An FGLM Method for Decoding Linear Codes*

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

December 18, 2008

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.

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

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

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.

We will present in this work only essential proofs, the rest of the proofs and a more detailed analysis of this method can be found in [BWB].

1 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.

Theorem 1.1. (Characterization of a linear code by its parity check matrix) *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*

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

By this theorem 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.

Theorem 1.2. (The error vector system) *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 t errors at the most[2] can be represented as:

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

[2] which are considered in the previous theorem.

2 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}.

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 2.1. 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 [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 2.2. *(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 [3] term ordering on $[\,X\,]$.

Proposition 2.3. (Properties of $<_e$)

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

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

Unfortunately, the term ordering $<_e$ is not admissible. But Proposition 2.3 is enough for getting our decoding algorithm. Everything will look as a normal use of border bases like in [MMM], [BBM].

Definition 2.4. *(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.

Definition 2.5. *(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 2.6. 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 2.7. (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. 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 Theorem 1.2 of Section 1). Let, on the other hand, w be the standard representation of e_y; accordingly, $Card(Ind(w)) \leq t$ [4]. Let us now suppose u such that $\xi(u) = yH$ and $u <_e w$; then by definition of $<_e$, we have that $weight(\psi(u)) \leq Card(Ind(u)) \leq Card(Ind(w))$ and by applying again Theorem 1.2 in Section 1, $\psi(u) = e_y$, i.e., there exists $v \neq 1$ such that $u = wv$; therefore, by Proposition 2.3.ii we have that $w <_e u$. □

[3] 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 $\prec$.

[4] $Card(Ind(w)) \geq weight(\psi(w))$. The equality holds if w is in standard representation (see Proposition 2 and Remark 2 in [BWB]).

3 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 $w' \in N$.

Note that property (iv) makes the properties of the set N weaker than the ones for $N_{<_e}$ when the order is admissible.

Properties of the function Matphi:

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

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

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 2.1, the variables will be represented with only one index.

[5] See [FGLM] for the definition of the function Matphi and [BBM] for the specific case of monoid rings.
[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 $\xi(wx)$.

Algorithm 3.1. *(FGLM for linear codes)*
Input: p, n, m, H the defining parameters for a given linear code.
Output: N, ϕ.

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

See justification of the algorithm in [BWB], where the proofs of termination and correctness are given. We have proved in [BWB] that Algorithm 3.1 computes a set N and a function Matphi with the properties required above. The only property that is not obvious from the construction of Algorithm 3.1 is the property (ii) of the set N of canonical forms. This property will be a consequence of Theorem 4.3.

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.

4 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 3.1. Previously, we have to introduce a canonical form function that will be denoted by CF. In fact CF is defined in a recursive way:

$$\begin{aligned} &CF: \quad [\,X\,] \quad \longrightarrow \quad N \\ &CF(1) := 1; \\ &CF(w) := \phi(CF(u), x_k), \\ &\text{where } w = ux_k \text{ and } u \in [\,X_k\,] = [\,x_1, \ldots, x_k\,]. \end{aligned} \tag{3}$$

Remark 4.1. 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 4.2 and Theorem 4.3 show some important properties that connect the function CF with our previous knowledge.

Lemma 4.2. (Canonical form of the error vectors) *Let v be a word in standard representation satisfying $Card(Ind(v)) \leq t$. Then, $CF(v) = v$.*

The proof [8] is done by induction on the length of v. The key point is to use Theorem 2.7 in connection with the fact that *List* is built by taking into consideration $<_e$.

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

Theorem 4.3. (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 of canonical forms. Let now w be an arbitrary word. We decompose w in the following form: $w = x_{i_1} \dots x_{i_k}$, where $x_{i_j} \leq_e x_{i_{j+1}}$. Thus $\xi(CF(w)) = \xi(\phi(CF(x_{i_1} \dots x_{i_{k-1}}), x_{i_k}))$ (by definition of CF), and $\xi(\phi(CF(x_{i_1} \dots x_{i_{k-1}}), x_{i_k})) = \xi(CF(x_{i_1} \dots x_{i_{k-1}})x_{i_k})$ (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: $\xi(CF(w)) = \xi(CF(1)x_{i_1} \dots x_{i_k})$ (ξ is a morphism from $[X]$ to F_q^{n-k}). But $CF(1) = 1$, so we are done. □

Remark 4.4. **(i)** As a direct conclusion of this theorem we have that the cardinality of N is q^{n-k}.

(ii) We know from Theorem 2.7 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 4.2, 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 [9].

Theorem 4.5. (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 4.2, $w_e \in N$. But $\psi(w)H = \psi(w_e)H$, which amounts to $\xi(w) = \xi(w_e)$. On the other hand, by Theorem 4.3, $\xi(w) = \xi(CF(w))$. So, $\xi(w_e) = \xi(CF(w))$ and therefore, $w_e = CF(w)$ [10]. □

Note that from Theorem 4.5 we can already extract a decoding algorithm. It is only necessary to know the error-correcting capability of the code.

