AAECC **8**, 469–483 (1997)

AAECC
Applicable Algebra in Engineering, Communication and Computing

On the Decoding of Cyclic Codes Using Gröbner Bases

Philippe Loustaunau[1], Eric V. York[2,*]

[1] Department of Mathematical Sciences, George Mason University, Fairfax, VA 22030, USA (e-mail: ploust@gmu.edu)
[2] Department of Mathematics, University of Notre Dame, IN 46556-5683, USA (e-mail: eyork@nd.edu)

Received: October 16, 1996; revised version: April 5, 1997

Abstract. In this paper we revisit an algorithm presented by Chen, Reed, Helleseth, and Troung in [5] for decoding cyclic codes up to their true minimum distance using Gröbner basis techniques. We give a geometric characterization of the number of errors, and we analyze the corresponding algebraic characterization. We give a characterization for the error locator polynomial as well. We make these ideas effective using the theory of Gröbner bases. We then present an algorithm for computing the reduced Gröbner basis over $\mathbb{F}_2$ for the syndrome ideal of cyclic codes, with respect to a lexicographic term ordering. This algorithm does not use Buchberger's algorithm or the multivariable polynomial division algorithm, but instead uses the form of the generators of the syndrome ideal and an adaptation of the algorithm introduced in [11]. As an application of this algorithm, we present the reduced Gröbner basis for the syndrome ideal of the [23, 12, 7] Golay code, and a decoding algorithm.

Keywords: Decoding, Cyclic codes, Gröbner basis, Zero-dimensional ideals.

1 Introduction

From an algebraic viewpoint, cyclic codes have a very rich structure. Because of this, many elegant decoding procedures have been developed for codes in this class. Some of the classical ones are presented in [2, 20, 21]. Recently, several authors (e.g. [5, 6, 7, 8, 16]) have applied the theory of Gröbner bases to the problem of decoding cyclic codes. In [16] the author uses the theory of Gröbner bases to solve the key equation and presents a method for decoding with complexity equal to that of the Berlekamp-Massey algorithm [3]. In [7, 8] the author presents a method for decoding cyclic codes up to their true distance. This methods uses

* Partially supported by a fellowship from the Center for Applied Mathematics at the University of Notre Dame.
Correspondence to: P. Loustaunau

elimination theory and requires the computation of a Gröbner basis for every non-zero syndrome received. This is clearly undesirable given the fact that Gröbner bases computation (via Buchberger's algorithm) have exponential complexity. In [5, 6] the authors revisit the algorithm presented in [7, 8] and attempt to improve this algorithm by proposing the computation of a "generalized" Gröbner basis to be done at the beginning, for a generic syndrome. Using this "generalized" Gröbner basis, the authors propose a decoding method for cyclic codes similar to the method proposed in [7, 8], now eliminating the need for a Gröbner basis computation at each step. However, the complexity of computing this "generalized" Gröbner basis using Buchberger's algorithm (with lexicographic ordering) increases significantly, since the number of variables is much larger in the "generalized" Gröbner basis case than in the case of the Gröbner basis used in [7, 8], making the approach in [5] also impractical. In particular, if n is the length of the code, and t is the number of errors it can correct, then an estimate for the complexity of computing a Gröbner basis for the polynomials given in [6] with respect to the lexicographic ordering using Buchberger's algorithm is $\approx (n+1)^{\mathcal{O}(t^3)}$ (see [4, 11]). The method we propose will reduce this complexity to $\approx (n+1)^{\mathcal{O}(t)}$ (see Section 4).

The idea of applying Gröbner bases techniques and elimination theory for decoding is quite nice and was originally presented by Cooper in [7, 8]. We note however, that in [5], the authors attempt at generalizing this method is incomplete. In particular, after Theorem 2.3 in that paper, the authors claim that the reduced Gröbner basis for a zero-dimensional ideal in the variables $x_1, x_2, \ldots x_s, z_1, z_2, \ldots, z_t$, with respect to the lexicographic term ordering with $x_1 < x_2 < \ldots < x_s < z_t < z_{t-1} < \cdots < z_1$, contains exactly one polynomial in the variables $x_1, x_2, \ldots, x_s, z_t$, exactly one polynomial in the variables $x_1, x_2, \ldots, x_s, z_t, z_{t-1}$, and so on up to $x_1, x_2, \ldots, x_s, z_t, z_{t-1}, \ldots, z_1$. They then go on to build their entire algorithm on this claim. This claim is clearly not true (see the examples contained in this paper). Furthermore, as we mentioned above, the computation of Gröbner bases via Buchberger's algorithm has exponential complexity, thus severely limiting the size of cyclic codes to which the algorithm in [5] can be applied. In fact, on a *Sun SparcStation 10* with 32 megs of memory, running *SunOs 4.1.3*, we were not able to compute the Gröbner bases, using CoCoA, MACAULAY or the GB package in MAPLE's share library, even for small examples of cyclic codes.

In this paper we do two things. First, we put on solid theoretical grounds the techniques presented in [5] of using Gröbner bases for decoding cyclic codes, and we present an algorithm, based on these ideas, for the decoding of cyclic codes. Second, we show that one can compute a Gröbner basis for the syndrome ideal using an adaptation of the algorithm of Faugère, Gianni, Lazard, and Mora presented in [11].

The paper is structured as follows. In Section 2 we describe the ideal and the variety generated by the set of equations arising from the non-zero syndromes of a code $\mathscr{C}$. Then in Section 3 we use elimination and Gröbner bases theory to relate the error locator polynomial to a certain elimination ideal of the syndrome ideal. In Section 4 we outline a method for computing the reduced Gröbner basis for the syndrome ideal with respect to the lexicographic term ordering. In Section 5 we discuss some improvements on this algorithm when applied to the problem of computing Gröbner bases for binary cyclic codes. Finally, in Section 6, we present the reduced Gröbner basis for the syndrome ideal of the [23, 12, 7] Golay code.

Throughout this paper we use fundamental notions concerning Gröbner bases and coding theory. We assume that the reader is familiar with the basic principles

in each of these areas. We refer the reader to [1, 9] for a comprehensive introduction to the theory of Gröbner bases and to [19] for a comprehensive introduction to the theory of cyclic codes. For further applications of Gröbner bases to error-correcting codes see [14, 15, 22]. For an overview of the applications of Gröbner bases to error-correcting codes, as well as an alternative set of equations for decoding the Golay code via Gröbner bases see [10].

