,1,4,4,4,2,2,2,2,2,2,4,4,1,1,1,2,2,2,2,4,1,1,1,2,2,2,1,1,1,1].\end{aligned}$$

*It is also interesting to note that there are more cosets with one leader (19) than the cosets corresponding to* $B(\mathscr{C}, t)$ *(11). Therefore, there are* 30 *of the* 64 *cosets where the* CDP has a unique solution.

# 4 Computing a test set

In this section we show how the Algorithm CLBC can be adapted to compute a set of codewords which is a test set (see [2]). We will call this set of codewords the leader codewords, by using this set, we show an algorithmic way for computing all coset leaders corresponding to the coset of a given received word. We also show the connection between the leader codewords and the set of zero neighbors [2]. A test set $\mathscr{T} \subseteq \mathscr{C}$ is a set of codewords such that every word $\mathbf{y}$ either lies in $D(\mathbf{0})$ (the Voronoi region of the all-zero vector, see more details in Section 4.1) or there is a $\mathbf{v} \in \mathscr{T}$ such that $\mathrm{wt}(\mathbf{y} - \mathbf{v}) < \mathrm{wt}(\mathbf{y})$. The algorithm which uses a test set for doing complete decoding is called a gradient-like decoding algorithm.

**Definition 4.1 (leader codewords)** *The set of leader codewords is defined as*

$$\begin{aligned}\mathrm{L}(\mathscr{C}) = \quad &\{\mathbf{n}_1 + \mathbf{e}_i + \mathbf{n}_2 \mid i = 1, \ldots n,\, i \notin \mathrm{supp}(\mathbf{n}_1),\, \mathbf{n}_1 + \mathbf{e}_i \neq \mathbf{n}_2,\\ &\mathbf{n}_1, \mathbf{n}_2 \in \mathrm{CL}(\mathscr{C}) \text{ and } H \cdot (\mathbf{n}_1 + \mathbf{e}_i) = H \cdot \mathbf{n}_2\}.\end{aligned}$$

As a matter of efficiency we are only interested in the case $\mathrm{supp}(\mathbf{n}_1 + \mathbf{e}_i) \cap \mathrm{supp}(\mathbf{n}_2) = \emptyset$.

**Algorithm 2 (CLBC2)**

**Input:** *H: A parity check matrix of a binary code* $\mathscr{C}$.

**Output:** $\mathrm{CL}(\mathscr{C}), \mathrm{L}(\mathscr{C})$*: The set of coset leaders and the set of leader codewords.*

1: List $\leftarrow [\mathbf{0}]$, N $\leftarrow \emptyset$, $r \leftarrow 0$, $\mathrm{CL}(\mathscr{C}) \leftarrow []$, $\mathrm{L}(\mathscr{C}) \leftarrow []$ .
2: ***while*** List $\neq \emptyset$ ***do***
3: $\quad \tau \leftarrow \mathrm{NextTerm}[\mathrm{List}]$, $\mathbf{s} \leftarrow \tau H$
4: $\quad k \leftarrow \mathrm{Member}[\mathbf{s}, \{\mathbf{s}_1, \ldots, \mathbf{s}_r\}]$
5: $\quad$ ***if*** $k \neq$ false ***then***
6: $\quad\quad$ ***if*** $\mathrm{wt}(\tau) = \mathrm{wt}(\tau_k)$ ***then***
7: $\quad\quad\quad \mathrm{CL}(\mathscr{C})[\tau_k] \leftarrow \mathrm{CL}(\mathscr{C})[\tau_k] \cup \{\tau\}$
8: $\quad\quad\quad \mathrm{List} \leftarrow \mathrm{InsertNext}[\tau, \mathrm{List}]$
9: $\quad\quad$ ***end if***
10: $\quad\quad$ ***for*** $i \in \mathrm{supp}(\tau)$ *such that* $\tau = \tau' + \mathbf{e}_i$ *with* $\tau' \in \mathrm{CL}(\mathscr{C})$ ***do***
11: $\quad\quad\quad \mathrm{L}(\mathscr{C}) \leftarrow \mathrm{L}(\mathscr{C}) \cup_{\tau_j \in \mathscr{C}[\tau_k]} \{\tau + \tau_j\}$
12: $\quad\quad$ ***end for***
13: $\quad$ ***else***
14: $\quad\quad r \leftarrow r + 1$, $\mathbf{s}_r \leftarrow \mathbf{s}$, $\tau_r \leftarrow \tau$, $\mathrm{N} \leftarrow \mathrm{N} \cup \{\tau_r\}$
15: $\quad\quad \mathrm{CL}(\mathscr{C})[\tau_r] \leftarrow \{\tau\}$
16: $\quad\quad \mathrm{List} \leftarrow \mathrm{InsertNext}[\tau_r, \mathrm{List}]$
17: $\quad$ ***end if***
18: ***end while***
19: ***return*** $\mathrm{CL}(\mathscr{C}), \mathrm{L}(\mathscr{C})$

