An FGLM Method for Decoding Linear Codes*

Mijail Borges Quintana † Franz Winkler ‡
Miguel A. Borges Trenard †

† Departamento de Matemática, Facultad de Ciencias.
Universidad de Oriente, Santiago de Cuba 90500, Cuba.
{mijail, mborges}@csd.uo.edu.cu

‡ RISC-Linz. Johannes Kepler University. Austria.
winkler@risc.uni-linz.ac.at

May 8, 2000

Abstract

A new algorithmic method is presented for the decoding process of a linear code. This method has the flavor of the FGLM techniques that belong to the framework of the Gröbner Bases Theory. This method consists of two algorithms, one for the preparation phase of the code, and the other one for the decoding process. Starting from a parity check matrix of the linear code, our approach enables the user to decode a given codeword in a number of steps proportional to its length. Complexity analysis, complete examples and comments about the application of FGLM techniques to linear codes are also given in this paper.

Key words error-correcting codes – linear codes – FGLM – Gröbner bases

---

*This work was done while the first author spent the academic year 1999/2000 at RISC-Linz, supported by ÖAD. We also acknowledge partial support by the Austrian Fonds zur Förderung der wissenschaftlichen Forschung, under project Nr. SFB 013/F1304. The first author's email address at RISC-Linz is mborges@risc.uni-linz.ac.at

1 Introduction

In this paper we present a new method for decoding linear codes. Gröbner bases tools play an essential role in our method. But our approach is different from other known applications of Gröbner bases in coding theory [1], such as to find a Gröbner basis of the error locator ideal and a systematic method for encoding and decoding m-dimensional cyclic codes. FGLM techniques have been already related with coding theory in some literature (see [Sak], [CLO]), but these connections are rather different from the approach presented in this work.

Our method has strong connections with the general method for decoding a linear code known as syndrome decoding. In the syndrome decoding method one looks for the coset leader of the syndrome corresponding to the received word. In our approach, we look directly for the coset leader corresponding to the received word. Before the stage of decoding, as in the syndrome decoding method, we need to perform preliminary calculations, which essentially amount to an execcution of a variant of the procedure Matphi of the FGLM basis transformation algorithm [FGLM]. The process of decoding amounts to reducing a word to its canonical form by means of a border basis. Preliminary calculations are guaranteed to be done with a very good complexity and with no more memory required than for the syndrome decoding method. The low complexity for the decoding algorithm could be a good reason for using this approach, wherever the requirements of our method can be satisfied.

Overview: In Section 2, a short introduction to linear codes is presented. In Section 3, we connect a monoid and a term ordering on the monoid with a given linear code. A variant of the FGLM algorithm is introduced in Section 4 for the setting constructed in Section 3. The crucial results, Lemma 1 and Theorems from 3 to 5, are given in Section 5. As a consequence of these results, an algorithm for decoding linear codes is derived at the end of this section. Section 6 presents an example of how this method works. Finally, Section 7 is devoted to complexity analysis, where not only the complexity functions of the algorithms are provided, but also, a comparison with other algorithms and some advantage and limitations of this approach are discussed.

[1] See [Sak], [CLO], [CCS].

2 Linear codes

In this section we review the main notions of the theory of linear codes. The reader not familiar with this topic could consult, for example, [vL], [CLO]. We will also give some results relating to linear codes that will be frequently used in this work.

Let F_q be the Galois Field of q elements ($q = p^m$, where p is a prime number). The elements of F_q are represented by $a_0 + \ldots + a_{m-1}\alpha^{m-1}$, where α is a root of an irreducible polynomial of degree m over F_p and, for $j \in [0, m-1]$, $\alpha_j \in F_p$.

Let E be a linear map from F_q^k to F_q^n for $k < n$. Let $C = E(F_q^k) \subset F_q^n$ be the linear code defined by E. The elements in C are called codewords.

There exists a matrix $H_{n\times(n-k)}$ with coefficients in F_q, such that a vector c in F_q^n is a codeword if and only if it satisfies the system

$$cH = 0. \tag{1}$$

So the set C can be characterized as the set of all elements in F_q^n satisfying System (1). H is called *a parity check matrix for the linear code.*

We will use in this work the well known Hamming distance. The minimum distance of a code C is the minimal Hamming distance $d(c, c')$ of two codewords c, c'. The weight of a codeword is its distance from zero. If d is the minimum distance for the code C, it is well know that one can correct up to t errors, where $d = 2t + 1$ or $d = 2t + 2$. t is called the error-correcting capability of the code.

Given the received word $y \in F_q^n$ such that it contains at most t errors, there exists only one solution e of the system

$$eH = yH, \tag{2}$$

where e has weight less than or equal to t. This solution e of the System (2) is such that $y - e = c$ is the corresponding codeword for the received word y, that is, e is the error vector.

The value yH is known as the *syndrome* of the vector $y \in F_q^n$, that is, C is the subspace of the vectors whose syndrome is zero. The set of $y \in F_q^n$ with at most t errors can be represented as:

$$B(C, t) := \{\, y \in F_q^n \mid \underset{c\in C}{\exists}\, d(c, y) \le t \,\}.$$

For readers familiar with reduction processes and normal form concepts it might be possible to guess a relationship between those notions and the following facts (which were previously considered in this work): For each vector $y \in F_q^n$, with at most t errors, the vector e exists; moreover, it is uniquely determined by y and has the minimal possible weight among the vectors that are congruent to y modulo C. Therefore, it seems natural to consider e as the "normal form of y" (note also that y will have normal form zero if and only if y is a codeword). Actually these ideas have been already used in some method for decoding such as the general method known as syndrome decoding see [vL]. The main ideas behind this general method are applied for our algorithm, by using the capabilities of the structure of a linear code and connecting them with an efficient application of FGLM techniques. We will also show how to deal with the vectors that are out of the set $B(C, t)$.

3 Associating a monoid with a linear code

In connection with the vector space F_q^n a set X of variables will be introduced so that later we can relate the free commutative monoid in these variables with the structure of the linear code.

Given $(y_1, \ldots, y_n) \in F_q^n$, each component y_i can be represented as $y_i = \sum_{j=1}^{m} \beta_{ij}\alpha^{j-1}$. Based on this representation, the alphabet $X = \{x_{11}, \ldots, x_{1m}, \ldots, x_{n1}, \ldots, x_{nm} \mid i \in [1, n] \text{ and } j \in [1, m]\}$ is introduced and the free commutative monoid $[\,X\,]$ is associated with F_q^n in the following way:

Let ψ be the mapping from X to F_q^n that sends every x_{ij} to the corresponding vector with zero in all its components up to the position i which is equal to α^{j-1}.

$$\begin{array}{rccl} \psi: & X & \longrightarrow & F_q^n \\ & x_{ij} & \longrightarrow & (\underbrace{0, \ldots, 0}_{i-1 \text{ zeros}}, \alpha^{j-1}, 0, \ldots, 0). \end{array} \tag{3}$$

As $[\,X\,]$ is freely generated by X and ψ is a mapping of X onto a generating set of the additive monoid F_q^n, ψ can be extended in a natural way to a linear morphism of $[\,X\,]$ onto F_q^n; moreover, ψ maps $\prod_{i=1}^{n}\prod_{j=1}^{m} x_{ij}^{\beta_{ij}}$ into the vector $(\sum_{j=1}^{m} \beta_{ij}\alpha^{j-1})_i$ modulo p.