2 The Syndrome Variety

First we recall some basic facts about cyclic codes. Let $\mathscr{C}$ be a rate k/n cyclic code defined over $\mathbb{F}_q$ with generator polynomial $g(x)$ of degree $r = n - k$, where we assume $\gcd(n, q) = 1$. It is well known (see e.g. [19]) that $\mathscr{C}$ can be equivalently described as the set of polynomials $f(x)$ over $\mathbb{F}_q$ having degree less than n that vanish at the roots of $g(x)$. Let $\alpha \in \mathbb{F}_{q^m}$ be a primitive n-th root of unity, where $\mathbb{F}_{q^m}$ is the splitting field of $x^n - 1$. Let $\alpha^{i_1}, \alpha^{i_2}, \ldots, \alpha^{i_r}$ be the roots of $g(x)$, and $\{i_1, i_2, \ldots, i_r\} \subseteq \{0, 1, 2, \ldots, n-1\}$. Then $\mathscr{C}$ can be viewed as the $\mathbb{F}_q$-kernel of the parity check matrix:

$$H = \begin{pmatrix} 1 & \alpha^{i_1} & \alpha^{2i_1} & \cdots & \alpha^{(n-1)i_1} \\ 1 & \alpha^{i_2} & \alpha^{2i_2} & \cdots & \alpha^{(n-1)i_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{i_r} & \alpha^{2i_r} & \cdots & \alpha^{(n-1)i_r} \end{pmatrix}. \tag{1}$$

Let $\boldsymbol{c} \in \mathbb{F}_q^n$ be a code word and let $\tilde{\boldsymbol{c}}$ be the received message, then $\tilde{\boldsymbol{c}} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{e} = (e_0, e_1, \ldots, e_{n-1})' \in \mathbb{F}_q^n$ is the error vector. To compute $\boldsymbol{e}$ we first compute the syndrome $\boldsymbol{s} = (s_1, s_1, \ldots, s_r)' = H\tilde{\boldsymbol{c}} = H\boldsymbol{e} \in \mathbb{F}_{q^m}^r$ and we solve the following system of r equations:

$$e_0 + e_1\alpha^{i_j} + e_2\alpha^{2i_j} + \ldots + e_{n-1}\alpha^{(n-1)i_j} = s_j, \quad j = 1, \ldots, r. \tag{2}$$

If we assume that the minimum distance of $\mathscr{C}$ is $2t + 1$ and that there are at most t errors, and hence at most t non-zero coordinates in the vector $\boldsymbol{e}$, then there are several well-known algorithms (see e.g., [2, 20, 21] and more recently [6, 12, 13, 16]) to solve the system defined by (2). All of these are based on the following fundamental theorem:

Theorem 2.1 *Let $\mathscr{C}$ be a rate $\frac{k}{n}$ code of distance $d \geqq 2t + 1$. To each error pattern $\boldsymbol{e}$ with $\mathrm{wt}(\boldsymbol{e}) \leqq t$ there corresponds one and only one syndrome s.*

In what follows, we present an algorithm that solves System (2) using the theories of Gröbner bases and elimination.

For the remainder of this paper we will assume that the distance of $\mathscr{C}$ is at least $2t + 1$ and that the weight of the error vector $\boldsymbol{e}$ is τ where $\tau \leqq t$. We now express the solutions of System (2) as points in a variety defined by multivariable polynomials. As in [5], we introduce variables x_j, z_κ and y_κ for $j = 1, \ldots r$ and $\kappa = 1, \ldots, t$ and we consider the following polynomials

$$\begin{aligned} f_j &= y_1 z_1^{i_j} + y_2 z_2^{i_j} + \ldots + y_t z_t^{i_j} - x_j, \quad j = 1, \ldots, r, \\ h_\kappa &= z_\kappa^{n+1} - z_\kappa, \quad \kappa = 1, \ldots, t, \\ \ell_\kappa &= y_\kappa^{q-1} - 1, \quad \kappa = 1, \ldots, t. \end{aligned} \tag{3}$$

Next, we let

$$F = \{ f_j | j = 1, \ldots, r\} \cup \{h_\kappa | \kappa = 1, \ldots, t\} \cup \{\ell_\kappa | \kappa = 1, \ldots, t\}. \tag{4}$$

Let I be the ideal generated by F, and let $\mathscr{V}(F)$ be the variety defined by F, that is

$$\mathscr{V}(F) = \{\boldsymbol{p} \in \mathbb{F}_{q^m}^{r2t} | f_j(\boldsymbol{p}) = h_\kappa(\boldsymbol{p}) = \ell_\kappa(\boldsymbol{p}) = 0, j = 1, \ldots, r, \kappa = 1, \ldots, t\} = \mathscr{V}(I).$$

It is easy to see that $\mathscr{V}(F)$ contains $(q-1)^t(n+1)^t$ points, and so I is a zero-dimensional ideal. We will refer to $\mathscr{V}(F)$ as the *syndrome variety* and I as the *syndrome ideal*. Note that the variety $\mathscr{V}(F)$ contains all the information needed to decode any received word $\tilde{\boldsymbol{c}}$. Indeed, given a syndrome $\boldsymbol{s} = H\tilde{\boldsymbol{c}} = H\boldsymbol{e}$, where $\mathrm{wt}(\boldsymbol{e}) \leqq t$, there are points $\boldsymbol{p} \in \mathscr{V}(F)$ that uniquely determines the error locations and error values corresponding to $\boldsymbol{e}$, namely

$$\boldsymbol{p} = (\underbrace{s_1, s_2, \ldots, s_r}_{x\text{-coordinates}}, \underbrace{0, \ldots, 0, \alpha^{\ell_1}, \alpha^{\ell_2}, \ldots, \alpha^{\ell_\tau}}_{z\text{-coordinates}}, \underbrace{*, \ldots, *, \beta_1, \beta_2, \ldots, \beta_\tau}_{y\text{-coordinates}}), \tag{5}$$