*Proof.* We have shown already that Algorithm CLBC computes the set of coset leaders, what we need to do is to identify the steps of the algorithm where it is necessary to insert the computation of the leader codewords. By considering Definition 4.1, it should be from Step 5 to Step 10 of Algorithm CLBC, because $\tau = \mathbf{n}_1 + \mathbf{e}_i$ and $\mathbf{n}_2$ are in the same coset. Therefore, the steps from 10 to 12 of Algorithm CLBC2 compute the leader codewords. □

**Theorem 4.1** *The subset* $\mathrm{L}^1(\mathscr{C})$ *of* $\mathrm{L}(\mathscr{C})$ *is a test set, where* $\mathrm{L}^1(\mathscr{C})$ *is defined as*

$$\begin{aligned}\mathrm{L}^1(\mathscr{C}) = \quad & \{\mathbf{n}_1 + \mathbf{e}_i + \mathbf{n}_2 \mid i = 1, \ldots n,\, i \notin \mathrm{supp}(\mathbf{n}_1),\, \mathrm{wt}(\mathbf{n}_1 + \mathbf{e}_i) > \mathrm{wt}(\mathbf{n}_2), \\ & \mathbf{n}_1 \in \mathrm{CL}(\mathscr{C}), \mathbf{n}_2 \in \mathrm{N} \text{ and } H \cdot (\mathbf{n}_1 + \mathbf{e}_i) = H \cdot \mathbf{n}_2\},\end{aligned}$$

*where* N *denotes the set of minimal elements w.r.t.* $\prec$ *of each coset.*

**Remark 4.1** *1. Note that if* $\mathrm{L}^1(\mathscr{C})$ *is a test set also* $\mathrm{L}(\mathscr{C})$ *is a test set. In order to construct the subset* $\mathrm{L}^1(\mathscr{C})$ *we consider fewer pairs* $(\mathbf{n}_1+\mathbf{e}_i,\mathbf{n}_2)$ *in contrast with the whole set of pairs considered in* $L(\mathscr{C})$*; however, we point out that the resulting set of codewords* $\mathbf{n}_1+\mathbf{e}_i+\mathbf{n}_2$ *maybe would be the same in some cases and if so* $\mathrm{L}^1(\mathscr{C})=\mathrm{L}(\mathscr{C})$.

2. *We emphasize also that the condition* $\mathrm{wt}(\mathbf{n}_1+\mathbf{e}_i)>\mathrm{wt}(\mathbf{n}_2)$ *in* $\mathrm{L}^1(\mathscr{C})$ *is just a question of better efficiency in order to compute the codewords. The difference between the two sets of leader codewords could lie only in the condition on* $\mathbf{n}_2$ *in* $\mathrm{L}^1(\mathscr{C})$*, that* $\mathbf{n}_2$ *is not only a coset leader (like in* $\mathrm{L}(\mathscr{C})$*) but the least element in its coset according to* $\prec$.