Remark 1. *Taking into consideration the comments made above, the variables x_{ij} can also be seen as the set of nm variables x_k, where $k = (i-1)m+j$. This last representation of variables will be used in the subsequent sections of this paper.*

It is well known that a linear code defines an equivalence relation $R(C)$ in F_q^n given as:

$$\underset{x,y\in F_q^n}{\forall} ((x,y) \in R(C) \Longleftrightarrow x - y \in C).$$

By the linear morphism ψ, the congruence modulo C can be transferred onto $[\,X\,]$ in the following way:

$$u \cong_C w \Longleftrightarrow (\psi(u), \psi(w)) \in R(C) \Longleftrightarrow \psi(u)H = \psi(w)H.$$

After setting $\xi(u) := \psi(u)H$ we can write this relation as

$$u \cong_C w \Longleftrightarrow \xi(u) = \xi(w).$$

Normally, in the literature, a free commutative monoid over an alphabet with n elements is associated, in a canonical form, to F_q^n. Using the representation given above makes it possible to associate the normal multiplication in the monoid with the $+$ operation in F_q^n. In fact, this is the very beginning for using as much as possible the linear structure of the linear code in relation with the monoid.

Before introducing the term ordering that will be used in this work, the following notations are introduced:

1	the unit for the product in $[\,X\,]$.
i, j, s, l	integers in the rank $[1, max(m,n)]$.
u, v, w	elements of $[\,\mathrm{X}\,]$ (terms).
Supp(w)	the set of variables that divide w, i.e. the support of w.
$Ind(w)$	$\{\, i \in [1,n] \mid \underset{j}{\exists}\, j \in [1,m] \wedge (x_{ij} \in Supp(w)) \,\}$, i.e. the indeces associated to w.
$Card(U)$	the cardinal of the set U.

Definition 1. (The Error Vector term ordering $<_e$) $u <_e w$ *(u is less than w w.r.t. the* error vector term ordering*), iff one of the following conditions holds:*

(i) *$Card(Ind(u)) < Card(Ind(w))$.*

(ii) *$Card(Ind(u)) = Card(Ind(w))$ and $u \prec w$, where $\prec$ denotes any arbitrary but fixed admissible [2] term ordering on $[X]$.*

It is easy to prove that $<_e$ is a total ordering in $[\,X\,]$. Note that condition (i) is the main characteristic of this term ordering and this fact will be strongly useful for us, in particular behind this condition is the reason for the name of $<_e$.

Proposition 1. (Properties of $<_e$)

(i) *$1 <_e u$, for all $u \neq 1$.*

(ii) *$u <_e ux$, for all $x \in X$.*

Proof: (i) Follows immediately from condition (i) in the definition of $<_e$, 1 is the only element in [X] having $Card(Ind(u)) = 0$.

It is obvious that $Card(Ind(u)) \leq Card(Ind(ux))$. If the inequality holds strictly then, due to condition (i) in Definition 1, (ii) follows. Assume we are in the case of the equality, then the decision for $<_e$ depends on $\prec$, an admissible term ordering in [X], and the admissibility condition is stronger than condition (ii).$\square$

Unfortunately, the term ordering $<_e$ is not admissible. But Proposition 1 is enough for getting our decoding algorithm. Everything will look as a normal use of border bases like in [MMM], [BBM]. However, it is not actually a reduction using border bases in the usual meaning. The problems appear in the intermediate results. These complications are due to the non-admissibility and the fact that in general linear codes are not perfect codes [3]. So the unique solution with minimal weight of the System (2) in Section 2 is only guaranteed for vectors y with distance not greater than t to a codeword.

Proposition 2. (Relation between the functions *Ind* and *weight*)

$$Card(Ind(w)) \geq weight(\psi(w)).$$

[2] In Gröbner Bases Theory, $\prec$ is admissible if 1 is the lowest element w.r.t. $\prec$ and the multiplication on $[\,X\,]$ is compatible with $\prec$.

[3] The Hamming codes and the Golay codes are the only linear codes that are perfect, see [vL].

Proof: Let $y := \psi(w) = (\sum_{j=1}^{m} y_{ij}\alpha^{j-1})_i$ (where $w := \prod_{i=1}^{n}\prod_{j=1}^{m} x_{ij}^{\beta_{ij}}$). Thus, by the definition of ψ we have:

$$y_{ij} = 0 \Longleftrightarrow \beta_{ij} \cong 0\ (mod\ p).$$

The inequality of the proposition comes from the above equivalence. □

Definition 2. (Standard representation) *The word w is said to be in* standard representation *iff all the exponents of the variables in the representation of w are less than p. Given $y \in F_q^n$, we say that w is the* standard representation *of y iff $\psi(w) = y$ and w is in standard representation.*

Remark 2. *The equality in the proposition above holds by using for a vector y its standard representation.*

The next proposition shows the reason for the dominating condition (i) in the definition of $<_e$. By this condition $<_e$ becomes compatible with the weight function for standard representations in $[\,X\,]$ of the vectors y in F_q^n.

Proposition 3. (Semicompatibility of $<_e$ with the weight function) *Let u and w be standard representations of $\psi(u)$ and $\psi(w)$ respectively. Then:*

$$weight(\psi(u)) < weight(\psi(w)) \Longrightarrow u <_e w.$$

The proof of this proposition comes from Proposition 2 and Remark 2.

Definition 3. (Normal form) *The word w is said to be the* normal form *of the vector $y \in F_q^n$ iff the following two conditions hold:*

(i) $\xi(w) = yH$.

(ii) $\underset{u\in[\,X\,]\setminus\{w\}}{\forall} ((\xi(u) = yH) \Longrightarrow w <_e u)$.

Conversely, we say that w is a normal form iff it is the normal form of $\psi(w)$.

Remark 3. *For a given syndrome, the normal form corresponding to this syndrome is the least word w according to $<_e$ such that $\xi(w)$ corresponds to this syndrome.*

Theorem 1. (Normal forms of the vectors in $B(C,t)$) *For every y in $B(C,t)$, w_e is the normal form corresponding to y if and only if $\psi(w_e)$ is the error vector corresponding to y and w_e is in standard representation.*

Proof: As the uniqueness of the so called normal form is guaranteed by its own definition, we only need to prove that the standard representation of the error vector of y satisfies the condition to be the normal form of y.