where the non-zero entries of $\boldsymbol{e}$ are located in the $\ell_1, \ell_2, \ldots, \ell_\tau$ coordinates and have values $\beta_1, \beta_2, \ldots, \beta_\tau$ respectively, and where $*$ indicates that this coordinate can be any non-zero element of $\mathbb{F}_q$ (there are $t - \tau$ such $\boldsymbol{y}$-coordinates, corresponding to the $t - \tau$ zero $\boldsymbol{z}$-coordinates). In fact there are $\binom{t}{\tau}\tau!(q-1)^{t-\tau}$ such points in $\mathscr{V}(F)$ corresponding to the syndrome $\boldsymbol{s}$: points that are obtained from (5) by a permutation of the $\boldsymbol{z}$-variables and the corresponding permutation of the $\boldsymbol{y}$-variables. Let $\mathscr{V}_s$ be the set of the $\binom{t}{\tau}\tau!(q-1)^{t-\tau}$ points corresponding to the syndrome $\boldsymbol{s}$. Let S be the set of all possible non-zero syndromes corresponding to error patterns with weight at most t. Define $\mathscr{E} = \cup_{s \in S} \mathscr{V}_s$. It is easy to see that $\mathscr{E}$ contains $(q-1)^t \sum_{j=1}^{t} \binom{n}{j}\binom{t}{j} j!$ points. Also, all the information needed to decode any received message is in the set $\mathscr{E}$. Therefore it is natural to study the set $\mathscr{E}$, however, the polynomials in (3) define a much larger variety $\mathscr{V}(F)$.

Example 2.2 Consider the primitive narrow sense BCH code with $q = 2$, $n = 31$, $k = 11$, and $t = 5$. It follows from the above discussion that:

$$|\mathscr{E}| = (q-1)^t \sum_{j=1}^{t} \binom{n}{j}\binom{t}{j} j! = 24444275$$

$$|\mathscr{V}(F)| = (q-1)^t(n+1)^t = 33554432.$$

So $\mathscr{V}(F)$ has over nine million more points that $\mathscr{E}$, or in other words, over 25% of the points in $\mathscr{V}(F)$ are not in $\mathscr{E}$ and so do not correspond to error vectors.

To deal with this, one could add new polynomials to F that would restrict the variety $\mathscr{V}(F)$ to $\mathscr{E}$. Indeed, if we add the $\binom{t}{2}$ polynomials $z_\kappa z_\lambda \left(\frac{z_\kappa^n - z_\lambda^n}{z_\kappa - z_\lambda}\right)$ to the set F, then the variety corresponding to the enlarged set of polynomials is $\mathscr{E}$. This follows from the fact that the new polynomials force z_κ and z_λ to be such that either at least one of them is zero, or they are both non-zero and they are distinct. But this approach makes the computational treatment of the variety $\mathscr{V}(F)$ prohibitively expansive. So we will develop a method which will allow us to extract the decoding information from $\mathscr{V}(F)$, which is located in $\mathscr{E}$, without computing $\mathscr{E}$ explicitly.

The key to decoding using the variety $\mathscr{V}(F)$ is to observe that the number of errors is t minus the number of zero $\boldsymbol{z}$-coordinates in the point $\boldsymbol{p}$ in (5), and that the number of zero $\boldsymbol{z}$-coordinates in the point $\boldsymbol{p}$ can be computed by looking at various projections of the variety $\mathscr{V}(F)$ onto $(\boldsymbol{x}, z_1, \ldots, z_k)$-planes, $k = 1, \ldots, t$. Since

projection corresponds to the elimination of variables in the ideal I, and since the presence of the origin in a variety can be determined easily by inspecting the constant terms of the polynomials of the defining ideal, we have a method for decoding. We now make these ideas more precise.

First, recall the notion of projection.

Definition 2.3 *Let $X \subset \mathbb{F}_{q^m}^b$. For any $a \leqq b$ we define the projection of X onto the first a coordinates by*:

$$\prod_a(X) = \{\boldsymbol{p} \in \mathbb{F}_{q^m}^a \mid \exists \tilde{\boldsymbol{p}} \in \mathbb{F}_{q^m}^{b-a} \text{ such that } (\boldsymbol{p}, \tilde{\boldsymbol{p}}) \in X\}. \tag{6}$$

Now we state the main result of this section relating the number of errors to the projections of $\mathscr{V}(F)$.

Theorem 2.4 *Let ζ_κ be the $1 \times \kappa$ zero vector. Given $\tilde{\boldsymbol{c}} \in \mathbb{F}_q^n$ and its corresponding syndrome $\boldsymbol{s} = H\tilde{\boldsymbol{c}}$, there are τ errors in $\tilde{\boldsymbol{c}}$ if and only if $(\boldsymbol{s}, \zeta_\kappa) \in \prod_{r+\kappa}(\mathscr{V}(F))$ for all $\kappa \leqq t - \tau$ and $(\boldsymbol{s}, \zeta_{t-\tau+1}) \notin \prod_{r+t-\tau+1}(\mathscr{V}(F))$.*

Proof. Let $\boldsymbol{s}$ be a syndrome corresponding to an error vector of weight τ, $1 \leqq \tau \leqq t$. Then a point $\boldsymbol{p}$ as in (5) is in $(\mathscr{V}(F)$ and so we have $(\boldsymbol{s}, \zeta_\kappa) \in \prod_{r+\kappa}(\mathscr{V}(F))$ for all $\kappa \leqq t - \tau$. Now suppose that $\tilde{\boldsymbol{p}} = (\boldsymbol{s}, \zeta_{t-\tau+1}) \in \prod_{r+t-\tau+1}(\mathscr{V}(F))$. Then $\tilde{\boldsymbol{p}}$ extends to a point $\boldsymbol{p}_0 = (\tilde{\boldsymbol{p}}, \hat{\boldsymbol{p}}_0) \in \mathscr{V}(F)$. Let

$$\hat{\boldsymbol{p}}_0 = (\underbrace{\gamma_1, \gamma_2, \ldots, \gamma_{\tau-1}}_{z\text{-coordinates}}, \underbrace{\eta_1, \eta_2, \ldots, \eta_t}_{y\text{-coordinates}}) = (\boldsymbol{\gamma}, \boldsymbol{\eta}).$$