3. *According to the previous item* $\mathrm{L}(\mathscr{C})$ *can be rewritten as*

$$\begin{aligned}\mathrm{L}(\mathscr{C})= \quad & \{\mathbf{n}_1+\mathbf{e}_i+\mathbf{n}_2 \mid i=1,\ldots n,\, i\notin \mathrm{supp}(\mathbf{n}_1),\, \mathrm{wt}(\mathbf{n}_1+\mathbf{e}_i)>\mathrm{wt}(\mathbf{n}_2),\\ & \mathbf{n}_1,\mathbf{n}_2\in \mathrm{CL}(\mathscr{C}) \text{ and } H\cdot(\mathbf{n}_1+\mathbf{e}_i)=H\cdot\mathbf{n}_2\}.\end{aligned}$$

*Proof.* Let $\mathbf{y}\in\mathbb{F}_2^n$ be a word which is not a coset leader. Let $s=\mathrm{supp}(\mathbf{y})$ and define $\mathbf{n}_1$ such that $\mathbf{n}_1=\mathbf{e}_{s_1}+\ldots+\mathbf{e}_{s_l}$ is a coset leader and $\mathbf{n}_1+\mathbf{e}_{s_{l+1}}$ is not a coset leader. It is clear that $l<|s|$, otherwise $\mathbf{y}$ would be a coset leader. Let $\mathbf{n}_2=\mathrm{N}(\mathbf{n}_1+\mathbf{e}_{s_{l+1}})$ and $\mathbf{c}=\mathbf{n}_1+\mathbf{e}_{s_{l+1}}+\mathbf{n}_2$, note that $\mathrm{wt}(\mathbf{n}_1+\mathbf{e}_{s_{l+1}})>\mathrm{wt}(\mathbf{n}_2)$; therefore, $\mathbf{c}\in\mathrm{L}^1(\mathscr{C})$ and $\mathrm{wt}(\mathbf{y}-\mathbf{c})<\mathrm{wt}(\mathbf{y})$, which completes the proof. □

The following theorem gives a bound for the weight of a leader codeword.

**Theorem 4.2** *For every codeword* $\mathbf{c}\in\mathrm{L}(\mathscr{C})$, $\mathrm{wt}(\mathbf{c})\leq 2\rho+1$, *where* $\rho$ *is the covering radius of* $\mathscr{C}$.

*Proof.* Let $\mathbf{c}\in\mathrm{L}(\mathscr{C})$, then there exist $\mathbf{n}_1,\mathbf{n}_2\in\mathrm{CL}(\mathscr{C})$ and $i\in\{1,\ldots,n\}$, $i\notin\mathrm{supp}(\mathbf{n}_1)$ such that $\mathrm{wt}(\mathbf{n}_1+\mathbf{e}_i)>\mathrm{wt}(\mathbf{n}_2)$ and $\mathbf{c}=\mathbf{n}_1+\mathbf{e}_i+\mathbf{n}_2$. Taking into account that $\mathrm{wt}(\mathbf{n}_1)\leq\rho$ and $\mathrm{wt}(\mathbf{n}_2)\leq\rho$, then, $\mathrm{wt}(\mathbf{c})\leq 2\rho+1$. □

**Theorem 4.3** *Let* $\mathbf{y}\in\mathbb{F}_2^n$ *be a received word, the following algorithm computes the set of coset leaders associated to* $\mathbf{y}$.

***Algorithm 3***

**Input:** $\mathbf{y}\in\mathbb{F}_2^n$, $\mathrm{L}(\mathscr{C})$.

**Output:** $\mathrm{CL}(\mathbf{y})$.

1. *Compute* $\mathrm{N}(\mathbf{y})$ *by gradient-like decoding using the test set* $\mathrm{L}^1(\mathscr{C})$.

2. Do $y = \mathrm{N}(\mathbf{y})$, $\mathbf{S} = \{\mathbf{y}\}$ *and* $\mathbf{L} = \mathrm{L}(\mathscr{C})$.

3. If *there exists* $\mathbf{c} \in \mathbf{L}$ *s.t.* $\mathrm{wt}(\mathbf{y}-\mathbf{c}) = \mathrm{wt}(\mathbf{y})$ then GOTO Step 4 else GOTO Step 5.