Let $y \in B(C,t)$, e_y be the error vector corresponding to y; then, $e_yH = yH$ and $weight(e_y) \leq t$ (see System (2) of Section 2). Let, on the other hand, w be the standard representation of e_y. By Proposition 2 and Remark 2 $weight(e_y) = Card(Ind(w))$; accordingly, $Card(Ind(w)) \leq t$. Let us now suppose u such that $\xi(u) = yH$ and $u <_e w$; then by definition of $<_e$, we have that $weight(\psi(u)) \leq Card(Ind(u)) \leq Card(Ind(w))$, thus, $\psi(u) = e_y$, i.e., there exists $v \neq 1$ such that $u = wv$; therefore, by Proposition 1.ii we have that $w <_e u$.□

If the given linear code is a perfect code then for every $y \in F_q^n$ Theorem 1 holds, although $<_e$ is not admissible. Results like Proposition 1.1 from i to iii in [BBM] are also valid for this case working with the reduction determined by $<_e$ and the congruence $R(C)$ [4]. Then the FGLM algorithm 2.1 in [BBM] specialized for the case of monoids in Section 4 can be applied to the monoid associated with the linear code and it will work in the usual way, that is, the set N and G built by the algorithm are really the full set of all the normal forms and the border basis with respect to $<_e$, respectively. To give a proof of these facts is out of the scope of this paper. Moreover, we do not need these strong results in order to achieve our goal: a decoding algorithm for a general linear code. For a general linear code these properties of N and G do no longer hold.

The following example will help to clarify the comments given above.

Example 1. *Assume we have a linear code over F_p^n with $p \geq 5$, $n \geq 4$ and $\prec$ is chosen to be the pure lexicographical order.*

Suppose we have $w = x_1^2x_2^3x_3^2$ a normal form, that is, the minimal word according to $<_e$ among all u satisfying $\xi(u) = \xi(w)$ then:

$$\begin{aligned} w = x_1w_1, &\quad w_1 = x_1x_2^3x_3^2\,; \\ w = x_2w_2, &\quad w_2 = x_1^2x_2^2x_3^2\,; \\ w = x_3w_3, &\quad w_3 = x_1^2x_2^3x_3. \end{aligned}$$

But, from the minimality of w, we cannot conclude that the w_i's are normal forms and this situation can not happen in the set of normal forms when the

[4] see [MR1] and [MR2] for details about reductions in monoids.

order is admissible as in [MMM] and [BBM]. For every w_i we could have the following:

$$\begin{aligned} w_{1_e} &= x_2^2 x_3^3, \\ w_{2_e} &= x_1 x_3^3, \\ w_{3_e} &= x_1 x_4, \end{aligned}$$

where each w_{i_e} is the normal form corresponding to $\psi(w_i)$. It is clear that for all $i \in [1, 3]$ $w_i >_e w_{i_e}$, but the reader can easily check that $w <_e x_i w_{i_e}$ for all $i \in [1, 3]$.

4 FGLM and linear codes

In this section we show how to use the FGLM techniques to compute the function Matphi [5] over a set of canonical forms N built by the algorithm. But this set N does not coincide with $N_{<_e}$, the set of all the normal forms in the sense of the reduction determined by $<_e$ and the congruence $R(C)$. However, N contains a very important subset of this set and precisely this fact will allow us to use the function Matphi for decoding a received word very fast.

The sources for the algorithm presented in this section have been [FGLM] and [BBM], particularly the procedure Matphi in the first paper and the Section 4 of the last paper (specializations to monoid and groups algebras).

In the following we give the essential properties that N and Matphi should have; we do it in order to be quite self contained and to clarify the differences between our set N and function Matphi and the ones given in the previous literature.

Properties of a set of canonical forms.

(i) $1 \in N$ and $N \subset [\,X\,]$.

(ii) $Card(N) = q^{n-k}$.

(iii) Two different words of N determine distinct coset module $R(C)$.

(iv) For all $w \in N$ there exists $x \in X$ such that $w = w'x$ and $w' \in N$.

[5] See [FGLM] for the definition of the function Matphi and [BBM] for the specific case of monoid rings.

Note that by property (iv), in the situation of the Example 1 given before, although w is a normal form it cannot be in N if at least one of w_i's in the example belongs to N, and if w_{i_e} belongs to N for all $i \in [1, 3]$ then by property (iii) above, $w_i \notin N$ for all $i \in [1, 3]$ and this implies that w would not be in N despite is a normal form.

Properties of the function Matphi:

(i) Matphi is a mapping ϕ from $N \times X$ onto N [6].

(ii) For all $(w, x) \in N \times X \quad \xi(\phi(w, x)) = \xi(wx)$ [7].

Our purpose will be to show that the algorithm presented herein builds such a set N and a function Matphi with the required properties.

Algorithm 1. *(FGLM for linear codes)*

Let us start making some references to some subroutines of the algorithm.

InsertNexts$[w, List]$ inserts properly the products wx (for $x \in X$) in $List$ and sorts it by increasing ordering with respect to the ordering $<_e$.

NextTerm$[List]$ removes the first element from $List$ and returns it.

Member$[v', \{v_1, \ldots, v_r\}]$ returns $True$ if $v' \in \{v_1, \ldots, v_r\}$ and $False$ otherwise. When the output is True, it also returns an additional output with the position j such that $v' = v_j$.

Following Remark 1, the variables will be represented with only one index.

Input: p, n, m, H the defining parameters for a given linear code.

Output: N, ϕ.

$List := \{1\}$; $N := \emptyset$; $r := 0$;
While $List \neq \emptyset$ **do**
  $w :=$ **NextTerm**$[List]$;
  $v' := \xi(w)$;
  $(\Lambda, j) :=$ **Member**$[v', \{v_1, \ldots, v_r\}]$;
 If $\Lambda =$ **True**
 then for each k such that $w = ux_k$ with $u \in N$ do
    $\phi(u, x_k) := w_j$;
      else $r := r + 1$;

[6] The image of (w, x) will be denoted by $\phi(w, x)$.

[7] it means that $\phi(w, x)$ is the representative element for the coset determined by $\xi(wx)$.

10. $v_r := v'$;
11. $w_r := w$; $N := N \cup \{w_r\}$;
12. $List :=$ **InsertNexts**$[w_r, List]$;
13. for each k such that $w = ux_k$ with $u \in N$ do
14. $\phi(u, x_k) := w$;
15. Return$[N, \phi]$.

Justification of the algorithm:

By N^0 we will refer to the subset of N that has been built by the algorithm until a given step and by $Eval(N^0)$ we denote the set of evaluations $(\{\xi(w) \mid w \in N^0\})$.

Termination of the algorithm comes from the fact that $List$ can not have more than nmq^{n-k} elements, because $Card(N) \leq q^{n-k}$ (the cardinal of F_q^{n-k}) and, each time a new word w is added to N; $InsertNext$ will insert in $List$ the mn products of w by the elements of X.

Now, what should correctness mean in this case? At the end, the algorithm has computed a set N and the function ϕ. The correctness of this algorithm means that the set N and the function ϕ satisfy the properties required above for a set of canonical forms and the function Matphi, respectively.

Theorem 2. (Correctness of Algorithm 1) *When Algorithm 1 stops, then the set N and the function ϕ satisfy the properties of canonical forms and Matphi, respectively.*

Proof: First let us see that N indeed has the required properties:

(i) results from the Steps 1 and 12.

(ii) will be proved later on (see (i) of Remark 5).

(iii) is a consequence of the Steps 4 (where the coset associated to w is computed), 5 (where one verifies whether the corresponding coset has been already seen), 10 and 11 (where the coset still not considered is introduced in two forms: as its representative word w and as the corresponding representative vector $\xi(w)$, the syndrome of w).