Clearly $\boldsymbol{\gamma}$ is not the zero vector since this would imply that $\boldsymbol{s} = 0$. Also, the non-zero coordinates of $\boldsymbol{\gamma}$ cannot all be distinct, since if they were, the non-zero coordinates of $\boldsymbol{\gamma}$ would represent an error pattern. Therefore we would have two distinct error vectors associated to the same syndrome (they arise from the two distinct points $\boldsymbol{p}$ and $\boldsymbol{p}_0$), contradicting Theorem 2.1. Therefore we must have $\gamma_i = \gamma_j$ for some i, j (in particular, $\boldsymbol{p}_0 \notin \mathscr{E}$). By the symmetry of (3) we can assume without loss of generality that $\gamma_1 = \gamma_2$. Now consider the point

$$\hat{\boldsymbol{p}}_1 = \begin{cases} (0,\ \gamma_2,\ \gamma_3, \ldots, \gamma_{\tau-1},\ \eta_1,\ \eta_1 + \eta_2,\ \eta_3, \ldots, \eta_t) & \text{if } \eta_1 + \eta_2 \neq 0 \\ (0,\ 0,\ \gamma_3,\ \gamma_4, \ldots, \gamma_{\tau-1},\ \eta_1,\ \eta_2, \ldots, \eta_t) & \text{otherwise.} \end{cases}$$

Then it is easy to check that $\boldsymbol{p}_1 = (\tilde{\boldsymbol{p}}, \hat{\boldsymbol{p}}_1) \in \mathscr{V}(F)$. We can proceed in this manner, eliminating all repeated coordinates in the $\boldsymbol{\gamma}$ vector. The remaining non-zero distinct coordinates of the resulting $\boldsymbol{\gamma}$ vector represent an error vector which corresponds to the syndrome $\boldsymbol{s}$, but with weight strictly less than τ. This is a contradiction to Theorem 2.1. Therefore $(\boldsymbol{s}, \zeta_{t-\tau+1}) \notin \prod_{r+t-\tau+1}(\mathscr{V}(F))$.

For the converse, suppose that $(\boldsymbol{s}, \zeta_\kappa) \in \prod_{r+\kappa}(\mathscr{V}(F))$ for all $\kappa \leqq t - \tau$ and $(\boldsymbol{s}, \zeta_{t-\tau+1}) \notin \prod_{r+t-\tau+1}(\mathscr{V}(F))$. Then $(\boldsymbol{s}, \zeta_{t-\tau})$ extends to a point $\boldsymbol{p} \in \mathscr{V}(F)$. Furthermore, using the same argument as above, this point can be related to a unique error vector of weight exactly τ. □

The next corollary follows directly from the proof of Theorem 2.4.

Corollary 2.5 *Let $\Gamma = \{\boldsymbol{p} \in \mathscr{V}(F) \mid \boldsymbol{p} = (\boldsymbol{s}, \zeta_{t-\tau}, *, *, \ldots, *)\}$. Then $\prod_{r+t-\tau+1}(\Gamma)$ contains exactly τ distinct points of the form $(\boldsymbol{s}, \zeta_{t-\tau}, \alpha^{\ell_i})$, where the error locations of $\tilde{\boldsymbol{c}}$* are given by ℓ_i, $1 \leqq i \leqq \tau$.

In the next section we turn our attention to developing a method for decoding cyclic codes using polynomial representations of these projections.

3 Elimination Theory and Decoding

Recall that the syndrome ideal I, is the ideal in $\mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$ generated by the polynomials in F, where we denote the variables $x_1, \ldots, x_r$ by $\boldsymbol{x}$ the variables $y_1, \ldots, y_t$ by $\boldsymbol{y}$, and the variables $z_1, \ldots, z_t$ by $\boldsymbol{z}$. Clearly we have $\mathscr{V}(I) = \mathscr{V}(F)$.

The following lemma is well-known and can be found in [1, Theorem 2.5.3]. By relating projection to elimination, it provides one of the two keys for developing the computational treatment of the Gröbner bases method for decoding.

Lemma 3.1 $\prod_{r+\kappa}(\mathscr{V}(F)) = \mathscr{V}(I \cap \mathbb{F}_q[\boldsymbol{x}, z_1, \ldots, z_\kappa])$.

The proof of Lemma 3.1 uses the fact that $\mathscr{V}(F)$ is finite, therefore any projection of the variety $\mathscr{V}(F)$ is finite, and hence closed in the Zarisky topology over $\mathbb{F}_q$ or its algebraic closure.

The second key to the decoding algorithm is the theory of Gröbner bases. We now review some basic notions that will be important for the actual implementation of this method. We refer the reader to [1] for a comprehensive introduction to the theory of Gröbner bases.

First we impose the lexicographic term ordering on $\mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$ with

$$x_1 < \ldots < x_r < z_1 < \ldots < z_t < y_1 < \ldots < y_t. \tag{7}$$

Then we define $\mathrm{lt}(f)$ to be the leading term of f with respect to the term ordering, where $f \in \mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$. A generating set $G = \{g_1, \ldots, g_t\}$ for the ideal I is a *Gröbner basis* for I if and only if

$$\mathrm{Lt}(I) = \langle \mathrm{lt}(f) | f \in I \rangle = \langle \mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t) \rangle = \mathrm{Lt}(G).$$

Gröbner bases can be computed using Buchberger's algorithm, and many computer algebra packages have this algorithm implemented (but not over arbitrary finite fields). Gröbner bases allow us to compute elimination ideals as the following theorem shows (see, for example, [1, Theorem 2.3.4]).

Theorem 3.2 *Let G be a Gröbner basis for I with respect to the term ordering described above. Then*

$$G_\kappa = G \cap \mathbb{F}_q[\boldsymbol{x}, z_1, \ldots, z_\kappa]$$

generates $I \cap \mathbb{F}_q[\boldsymbol{x}, z_1, \ldots, z_\kappa]$, and is a Gröbner basis for that ideal.

The ideals $\langle G_\kappa \rangle$ are known as *elimination ideals*. In view of Lemma 3.1 and Theorem 3.2 we can restate Theorem 2.4 and Corollary 2.5 in the following form:

Theorem 3.3 *There are τ errors in $\tilde{\boldsymbol{c}}$ if and only if for every $g \in G_\kappa$, we have $g(\boldsymbol{s}, \boldsymbol{\zeta}_\kappa) = 0$ for all $\kappa \leqq t - \tau$ and $g(\boldsymbol{s}, \boldsymbol{\zeta}_{t-r+1}) \neq 0$ for some $g \in G_{t-r+1}$.*

Corollary 3.4 *Let $G_{t-r+1} = \{g_1, g_2, \ldots, g_u\}$ and let $\boldsymbol{\xi}_{t-\tau}$ be the vector $(\boldsymbol{s}, \boldsymbol{\zeta}_{t-\tau}, z)$, where z is a new variable. Then the ideal $\langle G_{t-\tau+1}(\boldsymbol{\xi}_{t-\tau}) \rangle = \langle g_1(\boldsymbol{\xi}_{t-\tau}), g_2(\boldsymbol{\xi}_{t-\tau}), \ldots, g_u(\boldsymbol{\xi}_{t-\tau}) \rangle \in \mathbb{F}_q[z]$ is generated by the error locator polynomial. Furthermore*[1], *the error locator polynomial is in $\{g_1(\boldsymbol{\xi}_{t-\tau}), g_2(\boldsymbol{\xi}_{t-\tau}), \ldots, g_u(\boldsymbol{\xi}_{t-\tau})\}$.*

[1] The authors thank Prof. Teo Mora for pointing out references [17] and [18] which allowed the inclusion of this last statement in the corollary, and which improved the paper.