4. $L := L \setminus \{\mathbf{c}\}$*;* $\mathbf{S} := \mathbf{S} \cup \{\mathbf{y}-\mathbf{c}\}$*;* GOTO Step 3.

5. RETURN(**S**).

*Proof.* Step 1 of the algorithm is guaranteed by Theorem 4.1, let $y = \mathrm{N}(\mathbf{y})$. Let $i \in \mathrm{supp}(\mathbf{y})$ and $\mathbf{n}_1$ be such that $\mathbf{y} = \mathbf{n}_1 + \mathbf{e}_i$. By Theorem 2.1, $\mathbf{n}_1$ is a coset leader. The rest of the proof is straightforward from the definition of leader codewords. □

## 4.1 Leader codewords and Zero neighbors

In this section we have compiled some basic facts related to Zero-Neighbors (see [2]).

The Voronoi region $D(\mathbf{c})$ of $\mathbf{c} \in \mathscr{C}$ is defined as

$$D(c) = \{\mathbf{y} \in \mathbb{F}_2^n : \forall\, \mathbf{c}' \in \mathscr{C}\, (\mathrm{d}(\mathbf{y},\mathbf{c}) \leq \mathrm{d}(\mathbf{y},\mathbf{c}'))\}.$$

Note that $D(\mathbf{0})$, the voronoi region of the all-zero codeword, is the set of coset leaders.

Let $\mathrm{A} \subset \mathbb{F}_2^n$ and let $\chi(\mathrm{A})$ be formed by all words in $\mathbb{F}_2^n$ at distance 1 from A. The boundary of A is defined as

$$\partial(\mathrm{A}) = \chi(\mathrm{A}) \cup \chi(\mathbb{F}_2^n \setminus \mathrm{A}).$$

A nonzero codeword $\mathbf{z} \in \mathscr{C}$ is called a zeroneighbor if its Voronoi region shares a common boundary with $D(\mathbf{0})$, i.e., if $\partial(D(\mathbf{z})) \cap \partial(D(\mathbf{0})) \neq \emptyset$. Denoting the set of zeroneighbors by $\mathscr{Z}$, this set has the the following property

$$(\chi(D(\mathbf{0})) \cap D(\mathbf{z}) \neq \emptyset) \Rightarrow \mathbf{z} \in \mathscr{Z}.$$

The previous property is the only property of the set $\mathscr{Z}$ that is essential for successful decoding (see [2] ) and this set satisfies

$$\chi(D(\mathbf{0})) \subset \bigcup_{\mathbf{z} \in \mathscr{Z}} D(\mathbf{z}). \tag{2}$$

It is possible to restrict the test set of vectors by choosing a smallest subset of $\mathscr{Z}$ with this property, such a set is denoted by $\mathscr{Z}_{min}$, although such a set need not to be unique, its size is well defined.

**Theorem 4.4** $(\chi(D(\mathbf{0}))\cap D(\mathbf{z})\neq\emptyset)\Leftrightarrow\mathbf{z}\in \mathrm{L}(\mathscr{C})$.