(iv) is guaranteed by the Steps 11 and 12.

Now, regarding the properties that the function Matphi should have, let us consider $u \in N$ and $x \in X$. There are two possible paths the algorithm could follow for the pair (u, x):

(P1) $\xi(ux)$ has been already seen: $\phi(u,x)$ would be w_j (see Step 6), which is already in N and lies in the same coset as ux. In this case, $\phi(u,x)$ is defined in Step 8.

(P2) Otherwise: $\phi(u,x)$ should be equal to ux. This is guaranteed by the assigment in Step 14.

□

The reason for keeping *List* ordered will be discussed later, it is mainly for guaranteeing that a certain subset of $[\,X\,]$ is contained in N. Now, we know that after applying this algorithm we have two things, a set N and the function Matphi satisfying the required properties.

5 Decoding linear codes

In this section we design an easy to understand and fast decoding algorithm which takes as its input the output of Algorithm 1. Previously, we have to introduce a canonical form function that will be denoted by CF. In fact CF is defined in a recursive way:

$$\begin{array}{l} CF: \ [\,X\,] \ \longrightarrow \ N \\ CF(1) := 1; \\ CF(w) := \phi(CF(u), x_k), \\ \text{where } w = ux_k \text{ and } u \in [\,X_k\,] = [\,x_1, \ldots, x_k\,]. \end{array} \tag{4}$$

Remark 4. *The decomposition $w = ux_k$ means that x_k is the greatest variable that divides w, so, the above factorization of w is uniquely determined; one can continue in this way, now with $CF(u)$, until $CF(1)$ is reached.*

Lemma 1 and Theorem 3 show some important properties that connect the function CF with our previous knowledge.

Lemma 1. ("Canonical form" of the error vectors) *Let v be a word in standard representation satisfying $Card(Ind(v)) \leq t$. Then, $CF(v) = v$.*

Proof: Let us do the proof by induction on the length of v. By definition of CF, the property holds for length zero; hence, let us assume that the property is valid for every word whose length is less than n and satisfies the conditions of the lemma. Let now v be a word in standard representation whose length is n and $Card(Ind(v)) \leq t$. Decompose $v = ux_k$, where $u \in [\,X_k\,]$. Then:

$$CF(v) = \phi(CF(u), x_k) = \phi(u, x_k) \text{ (by induction hypothesis).}$$

Now, the key point is that by Theorem 1 v is a normal form. Therefore, $\phi(u, x_k) = ux_k$ because, as *List* is built by taking into consideration $<_e$, the coset associated with ux_k appears by the first time in the Algorithm 1 when ux_k is considered. □

One of the must important theorems of this paper is the following.

Theorem 3. (Every element has "Canonical Form") *For every element w in $[\,X\,]$, $CF(w)$ is the unique element in N with the same syndrome as w, that is:*

$$\xi(w) = \xi(CF(w)).$$

Proof: Uniqueness comes from the property (iii) of the set N. Let now w be an arbitrary word. We decompose w in the following form:

$$w = x_{i_1} \ldots x_{i_k}, \text{ where } x_{i_j} \leq_e x_{i_{j+1}}.$$

Thus $\xi(CF(w)) = \xi(\phi(CF(x_{i_1} \ldots x_{i_{k-1}}), x_{i_k}))$ (by definition of CF), and $\xi(\phi(CF(x_{i_1} \ldots x_{i_{k-1}}), x_{i_k})) = \xi(CF(x_{i_1} \ldots x_{i_{k-1}})x_{i_k})$ (by the property of the function ϕ).

If one continues now with $x_{i_{k-1}}$, until $CF(1)$, one can obtain the following equality:

$$\xi(CF(w)) = \xi(CF(1)x_{i_1} \ldots x_{i_k}) \text{ (ξ is a morphism from $[\,X\,]$ to F_q^{n-k}).}$$

But $CF(1) = 1$, so we are done.□

Remark 5.

(i) *As a direct conclusion of this theorem we have that the cardinality of N is q^{n-k}, because it is known, from coding theory, that there are exactly this amount of syndromes; and, from Theorem 3 we have that for each syndrome there exists exactly one element in N with this syndrome.*

(ii) *We know from Theorem 1 that, if* $Card(Ind(w)) \leq t$ *and* w *is in standard representation then* w *is a normal form, that is,* $w \in N_{<_e}$*. Now, from Lemma 1, we also have that* $w \in N$ *and precisely this is the important subset of* $N_{<_e}$ *that is a subset of* N*: the set of all the standard representations of the error vectors with weight at most* t*. Note that if* C *is a perfect code then this set is not only a subset of* $N_{<_e}$ *and* N*, but it is exactly equal to these two sets, that is,* $N_{<_e} = N$ *and, from this, we have in the case of perfect codes stronger properties than the ones that were previously stated for* N*.*

(iii) *It is usual in Gröbner Bases Theory that the names "canonical" and "normal" identify the same object. This is not the case in the framework of this paper, where the canonical form of a word* w *stands for the coset leader of the class of* w *modulo* $R(C)$ *(see Theorem 3), while its normal form is the least word w.r.t.* $<_e$ *that lies in the same coset as* w*. In Lemma 1, the reader can see a case where both names represent the same object.*

Theorem 4. (Decoding) *Let* w *be such that* $\psi(w) \in B(C, t)$*. Then* $\psi(CF(w))$ *is the error vector corresponding to* $\psi(w)$*.*

Proof: Let w_e be the standard representation of the corresponding error vector of $\psi(w)$. Consequentely, $Card(Ind(w_e)) \leq t$ and, by Lemma 1, $w_e \in N$. But $\psi(w)H = \psi(w_e)H$, which amounts to $\xi(w) = \xi(w_e)$. On the other hand, by Theorem 3, $\xi(w) = \xi(CF(w))$. So, $\xi(w_e) = \xi(CF(w))$ and therefore, $w_e = CF(w)$ [8].$\square$

Note that from Theorem 4 we can already extract a decoding algorithm. It is only necessary to know the error-correcting capability of the code. CF will always report a value no matter if $y \notin B(C, t)$, while the result of Theorem 4 only holds for $y \in B(C, t)$. If it were possible to know t at the end of the computation of the function Matphi, then the problem could be solved because, when $Card(Ind(CF(w))) > t$, we report an error instead of giving $\psi(CF(w))$ as the error vector. This is because in this case $\psi(w) \notin B(C, t)$.

Taking into consideration these arguments we are going to give a theorem that will allows us to compute t (the error-correcting capability of the code) in the computation process of the function Matphi. By this way, we are given to the user the two possibilities, either one assume that the possibility of an

[8] there exists only one element in N with the same value for ξ.

error is very improbable and one would like to get a result for every received word, or, if the received word could not be corrected by at most t errors then an error in the transmission process is reported. The decision is up to the user.

Theorem 5. (error-correcting capability of a given linear code) *Let List be the list of words in Step 3 of Algorithm 1 and let w be the first element analyzed by* NextTerm[*List*] *such that w does not belong to N and w is in standard representation. Then*

$$t = Card(Ind(w)) - 1.$$

Proof: The important fact is that, due to $<_e$, the elements in *List* are organized by levels with regard to $Card(Ind)$. That is, there cannot be considered a word w of $Card(Ind)$ equal to $s+1$ if all the words in *List* with $Card(Ind)$ less than or equal to s have not been considered before.