Proof. The first part of the Corollary follows directly from Theorem 2.4, Corollary 2.5, and Theorem 3.3. To prove the second part of the Corollary it is enough to show that $G_{t-\tau+1}(\boldsymbol{\xi}_{t-\tau})$ is a Gröbner basis for the ideal $\langle G_{t-\tau+1}(\boldsymbol{\xi}_{t-\tau})\rangle$, since, in the case of one variable, a generating set G for an ideal J is a Gröbner basis for J if and only if the gcd is in G. But $G_{t-r+1}(\boldsymbol{\xi}_{t-\tau})$ is the evaluation at all but one variable of a Gröbner basis, with respect to the lexicographic term ordering, for a zero dimensional ideal, namely I. That such an evaluation gives a Gröbner basis was proved in [17] and [18]. □

Remark 3.5. One could use the ideal $\langle G_{t-\tau}(\boldsymbol{\xi}_{t-\tau-1})\rangle$ to compute the error locator polynomial, since this ideal is generated by z times the error locator. This follows from the fact that, in this case, the variety corresponds to projection of points of $\mathscr{E}$ alone. Indeed, points of $\mathscr{V}(F)$, with $\boldsymbol{x}$-coordinates equal to some syndrome $\boldsymbol{s}$, which are not in $\mathscr{E}$, can be thought of as points in $\mathscr{V}_s$ where two or more of the zero $\boldsymbol{z}$-coordinates have been replaced by the same element of $\mathbb{F}_{q^m}$ (this follows from the argument in the proof of Theorem 2.4). Since the $t-\tau$ elimination ideal leaves only one $\boldsymbol{z}$-variable zero, this situation cannot occur. The same argument can be used to see that the generator for $\langle G_\kappa(\boldsymbol{\xi}_{\kappa-1})\rangle$ is $z^{n+1}-z$, for $\kappa < t-\tau$.

We now have a method to decode any incoming message. Indeed, given a syndrome, $\boldsymbol{s}$, we can evaluate the generators of G_κ, $\kappa = 1, \ldots, t$ successively, at $\boldsymbol{x} = \boldsymbol{s}$, and check whether the constant terms of the evaluated polynomials are zero. Once the first non-zero constant term is found, the error locator is computed and factored. The advantage of this method is that the difficult computation is done once, using the generic $\boldsymbol{x}$-, $\boldsymbol{y}$-, and $\boldsymbol{z}$-variables. After the elimination ideals have been computed with the generic variables, then finding the error locator is reduced to evaluating polynomials at the syndromes. We now take a detailed look at a simple example to emphasize the points discussed thus far:

Example 3.6 We consider the 3 error correcting [15, 5, 7] BCH code over $\mathbb{F}_2$. Since $t = 3$, we have 3 $\boldsymbol{z}$-variables, and since we are working over $\mathbb{F}_2$, there are no $\boldsymbol{y}$-variables. So the polynomials in (3) are then

$$z_1 + z_2 + z_3 + x_1,\ z_1^3 + z_2^3 + z_3^3 + x_2,\ z_1^5 + z_2^5 + z_3^5 + x_3,$$

$$z_1^{16} + z_1,\ z_2^{16} + z_2,\ z_3^{16} + z_3.$$

The Gröbner basis computation with respect to the lexicographic term ordering, with $x_1 < x_2 < x_3 < z_1 < z_2 < z_3$, gives us the following results (we omit the polynomials in the $\boldsymbol{x}$-variables alone, since they evaluate to zero at $\boldsymbol{x} = \boldsymbol{s}$):

$$\begin{aligned} G_1 = \{&z_1^{16} + z_1,\ z_1^3x_2 + z_1^3x_1^3 + z_1^2x_1x_2 + z_1^2x_1^4 + z_1x_3 + z_1x_1^2x_2 \\ &+ x_1x_3 + x_2^2 + x_1^3x_2 + x_1^6, \\ &z_1^3x_3 + z_1^3x_1^5 + z_1^2x_1x_3 + z_1^2x_1^6 + z_1x_2^9x_3^2 + z_1x_1^3x_2^8x_3^2 \\ &+ z_1x_2^4x_3^2 + z_1x_1^9x_2x_3^2 + z_1x_1^2x_3 + z_1x_1^{10}x_2^9 + z_1x_1^{13}x_2^8 + z_1x_1^{10}x_2^4 \\ &+ z_1x_1^4x_2 + z_1x_1^7 + x_1x_2^9x_3^2 + x_1^4x_2^8x_3^2 + x_1x_2^4x_3^2 + x_1^{10}x_2x_3^2 \\ &+ x_2x_3 + x_1^{11}x_2^9 + x_1^{14}x_2^8 + x_1^{11}x_2^4\} \end{aligned}$$

$$G_2 = G_1 \cup \{z_2^{16} + z_2,\ z_1 z_2^2 + z_2^2 x_1 + z_1^2 z_2 + z_2 x_1^2 + z_1^2 x_1 + z_1 x_1^2 + x_2$$
$$+ x_1^3,\ z_2^2 x_2 + z_2^2 x_1^3 + z_1 z_2 x_2 + z_1 z_2 x_1^3 + z_2 x_1 x_2 + z_2 x_1^4$$
$$+ z_1^2 x_2 + z_1^2 x_1^3 + z_1 x_1 x_2 + z_1 x_1^4 + x_3 + x_1^2 x_2,$$
$$z_2^2 x_3 + z_2^2 x_1^5 + z_1 z_2 z_3 + z_1 z_2 x_1^5 + z_2 x_1 x_3 + z_2 x_1^6 + z_1^2 x_3 + z_1^2 x_1^5$$
$$+ z_1 x_1 x_3 + z_1 x_1^6 + x_2^9 x_3^2 + x_1^3 x_2^8 x_3^2 + x_2^4 x_3^2 + x_1^9 x_2 x_3^2 + x_1^2 x_3 + x_1^{10} x_2^9$$
$$+ x_1^{13} x_2^8 + x_1^{10} x_2^4 + x_1^4 x_2 + x_1^7\}$$
$$G_3 = G_2 \cup \{z_3 + z_2 + z_1 + x_1\}.$$

This computation was done using the algorithm presented in Section 4.