As a simple consequence of the previous theorem we have that $\mathrm{L}(\mathscr{C})\subset\mathscr{Z}$.
*Proof.*
$\chi(D(\mathbf{0}))\cap D(\mathbf{z})\neq\emptyset\Leftrightarrow\exists\,\mathbf{n}_1\in D(\mathbf{0}), i\in\{1,\ldots,n\}\setminus\mathrm{supp}(\mathbf{n}_1)$
$(\,\mathbf{n}_1+\mathbf{e}_i\in\chi(D(\mathbf{0}))\wedge\mathbf{n}_1+\mathbf{e}_i\in D(\mathbf{z})\,)\Leftrightarrow$
$\exists\,\mathbf{n}_1\in D(\mathbf{0}), i\in\{1,\ldots,n\}\setminus\mathrm{supp}(\mathbf{n}_1)$
$(\,\mathbf{n}_1+\mathbf{e}_i\notin D(\mathbf{0})\wedge\forall\mathbf{c}\in\mathscr{C}\,(\mathrm{wt}(\mathbf{z}-(\mathbf{n}_1+\mathbf{e}_i))\leq\mathrm{wt}(\mathbf{c}-(\mathbf{n}_1+\mathbf{e}_i)))\,)\Leftrightarrow$
$\exists\,\mathbf{n}_1\in D(\mathbf{0}), i\in\{1,\ldots,n\}\setminus\mathrm{supp}(\mathbf{n}_1),\mathbf{n}_2\in\mathrm{CL}(\mathscr{C})$
$(\,\mathbf{n}_2=\mathbf{z}-(\mathbf{n}_1+\mathbf{e}_i)\wedge\mathrm{wt}(\mathbf{n}_2)<\mathrm{wt}(\mathbf{n}_1+\mathbf{e}_i)\,)\Leftrightarrow$
$\exists\,\mathbf{n}_1,\mathbf{n}_2\in\mathrm{CL}(\mathscr{C}), i\in\{1,\ldots,n\}\setminus\mathrm{supp}(\mathbf{n}_1)$
$(\,\mathbf{n}_2=\mathbf{z}-(\mathbf{n}_1+\mathbf{e}_i)\wedge H\cdot(\mathbf{n}_1+\mathbf{e}_i)=H\cdot\mathbf{n}_2\wedge\mathrm{wt}(\mathbf{n}_2)<\mathrm{wt}(\mathbf{n}_1+\mathbf{e}_i)\,)\Leftrightarrow\mathbf{z}\in\mathrm{L}(\mathscr{C})$.
□

**Remark 4.2** *By Theorem 4.4, Algorithm* CLBC2 *computes a set of zero-neighbors, which is the set of leader codewords. The algorithm could also compute a subset* $\mathrm{L}^1(\mathscr{C})$ *that could be smaller and is also a test set. Theorem 4.3 shows that the subset* $\mathrm{L}(\mathscr{C})$ *of* $\mathscr{Z}$ *allows to compute by Algorithm 3 all coset leaders corresponding to a given received word. Theorem 4.2 shows that not only the weight of the codewords in a minimal test set are bounded by* $2\rho+1$ *but for any leader coderword also.*

From Theorem 4.4 and the connection of this property with (2) we can get the following statement.

**Theorem 4.5** *Let* $\mathscr{C}$ *be a binary code, then any set* $\mathscr{Z}_{min}$ *is a subset of* $\mathrm{L}(\mathscr{C})$.

*Proof.* Let $\mathscr{Z}_{min}$ be a minimal test set. Then for any $\mathbf{z}\in\mathscr{Z}_{min}$ we have $\chi(D(\mathbf{0}))\cap D(\mathbf{z})\neq\emptyset$, since $z$ could otherwise be removed from $\mathscr{Z}_{min}$ and this would imply that it is not a minimal test set. Then, let be $\mathbf{n}_1\in D(\mathbf{0})$ and $i\notin\mathrm{supp}(\mathbf{n}_1)$ such that $\mathbf{n}_1+\mathbf{e}_i\in\chi(D(\mathbf{0}))$ and $\mathbf{n}_1+\mathbf{e}_i\in D(\mathbf{z})$. Let be $\mathbf{n}_2=\mathbf{z}-\mathbf{n}_1-\mathbf{e}_i$, thus, $\mathbf{n}_2\in\mathrm{CL}(\mathscr{C})$; consequently, $\mathbf{z}\in\mathrm{L}(\mathscr{C})$. □

**Example 4.1** *We use the same code represented in Example 3.1. There is a built in function "*CLBC2*" in GBLA_LC which performs Algorithm 2 to compute* $\mathrm{L}(\mathscr{C})$ *and a variant of the same function which computes* $\mathrm{L}^1(\mathscr{C})$ *and* $\mathrm{L}(\mathscr{C})$ *at the same*