Theorem 4.6. (error-correcting capability) *Let List be the list of words in Step 3 of Algorithm 3.1 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*

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

[8] See the proof in [BWB].

[9] If C is a perfect code $N_{<_e} = N$.

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

See the proof of this theorem in [BWB]. The important fact is that due to $<_e$ the elements in *List* are organized by levels with regard to $Card(Ind)$.

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 2.1).

Algorithm 4.7. *(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. $\quad w_e := \phi(w_e, x_i)$;
5. $\quad i :=$ **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 4.5 and 4.6.

5 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. 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 $x_{i+1} > x_i$. Only essential parts of the computation will be described.

Table1.

$$\left|\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right|$$

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

Application of Algorithm 3.1:

$List := \{1\}$; $N := \{\}$; $r := 0$; $w := 1$; $\xi(1) = (0,0,0)$; $N := N \cup \{1\} = \{1\}$;

$Neval := \{(0,0,0)\}$; $List := \{x_1, x_2, x_3, x_4, x_5, x_6\}$; $w := x_1$; $\xi(x_1) = (1,1,1)$; $N := \{1, x_1\}$; $Neval := \{(0,0,0),(1,1,1)\}$; After analyzing x_6 we have the following stage: $N := \{1, x_1, x_2, x_3, x_4, x_5, x_6\}$; $\phi(1, x_i) = x_i \ \forall i \in [1, 6]$. For $w = x_1x_2$, $w \notin N$ because $\xi(x_1x_2) = \xi(x_5) = (0,1,0)$, x_1x_2 is the first element in $List$ in standard representation that does not belongs to N [11], therefore, by Theorem 4.6, $t = 2 - 1 = 1$. For $w = x_2x_3$, $\xi(x_2x_3) = (1,1,0) \notin Neval$ then, x_2x_3 is the last element that is included in N.

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 $i \in [1, nm]$.

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

Examples of the decoding process:

(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 corresponding 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.

6 Complexity analysis

First of all the complexity analysis of Algorithm 4.7 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. Limitations of our method will be discussed later.

Theorem 6.1. (Complexity of Algorithm 4.7.) *Algorithm 4.7 computes the corresponding codeword of the received word in $\mathcal{O}((p-1)mn)$ operations.*

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

The proof comes from the fact that in order to compute $CF(w)$ it is necessary to perform as many steps as the length of the word w. The maximal length of a word in standard representation is $(p-1)mn$, from where the complexity function follows.

Comparison with other methods: It is well known that in the class of linear codes the class of Goppa codes [12] 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 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). 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 to being linear in n. 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 of our approach is that the algorithms known for Goppa codes are able to decode only up to half of the designed minimum distance of the code [13] and, by using Algorithm 4.7 one can correct up to t errors [14]. The reader should also note that this method proposed here is for general linear codes and not for a particular family.

Clearly the main limitation of this method is the memory required for keeping the information represented by Matphi. We recommend to see in [BWB] the detailed analysis of the last section devoted to complexity analysis. We will show below some interesting bounds for the memory required and complexity function in the computation with Algorithm 3.1 for a given code. The meaning of the constant can be found in [BWB].

The memory required for Algorithm 3.1 is $cmn^2q^{n-k} + c_1(n-k)q^{n-k} = q^{n-k}(cmn^2 + c_1n - c_1k)$, the number of operations performed by Algorithm 3.1 is $\mathcal{O}(mn^2q^{n-k})$ operations. In order to use Algorithm 4.7 for decoding we need to keep the information contained in Matphi, whose size is given by $q^{n-k}(nc + nmc' + c'')$.

Note that in order to apply this method for decoding one really needs to fulfil the requirements stated above. But Algorithm 3.1 is executed just once, in the preparation phase of the code. The complexity of Algorithm 3.1 is reasonable if one keeps in mind that this algorithm computes among all the q^n vectors in F_q^n the q^{n-k} representative 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. 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 3.1 will be able to decode by Algorithm 4.7. Due to the difference in the complexity functions between our algorithm and the algorithms given previously in the

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

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

[14] The error-correcting capability of the code.

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" (Matphi) is attached to the equipment (a PC for example), where you also have to install the decoder [15].

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 3.1.

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).

[BWB] M. Borges-Quintana, F. Winkler, M. Borges-Trenard. FGLM Techniques Applied to Linear Codes – An Algorithm for Decoding Linear Codes. Techn. Rep. *RISC-Linz, RISC - 00-14*, J. Kepler Univ., Linz, Austria (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).

[15] A program or device that is able to execute Algorithm 4.7 by using the "chip" Matphi.

[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).

Address of the authors:

† *Departamento de Matemática, Fac. de Ciencias. Universidad de Oriente, Santiago de Cuba 90500, Cuba.*
mijail@csd.uo.edu.cu, mborges@csd.uo.edu.cu

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