First $Card(Ind(w)) \geq t+1$ because of Lemma 1. We need to prove the equality. It is known, from the theory of linear codes, that t equals the greatest integer less than $(d-1)/2$. We know, from the theory of linear codes, that there exists a codeword c with weight exactly equal to d (see [vL], [CLO]).

Let us represent, for the sake of simplicity, such a c by $c = (c_{i_1} c_{i_2} \ldots c_{i_d})$, where the c_{i_j}'s denote only the components different from zero. Define y_1 and y_2 as follows:

$$y_1 := (c_{i_1} c_{i_2} \ldots c_{i_{t+1}}), \quad y_2 := -(c_{i_{t+2}} \ldots c_{i_d}).$$

Let w_{y_1} and w_{y_2} be the standard representations of y_1, y_2, respectively. Then:

$$w_{y_1} = w_{1_t} \prod_{l=1}^{s_1} x_{i_{t+1} j_l}, \quad w_{y_2} = w_{2_t} \prod_{l=1}^{s_2} x_{i_d k_l};$$

where each of the w_{i_t} above has $Card(Ind(w_{i_t})) \leq t$, for $i = 1, 2$.

Since w_{1_t} is in standard representation and $Card(Ind(w_{1_t})) = t$ by Lemma 1 $w_{1_t} \in N$. Then $w_{1_t} x_{i_{t+1} j_1} \in List$, if $w_{1_t} x_{i_{t+1} j_1} \notin N$ we have that the statement of the theorem holds because of the equality $Card(Ind(w_{1_t} x_{i_{t+1} j_1})\ ) = t+1$. Then let us assume that $w_{1_t} x_{i_{t+1} j_1} \in N$. But for the same reason the complete sequence $\{w_{1_t} x_{i_{t+1} j_1}, w_{1_t} x_{i_{t+1} j_1} x_{i_{t+1} j_2}, \ldots, w_{y_1}\}$ would have to be a subset of N, otherwise we would have the existence of such an element that satisfies the theorem. So w_{y_1} must be in N. The following two cases are distinguished:

(i) d is odd: $Card(Ind(w_{y_2})) = t$ and therefore, $w_{y_2} \in N$ by applying Lemma 1.

(ii) d is even: For the same reason as w_{y_1}, we would have that w_{y_2} must be also in N.

Thus w_{y_1} and w_{y_2} would have to be in N. But $y_1 = c + y_2$ and this implies $\xi(w_{y_1}) = \xi(w_{y_2})$. Therefore, not both words can be elements of N at the same time.$\square$

The algorithm for decoding will be presented now. First let us explain some procedures used by the algorithm.

Read: Reads the vector y and returns its standard representation in $[X]$.
NextVar: Returns the first index k of a variable such that $x_k \in$ *Supp(w)* and then computes $w := w/x_k$. If $w = 1$, $NextVar$ returns 0.

Remember that x_{ij} corresponds to $x_{(i-1)m+j}$ when we use only one index instead of two (see Remark 1).

Algorithm 2. *(The Algorithm for decoding)*

Input: A received vector y.

Output: The codeword corresponding to y, if $y \in B(C, t)$,
and the error message "more than t errors" otherwise.

1. $w :=$ **Read**(y); $w_e := 1$;
2. $i :=$ **NextVar**$[w]$;
3. **While** $i \neq 0$ **do**
4. $w_e := \phi(w_e, x_i)$;
5. $i :=$ **NextVar**$[w]$;
6. **If** $weight(\psi(w_e)) > t$ **then** Return["more than t errors"]
7. **else** Return$[y - \psi(w_e)]$

Steps from 3 to 5 correspond to the computation of $CF(w)$, the correctness of this algorithm follows directly from Theorems 4 and 5. Before giving a complexity analysis of the Algorithm 2, we will give an example on the use of the FGLM approach for computing the set N and the function Matphi and afterwards on using this results for decoding.

6 Example

In our example we will work with the linear code over F_2^6 determined by the parity check matrix given in the left hand side of table 1, the set C of codewords is given in the right hand side:

Table 1.

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad \begin{aligned} C = \{&(0,\,0,\,0,\,0,\,0,\,0),\,(1,\,0,\,1,\,1,\,0,\,0),\,(1,\,1,\,1,\,0,\,0,\,1),\\ &(1,\,1,\,0,\,0,\,1,\,0),(0,\,1,\,0,\,1,\,0,\,1),(0,\,1,\,1,\,1,\,1,\,0),\\ &(0,\,0,\,1,\,0,\,1,\,1),\,(1,\,0,\,0,\,1,\,1,\,1)\}\,; \end{aligned}$$

and the minimum distance is $d = 3$, so, $t = 1$, the numbers of variables is 6, $\prec$ is set to be the pure lexicographical ordering with $x_{i+1} > x_i$.

Application of Algorithm 1:

$List := \{1\}$; $N := \{\}$; $r := 0$; $w := 1$; $\xi(1) = (0,0,0)$; $N := N \cup \{1\} = \{1\}$; $Neval := \{(0,0,0)\}$; $List := \{x_1, x_2, x_3, x_4, x_5, x_6\}$; $w := x_1$; $\xi(x_1) = (1,1,1)$; $N := \{1, x_1\}$; $Neval := \{(0,0,0),(1,1,1)\}$; $List := \{x_2, x_3, x_4, x_5, x_6, x_1^2, x_1x_2, x_1x_3, x_1x_4, x_1x_5, x_1x_6\}$. After analyzing x_6 we have the following stage:

$N := \{1, x_1, x_2, x_3, x_4, x_5, x_6\}$;
$Neval := \{(0,0,0),(1,1,1),(1,0,1),(0,1,1),(1,0,0),(0,1,0),(0,0,1)\}$;
$List := \{x_1^2, x_2^2, x_3^2, x_4^2, x_5^2, x_6^2, x_1x_2, x_1x_3, x_2x_3, x_1x_4, x_2x_4, x_3x_4, x_1x_5, x_2x_5, x_3x_5, x_4x_5, x_1x_6, x_2x_6, x_3x_6, x_4x_6, x_5x_6\}$; $\phi(1, x_i) = x_i \;\forall\, i \in [1,\, 6]$.

Because $t = 1$, there won't be more words in N with weight 1, this means, there is still one element left in N because there are 7 elements in N of a total of 8 (2^{6-3}). For $w \in \{x_1^2, x_2^2, x_3^2, x_4^2, x_5^2, x_6^2\}$, $w \notin N$, no more elements are added to *List* and the function Matphi in the new pairs is, $\phi(x_i, x_i) = 1$. For simplifying we will represent this function Matphi as relations, for example, $G := \{x_i^2 = 1 \mid i \in [1,\, 6]\}$[9].
For $w = x_1x_2$, $w \notin N$ because $\xi(x_1x_2) = \xi(x_5) = (0,1,0)$; $G := G \cup \{x_1x_2 = x_5\}$ then, x_1x_2 is the first element in *List* in standard representation that does not belongs to N [10], therefore, by Theorem 5, $t = 2 - 1 = 1$. $w = x_1x_3$ also does not belongs to N, note that *List* only can increase when we have

[9] this set G corresponds to the relations determined by the border basis in the case of [MMM], [BBM].