*time. For this code we got that* $\mathrm{L}^1(\mathscr{C}) = \mathrm{L}(\mathscr{C})$ *and the set of leader codewords is*

$$\begin{aligned}\mathrm{L}(\mathscr{C}) = \quad & \{e_3+e_4+e_7+e_8, e_2+e_4+e_6+e_8, e_2+e_3+e_6+e_7,\\ & e_1+e_4+e_5+e_8, e_1+e_3+e_5+e_7, e_1+e_2+e_5+e_6,\\ & e_4+e_6+e_7+e_9+e_{10}, e_3+e_6+e_8+e_9+e_{10}, e_2+e_7+e_8+e_9+e_{10},\\ & e_2+e_3+e_4+e_9+e_{10}, e_1+e_5+e_6+e_7+e_8+e_9+e_{10},\\ & e_1+e_3+e_4+e_5+e_6+e_9+e_{10}, e_1+e_2+e_4+e_5+e_7+e_9+e_{10},\\ & e_1+e_2+e_3+e_5+e_8+e_9+e_{10}\}.\end{aligned}$$

*The only codeword different from the all-zero vector that is missing in* $\mathrm{L}(\mathscr{C})$ *is the codeword* $(1,1,1,1,1,1,1,1,0,0)$ *of weight* 8*, which is expected because in Example 3.1 we show that the covering radius of* $\mathscr{C}$ *is* 3 *and we have shown in Theorem 4.2 that the weight of a leader codeword is less than or equal to* $2\cdot 3+1=7$*.*

*Now in the following table we show the result of the computations with other two codes.*

| *code(*$n,k,d$*)* | $2^k$ | $2^{n-k}$ | $\mid \mathrm{L}^1(\mathscr{C}) \mid$ | $\mid \mathrm{L}(\mathscr{C}) \mid$ |
|---|---|---|---|---|
| *Golay Code(*23, 12, 7*)* | *4096* | *2048* | *253* | *253* |
| *BCH(*21, 12, 5*)* | *4096* | *512* | *478* | *549* |

*From this table one can see the nice property of the Golay code of being a perfect code, the code is bigger than the BCH code, because, although they have the same number of codewords, in the Golay code there are 4 times the number of cosets. However, the number of elements in* $\mathrm{L}(\mathscr{C})$ *is less than in the BCH code. Also because the Golay code is perfect, it is not a surprise that* $\mathrm{L}^1(\mathscr{C}) = \mathrm{L}(\mathscr{C})$ *in this case. Note also that the results for the BCH code say that* $\mathrm{L}^1(\mathscr{C})$ *could be smaller than* $\mathrm{L}(\mathscr{C})$*.*

*The following calculations were done with the computer "Roadrunner" (102 Gb RAM, 3.05 Ghz, using one of the 12 processors), at the beginning the computations last for more than 5 days and after some improvements in the implementation we have arrived to the following*

| *code(*$n,k,d$*)* | $2^k$ | $2^{n-k}$ | $\mid \mathrm{L}^1(\mathscr{C}) \mid$ | $\mid \mathrm{L}(\mathscr{C}) \mid$ | *time (h:min.:sec.)* |
|---|---|---|---|---|---|
| *QR(*31, 16, 7*)* | *65536* | *32768* | *5324* | *5828* | *15:47:21* |
| *BCH(*31, 16, 7*)* | *65536* | *32768* | *5606* | *5828* | *13:18:33* |

*Note that both codes have the same number of leader codewords, but the second one has more codewords in the subset* $\mathrm{L}^1(\mathscr{C})$*. Also the second code has 119568 coset leaders and the first one has 98550.*

# 5 Complexity Analysis

For a detailed complexity analysis and some useful considerations from the computational point of view we refer the reader to [7]. In the case of this paper, the difference is that we work out the set of all coset leaders and not only a set of canonical forms (N). Next theorem shows an upper bound for the number of iterations that Algorithm CLBC will perform.