Now we can apply the results of Section 3. Let α be a primitive element of $\mathbb{F}_{2^4}$, with $\alpha^4 + \alpha + 1 = 0$. Suppose that the zero codeword is sent and that the received message has one error in the second position. Then $s_1 = \alpha$, $s_2 = \alpha^3$ and $s_3 = \alpha^5$ and we have

$$G_1(\xi_0) = \{z^{16} + z\}, \quad G_2(\xi_1) = \{z^{16} + z,\ z\alpha^2 + z^2\alpha\}, \quad G_3(\xi_2) = \{z + \alpha\}.$$

Both $\mathscr{V}(G_1(\xi_0))$ and $\mathscr{V}(G_2(\xi_1))$ contain zero (with confirms, by Theorem 3.3, that there is one error), and $G_3(\xi_2)$ gives the error locator polynomial, by Corollary 3.4. Note that the generator of $\langle G_2(\xi_1)\rangle$ is $z^2 + z\alpha = z(z + \alpha)$, that is, as mentioned above, $\langle G_2(\xi_1)\rangle$ is generated by z times the error locator polynomial.

Now suppose that the received message has two errors, say in the second and fourth positions. Then $s_1 = \alpha + \alpha^3 = \alpha^9$, $s_2 = \alpha$ and $s_3 = \alpha^2 + \alpha + 1 = \alpha^{10}$ and we have

$$G_1(\xi_0) = \{z^{16} + z,\ z^3\alpha^{13} + z^2\alpha^7 + z\alpha^2,\ z^3\alpha^5 + z^2\alpha^{14} + z\alpha^9\},$$
$$G_2(\xi_1) = \{z^{16} + z,\ z^2\alpha^9 + z\alpha^3 + \alpha^{13},\ z^2\alpha^{13} + z\alpha^7 + \alpha^2,\ z^2\alpha^5 + z\alpha^{14} + \alpha^9\}.$$

$\mathscr{V}(G_1(\xi_0))$ contains zero (which confirms, by Theorem 3.3, that there are 2 errors), and $\langle G_2(\xi_1)\rangle$ gives the error locator polynomial, by Corollary 3.4. The polynomials in $G_2(\xi_1)$ which are not $z^{16} + z$ can be factored as follows:

$$z^2\alpha^9 + z\alpha^3 + \alpha^{13} = \alpha^9(z^2 + z\alpha^9 + \alpha^4) = \alpha^9(z + \alpha)(z + \alpha^3),$$
$$z^2\alpha^{13} + z\alpha^7 + \alpha^2 = \alpha^{13}(z^2 + z\alpha^9 + \alpha^4) = \alpha^{13}(z + \alpha)(z + \alpha^3),$$
$$z^2\alpha^5 + z\alpha^{14} + \alpha^9 = \alpha^5(z^2 + z\alpha^9 + \alpha^4) = \alpha^5(z + \alpha)(z + \alpha^3).$$

Note also that the two polynomials in $G_1(\xi_0)$ which are not $z^{16} + z$ are z times the error locator polynomial.

Finally suppose that the received message has three errors, say in the second, fourth, and seventh positions. Then $s_1 = \alpha + \alpha^2$, $s_2 = \alpha + \alpha^3$ and $s_3 = \alpha^5$ and we have

$$G_1(\xi_0) = \{z^{16} + z,\ z^3\alpha^7 + z^2\alpha^{12} + z\alpha^8 + \alpha^2,\ z^3 + z^2\alpha^5 + z\alpha + \alpha^{10}\}.$$

Since $\mathscr{V}(G_1(\xi_0))$ does not contain zero, there are 3 errors by Theorem 3.3, and $G_1(\xi_0)$ gives the error locator polynomial (by Corollary 3.4)

$$z^3\alpha^7 + z^2\alpha^{12} + z\alpha^8 + \alpha^2 = \alpha^7(z^3 + z^2\alpha^5 + z\alpha + \alpha^{10}) = \alpha^7(z + \alpha)(z + \alpha^3)(z + \alpha^6).$$

4 Fast Computation of G_κ

Computing Gröbner bases is in general very difficult, with doubly exponential complexity for arbitrary ideals and exponential for zero-dimensional ideals (such as I). In [5] the authors suggest using Buchberger's Algorithm for computing the Gröbner basis for I. In our situation though, we can improve greatly the computation with the following simple lemma:

Lemma 4.1 *The set F of generators for I is a Gröbner basis with respect to the lexicographic term ordering with*

$$y_1 < \ldots < y_t < z_1 < \ldots < z_t < x_1 < \ldots < x_r. \tag{8}$$

Indeed, with any term ordering where the $\boldsymbol{x}$-variables are greater than both the $\boldsymbol{z}$- and $\boldsymbol{y}$-variables, the leading terms of the polynomials given in F are relatively prime, and it is well-known that this condition implies that the set is a Gröbner basis. The problem with this is that it is a Gröbner basis for I but not with respect to the right term ordering. Instead of computing the desired Gröbner basis using Buchberger's algorithm, we can use linear algebra to transform the given Gröbner basis F into the desired one. The technique was introduced in [11], works only for zero-dimensional ideals, and has polynomial complexity in the degree of the ideal. We now outline this method.

Let $<_1$ be the term ordering given by (8) and let $<_2$ be the term orderng given by (7). We want to compute the reduced Gröbner basis G for I with respect to $<_2$. First we define the following map:

$$\begin{aligned} \mathscr{R}: \mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}] &\to \mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]/I \\ f &\mapsto r + I, \end{aligned} \tag{9}$$

where r is the unique remainder of the division of f by F using $<_1$ (the uniqueness follows from the fact that F is a Gröbner basis with respect to $<_1$).