[10] Note that x_i^2 is not in standard representation.

a new element for N. $w = x_2x_3$, $\xi(x_2x_3) = (1, 1, 0) \notin Neval$ then, x_2x_3 is the last element that is included in N, $\xi(x_2x_3)$ is included in $Neval$ and the corresponding multiples are included in $List$. Now we continue the process until $List = \{\}$ and the output of the algorithm is:
$N = \{1, x_1, x_2, x_3, x_4, x_5, x_6, x_2x_3\}$;
$G := \{x_1^2 = 1,\ x_2^2 = 1,\ x_3^2 = 1,\ x_4^2 = 1,\ x_5^2 = 1,\ x_6^2 = 1,\ x_1x_2 = x_5,\ x_1x_3 = x_4,\ x_1x_4 = x_3,\ x_2x_4 = x_6,\ x_3x_4 = x_1,\ x_1x_5 = x_2,\ x_2x_5 = x_1,\ x_3x_5 = x_6,\ x_4x_5 = x_2x_3,\ x_1x_6 = x_2x_3,\ x_2x_6 = x_4,\ x_3x_6 = x_5,\ x_4x_6 = x_2,\ x_5x_6 = x_3,\ x_2^2x_3 = x_3,\ x_2x_3^2 = x_2,\ x_1x_2x_3 = x_6,\ x_2x_3x_4 = x_5,\ x_2x_3x_5 = x_4,\ x_2x_3x_6 = x_1\}$.

Using the Matphi function for decoding (Algorithm 2): The Matphi function in practical application can be implement it with pointers and instead of having the elements in N as words we store the vector associated with each word in N, so, the decoding process will be faster in this way. The reader will see in the examples that the decoding process will be done without any search and without any arithmetical operation. It consists only of a walk through the Matphi where the path is indicated by the representation of the received word.

Before giving two examples of how to decode, the shape of the function Matphi in this case is given below. In the first argument of each component of Matphi the corresponding error vector is stored, in the second argument we store 1 if the error vector in the first argument has weight at most t and zero otherwise, and the list of mn components in the third argument correspond to pointers to the values $\phi(w, x_i)$ for $i \in [1, nm]$.

$$\begin{aligned}
[\ & [[0,0,0,0,0,0],\ 1,\ [2,3,4,5,6,7]],\ [[1,0,0,0,0,0],\ 1,\ [1,6,5,4,3,8]],\\
& [[0,1,0,0,0,0],\ 1,\ [6,1,8,7,2,5]],\ [[0,0,1,0,0,0],\ 1,\ [5,8,1,2,7,6]],\\
& [[0,0,0,1,0,0],\ 1,\ [4,7,2,1,8,3]],\ [[0,0,0,0,1,0],\ 1,\ [3,2,7,8,1,4]],\\
& [[0,0,0,0,0,1],\ 1,\ [8,5,6,3,4,1]],\ [[0,1,1,0,0,0],\ 0,\ [7,4,3,6,5,2]]\ ].
\end{aligned}$$

Let us show how Algorithm 2 works for two different received word.

(i) $y \in B(C, t)$: $y = (1, 1, 0, 1, 1, 0)$; $w_y := x_1x_2x_4x_5$; $\phi(1, x_1) = x_1$; $\phi(x_1, x_2) = x_5$; $\phi(x_5, x_4) = x_2x_3$; $\phi(x_2x_3, x_5) = x_4$, this means $e = \psi(x_4) = (0, 0, 0, 1, 0,\ 0)$, $weight(e) = 1$ then, the codeword co- rresponding to y is $c = y - e = (1, 1, 0, 0, 1, 0)$, the reader can see in table

1 above that this output c belongs to C.

(ii) $y \notin B(C,t)$: $y = (0,1,0,0,1,1)$; $w_y := x_2x_5x_6$; $\phi(1,x2) = x_2$; $\phi(x_2,x_5) = x_1$; $\phi(x_1,x_6) = x_2x_3$; $e = (0,1,1,0,0,0)$; $weight(e) > 1$ then, we report an error in the transmission process, in this case the reader can check that the vector y is out of the set $B(C,1)$ for the set C given in table 1. Note that we could also give the value $y - e$ as a result, this could be useful for applications of codes when to give a result is always necessary.

7 Complexity analysis

First of all the complexity analysis of Algorithm 2 will be presented below. The reader will see that this very good complexity suggests to try out this approach on some linear codes with practical interest. Limitation of this method will be discussed later.

Theorem 6. (Complexity of Algorithm 2.) *Algorithm 2 computes the corresponding codeword of the received word in $\mathcal{O}((p-1)mn)$ operations.*

Proof: In Step 1, for each power of α one could have the corresponding inner representation as $\alpha^s = \sum_{j=1}^{m} \beta_{sj}\alpha^{j-1}$, $s = 1 \ldots q-2$. This assumption is very natural for every algorithm for decoding since in the literature it is generally assumed that operations in F_q take constant time. This is only possible if one considers the finite field fixed and out of the complexity or if one stores additional information in order to compute the operations in F_q in constant time. Taking into account these arguments, Step 1 in Algorithm 2 consumes n operations for reading the received word and transforming it into the internal structure for carrying out the algorithm. Step 4 is an assignment of a coefficient in a matrix, and Steps 2 and 5 are like removing the first element in a list, so these steps take constant time. The loop from 3 to 5 is executed a number of times equal to the length of the received word represented as an element in $[\,X\,]$, the maximal length of a word in standard representation is equal to $(p-1)nm$. Then, Algorithm 2 computes the codeword corresponding to the received word in $\mathcal{O}((p-1)nm)$ operations. Note that if one considers the field F_q fixed, the algorithm takes linear time.$\square$

1- Build your chip and your decoder: Note that by this method the decoding process is completely automatic, without any arithmetical operation. Neither do we need to compute at the beginning the syndrome corresponding to the received word, nor do we need to find at the end by solving a system the corresponding message associated with the codeword. One can avoid this if Algorithm 1 takes as an input the parity check matrix associated with a generator matrix in Gauss Jordan form. If this is the case, at the end one only needs the codeword until the component k for reconstructing the message. In order to use this method two things are necessary: the first one, build your "chip", the Matphi in Algorithm 1 and store it in a practical way according to the use of the particular code; the second thing, to install your decoder, a program or device that is able to execute Algorithm 2 by using the "chip".