**Theorem 5.1** CLBC *computes the set of coset leaders of a given binary code* $\mathscr{C}$ *of length n after at most* $\mathbf{n}|\mathrm{CL}(\mathscr{C})|$ *iterations.*

*Proof.* Note that the number of iterations is exactly the size of List. In the proof of Theorem 3.2 was shown that the algorithm follows this definition to construct the object List. It is clear that the size of List is bounded by $n|\mathrm{CL}(\mathscr{C})|$, note that we can write List, as a set as follows

$$\mathrm{List} = \{\mathbf{w} + \mathbf{e}_i \mid \mathbf{w} \in \mathrm{CL}(\mathscr{C}) \text{ and } i \in \{1, \ldots, n\}\}.$$

□

**Remark 5.1**

1. *By the proof above we require a memory space of* $O(n|\mathrm{CL}(\mathscr{C})|)$.
   *We assume that for computing the set of coset leaders it is required at least* $O(|\mathrm{CL}(\mathscr{C})|)$*; therefore,* CLBC *is near the optimal case of memory requirements.*

2. CLBC *generates at most* $n|\mathrm{CL}(\mathscr{C})|$ *words from* $\mathbb{F}_2^n$ *to compute the set of coset leaders. An algorithm for computing this set needs to generate from* $\mathbb{F}_2^n$ *at least a subset formed by all coset leaders, i.e.* $|\mathrm{CL}(\mathscr{C})|$*; therefore, the algorithm is near the optimal case of computational complexity.*

3. CLBC2 *by its nature has the same complexity as* CLBC. *The advantage of computing the set of leader codewords is that it supplies an algorithmic way of solving similar problems that Matphi solves but with a structure which is considerately smaller.*

# 6 Conclusion

The Algorithm CLBC is formulated in this paper, which turns out to be quite efficient for computing all coset leaders of a binary code from memory requirements and computational complexity view. Although, as it is expected, the complexity of the algorithm is exponential (in the number of check positions).

The difference of CLBC with its predecessors rely also in the computation of all coset leaders instead of a set of representative leaders for the cosets, however, it supports the computation of the function Matphi. We remark that the computation of Matphi is not necessary at all for the main goal of computing the coset leaders. We have kept this resource in the algorithm because this structure provides some computational advantages (see [6, 7, 8]) and, although the algorithm will be clearly faster without computing Matphi, the nature of the computational complexity and space complexity will remain the same.

Algorithm CLBC can be transformed into Algorithm CLBC2 to compute the set of leader codewords, which turn out to be a set of zeroneighbors that is a test set. The set of leader codewords allows to compute all coset leaders corresponding to the coset of a received word. We have shown also that the set of leader codewords contains any minimal test set.

Unfortunately, a generalization to non-binary linear codes is not trivial from this work. The main reason seems to be that the solution based on Hamming weight compatible orderings will not continue being possible; the *error vector ordering* it is used in the general approach [7] is not a total degree compatible ordering, although it allowed to set up the computational environment in order to compute Gröbner representations for linear codes.

## Acknowledgment:

Authors are very acknowledged to anonymous referee for interesting and detailed comments in an earlier version of this report.

# References

[1] W. W. Adams, Ph. Loustaunau. An introduction to Gröbner bases. American Mathematical Society, Providence, RI (1994).

[2] A. Barg. Complexity issues in coding theory. In Handbook of Coding Theory, Elsevier Science, Vol. 1, 1998.

[3] T. Becker, V. Weispfenning. Gröbner Bases. A Computational Approach to Commutative Algebra. Springer-Verlag, New York (1993).

[4] E.R. Berlekamp, R.J. McEliece, H.C.A. van Tilborg. On the Inherent Intractability of Certain Coding Problems. IEEE Transactions on Information Theory, Vol. IT-24, N. 3, 384–386 (1978).

[5] M. Borges-Quintana, M.A. Borges-Trenard, P. Fitzpatrick, E. Martínez-Moro. On a Gröbner bases and combinatorics for binary codes. Appl. Algebra Engrg. Comm. Comput, Vol. 19, N. 5, 393–411 (2008).