Remark 4.2. Since F is a Gröbner basis with respect to $<_1$, the above map is a vector space homomorphism. We will make use of the map $\mathscr{R}$ to compute a Gröbner basis for the kernel with respect to $<_2$. Also note that $\mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]/I$ is isomorphic to $\mathbb{F}_q^D$, where $D = (n+1)^t(q-1)^t$. In fact a basis for this space is easily deduced from the structure of F and consists of all monomials of the form $\prod_{i=0}^{t} y_i^{a_i} z_i^{b_i}$, where $0 \leqq a_i \leqq q-2$, and $0 \leqq b_i \leqq n$.

Next, we order the power products X_i (i.e. the monomials in $\mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$) using $<_2$

$$X_0 <_2 X_1 <_2 X_2 <_2 \ldots <_2 X_\eta <_2 \ldots$$

Now any power product X_i is either reduced with respect to G, a leading power product of some polynomial in G, or a multiple of a leading power product of some polynomial in G. Moreover, a polynomial with leading power product X_η (with respect to $<_2$) is in I if and only if there exists $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_\eta) \in \mathbb{F}_q^\eta$ such that:

$$\mathscr{R}\left(\sum_{i=0}^{\eta} \sigma_i X_i \right) = \sum_{i=0}^{\eta} \sigma_i \mathscr{R}(X_i) = 0.$$

So the problem of deciding whether a polynomial with leading power product X_η is in G translates into finding an element σ in the kernel of the $(\eta+1) \times D$ matrix defined by the $\mathscr{R}(X_i)$'s. Since there are only finitely many reduced power products,

there are only finitely many such systems to solve. Moreover, once a polynomial in G has been found, we do not need to consider any power products which are a multiple of the leading term of that polynomial. We now outline the algorithm below (see [11] for a more detailed presentation).

INPUT:
$<_1$ and F.
OUTPUT:
G, the reduced Gröbner basis for I with respect to $<_2$.
LOCAL VARIABLES:
PowerProducts - the list of power products to be considered, ordered with respect to $<_2$.
LPP - a list containing the leading power products of G
MBasis - a list of pairs of the form $[X_i, \mathscr{R}(X_i)]$ where X_i is a basis element of $\mathbb{F}_q[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]/\langle G\rangle$
SUBROUTINES:
$\mathscr{R}$ - as defined by (9).
NextMonom - removes first element from PowerProducts and returns null if empty.
InsertNexts (X_i) - adds to the list PowerProducts the elements $x_1X_i, \ldots, x_rX_i, z_1X_i, \ldots, z_tX_i, y_1X_i, \ldots, y_tX_i$, and orders them with respect to $<_2$
BEGIN
LPP := [ ]; G := [ ]; MBasis := [ ];
PowerProducts := [ ]; $X_i := 1$;
WHILE $X_i \neq$ null DO
IF X_i is not divisible by some element of LPP
THEN:
compute $\mathscr{R}(X_i)$;
IF there exists a linear relation $\mathscr{R}(X_i) = \sum_{\mathscr{R}(X_j)\in \text{Mbasis}} \sigma_j \mathscr{R}(X_j)$
THEN:
$g_i := X_i - \sum_{X_j \in \text{Mbasis}} \sigma_j X_j$
$G := [g_i, G]$;
LPP := $[X_i$, LPP];
ELSE:
Mbasis := $[[X_i, \mathscr{R}(X_i)]$, MBasis];
InsertNexts(X_i);
X_i := NextMonom;
END: G

There is no need to compute the entire Gröbner basis in our case. Since we are only interested in the G_κ, one could even limit the list PowerProduct to be only the list of monomials not containing any y_j by changing the definition of the command InsertNexts. Of course, when $q > 2$, one could compute the entire Gröbner basis and use the ideas presented in Section 3 to obtain the error evaluator polynomial as well. However, there are more efficient ways to compute

the error evaluator polynomial given the error locator polynomial (which we have computed once we have all the polynomials in the $\boldsymbol{x}$- and $\boldsymbol{z}$-variables in the Gröbner basis G).

The standard implementation of the map $\mathscr{R}(X_i)$ is via the multivariable division algorithm, and making use of the fact that the elements of `MBasis` are already partially reduced. An alternative way to compute the $\mathscr{R}(X_i)$ is presented in [11]. The vectors in `MBasis` can be quite large even for relatively small cyclic codes. For the code presented in Example 3.6, the vectors $\mathscr{R}(X_i)$ contain $2^{12} = 4096$ elements. It follows from the structure of F that the $\mathscr{R}(X_i)$'s will be relatively sparse, so techniques for computing with sparse matrices must be employed. We will present a number of improvements in the implementation of this algorithm in the next section for the case of binary cyclic codes.

Finally, we note that in [11] the authors estimate the complexity of their algorithm for finding a Gröbner basis with respect to $<_2$, given a Gröbner basis with respect to $<_1$, as $\mathscr{O}(vD^3)$, where v is the number of variables and D is the number of points in the variety (counting multiplicity). From (3), we see that $v = 2t + r$ and $D = (n+1)^t$. Since $2t + r < 2(n+1)$ we can approximate the complexity of this algorithm in terms of n and t as $\mathscr{O}((n+1)^{3t+1})$.

5 Computing G_κ over $\mathbb{F}_2$

There are several improvements one can make when applying the algorithm of the previous section to the class of binary cyclic codes. The most obvious simplification in the binary case is that there is no need to consider the $\boldsymbol{y}$-variables. From this observation several things follow.

The first is that the polynomial $(\sum_{i=1}^{t} z_i) + x_1$ is in the desired Gröbner basis G, and its leading term is z_t. This follows from the structure of the Gröbner basis for zero dimensional ideals computed with respect to the lexicographic term ordering (see e.g. [1, Corollary 2.2.11]). Therefore no polynomial in G, other than $(\sum_{i=1}^{t} z_i) + x_1$, contains power products divisible by z_t, and hence we can remove the variable z_t from the list of variables in the subroutine `InsertNexts`, giving us one less variable to work with.

The second consequence of the absence of $\boldsymbol{y}$-variables is that the basis for the vector space $\mathbb{F}_q[\boldsymbol{x}, \boldsymbol{z}]/I$ simplifies to:

$$\Psi = \left\{ z \mid z = \prod_{i=0}^{t} z_i^{b_i},\ 0 \leqq b_i \leqq n \right\}. \tag{10}$$

Ψ has $(n+1)^t$ elements and is obtained from the Gröbner basis F.