2- Comparison with other methods: It is well know that in the class of linear codes the class of Goppa codes [11] turns out to be of high practical interest because of the parameters of the codes and the algorithms for decoding. In many of the papers that have been quoted in the references (see [FR], [Du], [HøPe]), the algorithms for decoding the Goppa codes take $\mathcal{O}(n^3)$ operations. It is easy to see the difference between this complexity and the complexity of our algorithm. Normally n is greater than $q = p^m$ ($n > q$ is the most interesting case) and $(p-1)m$ is smaller than p^m. Therefore, the complexity of our algorithm is less than $\mathcal{O}(n^2)$ if $n \geq (p-1)m$. Note also that the bigger the difference $n - q$ is, the closer our algorithm is being linear in n. The approach of Sakata for decoding one point Goppa codes as generalization of the Berlekamp Massey algorithm is described in [SJH], a detailed complexity analysis is presented in that paper and is given by the following expression; $\mathcal{O}(a_1 n^2) + \mathcal{O}(q^{N+1}[a_1 + \ldots + a_N]) + \mathcal{O}(nNq^N)$, where the set $\{a_1, \ldots, a_N\}$ is a minimal set of generators for the semigroup of the nongap sequence. Note that already the third term in this complexity $\mathcal{O}(nNq^N)$ is bigger than the complexity of our algorithm. In [Sak] the author states that his approach for decoding a one point Goppa code has complexity $\mathcal{O}(n^a)$ with $2 < a \leq 3$, which is generally less than $\mathcal{O}(n^3)$, but it is always bigger than $\mathcal{O}(n^2)$. In [SJH] a special case is presented where the complexity is $\mathcal{O}(n^{5/2})$ and in this case our algorithm takes $\mathcal{O}(n^{5/4})$. In [JLJH] and [SJMJH] another special case is presented where the complexity is $\mathcal{O}(n^{7/3})$ and in this case our algorithm takes $\mathcal{O}(n^{4/3})$. Another advantage

[11] See [Pre] for treatment of algebraic geometric codes.

of our approach is that the algorithms known for Goppa codes are able to decode only up to the half of the designed minimum distance of the code [12], Algorithm 1 of this paper gives us the possibility of knowing in the computation process the true value t of the error-correcting capability of the code and by using Algorithm 2 one can correct up to t errors. The reader should also note that this method proposed here is for general linear codes and not for a particular family.

3- Some limitation of this method: Clearly the main limitation of this method is the memory required for keeping the information represented by $Matphi$, in subsection 5 an upper bound of the memory required for computing $Matphi$ in terms of the parameters of the code will be presented. In addition we will present the complexity analysis of Algorithm 1 and some practical comments for implementation. Before trying to apply this approach to a particular code, it is strongly recommended to analyze the memory required for computation of Algorithm 1, and the memory required for keeping the information of the $Matphi$ in order to use it for decoding by means of Algorithm 2. Also with the complexity analysis of Algorithm 1, it is possible to know an estimated time for Algorithm 1 in computing with a particular code. So, one really needs to fulfil these requirements in order to apply Algorithm 1 to a given code. But note that this is executed just once, in the preparation phase of the code. We only need one computer powerful enough for computing the Matphi for the code. Then, if the code is really big, at least the same computer that was able to execute Algorithm 1 will be able to decode by Algorithm 2. Due to the difference in the complexity functions between our algorithm and the algorithms given previously in the literature, our algorithm can be faster for big codes even when we have to store the Matphi in external memory. In this case, your "chip" is attached to the equipment (a PC for example), where you also have to install the decoder.

4- FGLM for linear codes, syndrome decoding and Todd Coxeter: Note that Algorithm 1 gives us a very efficient way of computing representative elements of all the cosets determined by the code (the coset leaders). All the error vectors with weight at most t are representative elements of their corresponding cosets. The space F_q^n contains q^n elements and we need only to look at mnq^{n-k} elements, where all of them give information for building the function Matphi. Also, in Subsection 5, the reader will see that Algo-

[12] In general less than the real minimum distance of the code.

rithm 1 can be adapted in order to get a complexity that depends linearly on $Card(N)$ [13], while in normal applications of FGLM techniques the number of operations is quadratic in this factor (see [BBM]). Consequently, the Matphi can be built without having to do a search and ordering procedure in a big list of elements. From the theoretical point of view Algorithm 1 computes nothing more than the cosets determined by the code and Matphi, that allows us to multiply cosets. This is the same idea as in the well know Todd-Coxeter algorithm, but note that the entry in that method is a set of relations, which contrasts with our algorithm where the input is a generating set of a given monoid with computable multiplication (in this case F_q^n). Similar ideas can be seen in [MRM], also in the sense of the relation between FGLM techniques and the Todd-Coxeter algorithm.

Algorithm 1 together with Algorithm 2 are essentially an efficient variant of the syndrome decoding method. As for any syndrome decoding method, the required memory is a problem. We propose an efficient way of storing and computing the necessary information. Matphi is a plane structure easy to implement and it allows us to avoid search in a big list in the decoding phase. Only the precomputed set of canonical forms is presented in the domain of Matphi. Given a received word, Algorithm 2 is able to get the corresponding codeword or detect an error, by following a concrete path through Matphi determined by the sequence of letters in the received word.

5- On some comment about implementation and requirements of Algorithm 1 : First, in practice it is not necessary at all to work with variables, the goal of introducing the variables is to allow us to present the results and the method in a more compacted way and much easier to understand. The necessary information to be kept is related with $Matphi$, $List$ and $Neval$.

The total amount of representative elements is q^{n-k}, $List$ can not have more than mnq^{n-k} elements and when an element w is removed from $List$ the corresponding information is added to $Matphi$. If w is a new element to be included in N, then the corresponding vector $\psi(w)$ is added to $Matphi$ and in all cases a pointer to the representative element in N of the coset corresponding to $\xi(w)$ is added to $Matphi$. Therefore, the number of elements in $List$ and $Matphi$ is less than mnq^{n-k}, assuming that c is the size of memory required for storing an element in F_q, the size of storing a vector in F_q^n would be cn and then, cmn^2q^{n-k} is an upper bound for the memory required

[13]In this case $Card(N) = q^{n-k}$.

for List and Matphi together. That is, in each moment of the computation with Algorithm 1, the size of *List* and *Matphi* together is bounded by this quantity.

For *Neval* the bound for the memory is $c_1(n-k)q^{n-k}$. Consequently, a bound for the memory required to be able to compute with Algorithm 1 for a given linear code is:

$$cmn^2q^{n-k} + c_1(n-k)q^{n-k} = q^{n-k}(cmn^2 + c_1n - c_1k).$$

Note that the smaller $n-k$ is, the less information we have to store. On the other hand, the size of the matrix *Matphi* is:

$$q^{n-k}(nc + nmc' + c''),$$

where c' is the space necessary for storing a pointer to a position in the matrix and c'' denotes the cost of storing another additional information. For example see the Matphi in the example of Section 6. The second component is an optional component that expresses whether the corresponding vector in the first component has weight less than or equal to t or not. By this way one can avoid the computation of the weight in the decoding process.

5.1- Complexity analysis of Algorithm 1: In Step 4 we consider that $w = ux_k$, where $u \in N$ was already analyzed by the algorithm and we compute $\xi(w)$ as $\xi(u) + \xi(x_k)$ which takes nc_1 operations in F_q.