[6] M. Borges-Quintana, M.A. Borges-Trenard, E. Martínez-Moro. A general framework for applying FGLM techniques to linear codes. AAECC 16, Lecture Notes in Comput. Sci., Springer, Berlin, Vol. 3857, 76–86, 2006.

[7] M. Borges-Quintana, M.A. Borges-Trenard, E. Martínez-Moro. On a Gröbner bases structure associated to linear codes. J. Discrete Math. Sci. Cryptogr., Vol. 10, N. 2, 151–191 (2007).

[8] M. Borges-Quintana, M.A. Borges-Trenard, E. Martínez-Moro. A Gröbner representation of linear codes. In: T. Shaska, W.C. Huffman, D. Joyner, V. Ustimenko (eds.) Advances in Coding Theory and Cryptography, 17–32, World Scientific (2007).

[9] M. Borges-Quintana, M.A. Borges-Trenard, E. Martínez-Moro. GBLA-LC: Gröbner Bases by Linear Algebra and Linear Codes. In: ICM 2006. Mathematical Software, EMS, 604–605 (2006).

[10] J. Bruck, M. Naor. The Hardness of Decoding Linear Codes with Preprocessing. IEEE Trans. on Inf. Th. , Vol 36, NO. 2, (1990)

[11] The GAP Group, GAP – Groups, Algorithms, and Programming. Version 4.12 (2009). `http://www.gap-system.org`.

[12] W.C. Huffman, V. Pless. Fundamentals of error-correcting codes. Cambridge University Press, Cambridge (2003).

[13] J. Massey. Minimal codewords and secret sharing. In: Proc. 6th Joint Swedish–Russian Workshop on Information Theory, Mölle, Sweden, 246–249 (1993).

# Technical Reports of the Doctoral Program "Computational Mathematics"

**2012**

**2012-01** M.T. Khan: *Formal Semantics of* MiniMaple January 2012. Eds.: W. Schreiner, F. Winkler

**2012-02** M. Kollmann, W. Zulehner: *A Robust Preconditioner for Distributed Optimal Control for Stokes Flow with Control Constraints* January 2012. Eds.: U. Langer, R. Ramlau

**2012-03** W. Krendl, V. Simoncini, W. Zulehner: *Stability Estimates and Structural Spectral Properties of Saddle Point Problems* February 2012. Eds.: U. Langer, V. Pillwein

**2012-04** V. Pillwein, S. Takacs: *A local Fourier convergence analysis of a multigrid method using symbolic computation* April 2012. Eds.: M. Kauers, W. Zulehner

**2012-05** I. Georgieva, C. Hofreither: *Tomographic Reconstruction of Harmonic Functions* April 2012. Eds.: U. Langer, V. Pillwein

**2012-06** M.T. Khan: *Formal Semantics of a Specification Language for* MiniMaple April 2012. Eds.: W. Schreiner, F. Winkler

**2012-07** M. Borges-Quintana, M.A. Borges-Trenard, I. Márquez-Corbella and E. Martínez-Moro: *Computing coset leaders and leader codewords of binary codes* May 2012. Eds.: F. Winkler, V. Pillwein

The complete list since 2009 can be found at
`https://www.dk-compmath.jku.at/publications/`

# Doctoral Program
# "Computational Mathematics"

**Director:**

Prof. Dr. Peter Paule
Research Institute for Symbolic Computation

**Deputy Director:**

Prof. Dr. Bert Jüttler
Institute of Applied Geometry

**Address:**

Johannes Kepler University Linz
Doctoral Program "Computational Mathematics"
Altenbergerstr. 69
A-4040 Linz
Austria
Tel.: ++43 732-2468-6840

**E-Mail:**

office@dk-compmath.jku.at

**Homepage:**

http://www.dk-compmath.jku.at

Submissions to the DK-Report Series are sent to two members of the Editorial Board who communicate their decision to the Managing Editor.