For the function *Member* in Step 5, we recommend before starting the computation with Algorithm 1 to build *Neval* as a list of q^{n-k} elements and every element equal to zero. Then, after computing v' in Step 4, *Member* can first compute the isomorphic representation i of v' as a number in the numeric representation in base q. After this, $Neval[i] = 0$ means that w is a new element to be included in N and the component i of *Neval* is replaced by $[v', r]$, v' is needed for computing others $\xi(w')$ such that $w' = wx_j$ and r indicates the element in N corresponding to this syndrome $(\xi(w))$. Computing the number corresponding to v' takes $n-k$ multiplications and $n-k$ sums then, with this implementation of the function *Member* it takes only $(n-k)c_2$ operations. The other step that could not take constant time inside the loop is the one corresponding to *InsertNexts*.

The function of *InsertNexts* is to build the *List* in such a way that Lemma 1 and Theorems 4 and 5 holds. Taking into consideration that all the standard representations with $Card(Ind) \leq t$ are normal forms and that one does not need any particular representative element for the cosets corresponding to vectors that do not belong to $B(C,t)$, $List$ can be built in regard to $Card(Ind)$ and not necessarily to $<_e$ [14]. In this way, $List$ would be a list of lists and all the elements in the same component of $List$ would have the same $Card(Ind)$, i.e, we guarantee to keep the needed conditions and at the same time $List$ is built without any search and ordering operation. The function $NextTerm$ takes the first element in the list, that is the first component of $List$. When this first component becomes empty, it is removed from $List$ by $NextTerm$. By this way we have reduced the Step 12 to constant time [15]. In this approach every element will be introduced as many times as variables are contained in its support and because of this Algorithm 1 changes a little. The Steps 7 and 13 disappear. Step 6 also changes, now we do not collect every occurrence of w in $List$ but we add it to $List$ onces more when it appears in $InsertNexts$. As a consequence of these changes in Algorithm 1, in order to know whether $w \notin N$ (which corresponds to Step 6) it is not enough to check whether Λ computed in Step 5 is $True$ (which means $[\xi(w), j] \in Neval$ [16]), but in addition we need to check that $w \neq w_j$. Therefore, for verifying the conditions of Theorem 5 for computing t, we need to check whether w is in standard representation, $\Lambda = True$, and $w \neq w_j$. To verify if w is in standard representation can be done in constant time by considering that we know that $w = ux_k$ with $u \in N$ and $x_j \in X$, and for u we know already whether it is in standard representation or not. To compare two elements in F_q^n takes nc_3 operations. Then, the operations to be performed inside the while loop are:

$$nc_1 + (n-k)c_2 + nc_3 + c_4,$$

where c_4 corresponds to the rest of the steps in the loop that take constant time, thus, this number of operations are dominated by $\mathcal{O}(n)$, therefore, Steps from 2 to 15 take:

$$\mathcal{O}(n(nmq^{n-k})) = \mathcal{O}(mn^2q^{n-k}) \text{ operations,}$$

because exactly mnq^{n-k} elements will be analyzed by $NextTerm$.

Note that this is a reasonable complexity if one keeps in mind that this algorithm computes among all the q^n vectors in F_q^n the q^{n-k} representative

[14] See Propositions 2 and 3.

[15] In the usual case of FGLM this step dominates the complexity of the algorithm.

[16] See in 5.1 above how we propose to build Neval.

vectors for the cosets determined by the code. Moreover, these representative elements are not arbitrarily chosen. The algorithm also computes the $Matphi$, which gives us a way of multiplying cosets.

If one considers n and d fixed, it is an interesting problem in Coding Theory to find k as big as possible such that such a $[n, k, d]$ linear code exists. We know from the bounds given before that the better the code is (which means, if n and d are fixed, the bigger k is) the less information one needs to store and the fewer operations are needed in Algorithm 1.

References

[BBM] M. Borges-Trenard, M. Borges-Quintana, T. Mora. Computing Gröbner Bases by FGLM Techniques in a Noncommutative Setting. *J. Symbolic Computation* (to appear 2000).

[CCS] A. M. Cohen, H. Cuypers, H. Sterk (Eds). *Some Tapas of Computer Algebra.* Springer-Verlag, Berlin (1999).

[CLO] D. Cox, J. Little, D. O'Shea. *Using Algebraic Geometry.* Springer-Verlag, New York (1998).

[Du] I. M. Duursma. Majority Coset Decoding. *IEEE. Trans. On Inf. Theory*, vol. 39/3, pp. 1067-1070 (1993).

[FR] G.-L. Feng, T.R.N. Rao. Decoding Algebraic-Geometric Codes up to the Designed Minimum Distance. *IEEE Trans. On Inf. Theory*, vol. 39/1, pp. 37-45 (1993).

[FGLM] J.C. Faugere, P. Gianni, D. Lazard, T. Mora. Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering. *J. Symbolic Computation*, 16, pp. 329-344 (1993).

[HøPe] T. Høholdt, R. Pellikaan. On the Decoding of Algebraic-Geometric Codes. *IEEE Trans. On Inf. Theory*, vol. 41/6, pp. 1589-1614 (1995).

[MMM] M. G. Marinari, H. M. Möller, T. Mora. Gröbner Bases of Ideals Defined by Functionals with an Application to Ideals of Projective Points. *Applicable Algebra in Engeneering, Communication and Computing*, vol. 4, pp. 103-145 (1993).

[MR1] K. Madlener, B. Reinert. String Rewriting and Gröbner bases – A General Approach to Monoid and Group Rings. In *Proceedings of the Workshop on Symbolic Rewriting Techniques*, Monte Verita, Birkhäuser, pp 127-180, 1995 (printed 1998).

[MR2] K. Madlener, B. Reinert. Relating Rewriting Techniques on Monoids and Rings: Congruences on Monoids and Ideals in Monoid Rings. *Theoretical Computer Sciences*, 208, pp. 3-31 (1998).

[MRM] K. Madlener, B. Reinert, T. Mora. A note on Nielsen Reduction and Coset Enumeration. *Proc. ISSAC 98*, pp. 171-178 (1998).

[Pre] O. Pretzel. *Codes and Algebraic Curves.* Clarendon Press. Oxford (1998).

[SJH] S. Sakata, H. E. Jensen, T. Høholdt. Generalized berlekamp-Massey Decoding of Algebraic-Geometric Codes up to Half the Feng-Rao Bound. *IEEE Trans. On Inf. Theory*, vol. 41/6, pp. 1762-1768 (1995).

[SJMJH] S. Sakata, J. Justesen, Y. Madelung, H. E. Jensen, T. Høholdt. Fast Decoding of Algebraic-Geometric Codes up to the designed Minimum Distance. *IEEE Trans. On Inf. Theory*, vol 41/5, pp. 1672-1677 (1995).

[Sak] Shojiro Sakata. Gröbner Bases and Coding Theory. In *Gröbner Bases and Applications (Proc. of the Conference 33 Years of Gröbner Bases).* B. Buchberger, F. Winkler (eds.). Cambridge University Press, London Mathematical Society Lecture Notes Series, vol. 251, pp. 205-220 (1998).

[JLJH] J. Justesen, J. Larsen, H. E. Jensen, T. Høholdt. Fast Decoding of Codes from Algebraic Plane Curves. *IEEE Trans. On Inf. Theory*, vol. 38/1, pp. 111-119 (1992).

[vL] J. H. van Lint. *Introduction to Coding Theory (2nd ed.).* Springer-Verlag, Berlin (1992).