The structure of the Gröbner basis F also greatly simplifies the map $\mathscr{R}$. As pointed out in [11], every monomial in `PowerProducts` is either $x_j X$ or $z_j X$ where $1 \leqq j \leqq t$ (in our case we do not have $z_t X$), and X is a monomial that has been previously reduced with respect to F. First, we address the case of reducing monomials of the form $z_j X$. Let r_X be the polynomial corresponding to $\mathscr{R}(X)$. Then, to compute $\mathscr{R}(z_j X)$ one simply has to multiply each power product in r_X by z_j and replace any z_j^{n+1} by z_j. To compute the reduction of monomials of the form $x_j X$, one replaces x_j by $z_1^{i_j} + z_2^{i_j} + \ldots + z_t^{i_j}$, and one uses the fact that $\mathscr{R}$ is linear to obtain

$$\mathscr{R}(x_j X) = \mathscr{R}(z_1^{i_j} X) + \mathscr{R}(z_2^{i_j} X) + \ldots + \mathscr{R}(z_t^{i_j} X). \tag{11}$$

The computation of the right-hand side is done exactly as indicated above, reducing the corresponding powers of z_j that appear in the respective summands. Hence, the map $\mathscr{R}$ in the case of binary cyclic codes is quite simple.

Next, we address the problem of representing the vectors in MBasis. First, we note that to each element $X = \prod_{i=0}^{t} z_i^{b_i}$ of Ψ, we can associate the unique t-tuple $(b_1, b_2, \ldots, b_t) \in \mathbb{Z}_+^t$. Let S be the set of all t-tuples corresponding to elements of Ψ, and let $\mathscr{P}(S)$ be the power set of S. Then define the map χ by:

$$\begin{aligned} \chi : \operatorname{span}(\Psi) &\to \mathscr{P}(S) \\ f &\mapsto \{s \mid z^s \in f\}. \end{aligned} \tag{12}$$

Instead of storing binary vectors of length $(n+1)^t$ corresponding to $\mathscr{R}(X_i)$ in MBasis, we store the support set of t-tuples $\chi(\mathscr{R}(X_i))$. One can perform the vector addition $\mathscr{R}(X_i) + \mathscr{R}(X_j)$ using the χ-representation.

$$\chi(\mathscr{R}(X_i) + \mathscr{R}(X_j)) = [\chi(\mathscr{R}(X_i)) \cup \chi(\mathscr{R}(X_j))] \operatorname{minus} [\chi(\mathscr{R}(X_i)) \cap \chi(\mathscr{R}(X_j))]. \tag{13}$$

To check for linear relations in the WHILE loop of the algorithm, one can store a *row echelon* version of the vectors in MBasis. This requires keeping track of two additional sets. The first is the list of supports that correspond to the *pivots* of the row echelon form of MBasis, denote this set by Pivots. The second is a set that keeps track of which vectors in MBasis have non-empty support at position j, where j is not a pivot. We call this set MSupport. Then, after each computation of $\mathscr{R}(X_i)$, to check for linear relation with the preceding elements of MBasis we proceed as follows: we intersect $\chi(\mathscr{R}(X_i))$ with Pivots, we "add" the supports of the vectors in MBasis corresponding to $\chi(\mathscr{R}(X_i)) \cap$ Pivots to $\chi(\mathscr{R}(X_i))$, and we check if it is the empty set. If so, we have a new element in the Gröbner basis G, namely the sum of the corresponding X_i's. If it is not empty, we obtain one more element in the basis MBasis, we then choose one element from the remaining supports, add this to the set Pivots, and row reduce MBasis using the set MSupport, making sure to update MSupport as we go along.

To compute $\mathscr{R}(z_j^b X)$ using the supports is also straight forward. To do this one simply adds b to the jth element of each member of $\chi(\mathscr{R}(X))$ making sure to identify $n+1$ to 1 as necessary. Using this and (11), the computation of $\mathscr{R}(X)$ for any X in PowerProducts becomes relatively simple. As an alternative to storing t-tuples corresponding to the elements of Ψ, one could store integers corresponding to an ordering of Ψ and work with sets of integers instead of t-tuples, adjusting the map $\mathscr{R}$ accordingly.

6 Decoding the [23, 12, 7] Golay Code

Using the algorithm presented in Section 4 and the improvements discussed in Section 5 we now present a method for the decoding of the binary [23, 12, 7] Golay Code. The following Gröbner basis was computed using a program written in the MAPLE programming language.

Recall (see e.g. [19, Chapter 7.6]) that the non-primitive BCH code of design distance 3 and length 23 is equivalent to the [23, 12, 7] Golay Code. Hence, from (3) we obtain

$$F = \{x_1 + z_1 + z_2 + z_3,\ z_1^{24} + z_1,\ z_2^{24} + z_2,\ z_3^{24} + z_3\}.$$

The G_κ's obtained from the Gröbner Basis for F with respect to $<_2$ are as follows:

$$G_0 = \{x_1^{2048} + x_1\}$$

$$\begin{aligned} G_1 = G_0 \cup \{ & z_1 + z_1^{24}, \\ & z_1^3 x_1^{24} + z_1^3 x_1 + z_1^2 x_1^{25} + z_1^2 x_1^2 + z_1 x_1^{1337} + z_1 x_1^{1291} + z_1 x_1^{1176} \\ & + z_1 x_1^{1153} + z_1 x_1^{1061} + z_1 x_1^{1038} + z_1 x_1^{808} + z_1 x_1^{785} + z_1 x_1^{555} \\ & + z_1 x_1^{532} + z_1 x_1^{440} + z_1 x_1^{394} + z_1 x_1^{348} + z_1 x_1^{325} + z_1 x_1^{187} \\ & + z_1 x_1^{164} + z_1 x_1^{95} + z_1 x_1^{26} + x_1^{1338} + x_1^{1292} + x_1^{1177} + x_1^{1154} + x_1^{1062} \\ & + x_1^{1039} + x_1^{809} + x_1^{786} + x_1^{556} + x_1^{533} + x_1^{441} + x_1^{395} + x_1^{349} + x_1^{326} \\ & + x_1^{280} + x_1^{257} + x_1^{188} + x_1^{165} + x_1^{96} + x_1^4 \} \end{aligned}$$

$$\begin{aligned} G_2 = G_1 \cup \{ & z_2^{24} + z_2, z_2^2 z_1 + z_2^2 x_1 + z_2 z_1^2 + z_2 x_1^2 + z_1^2 x_1 + z_1 x_1^2 + x_1^{256} + x_1^3, \\ & z_2^3 x_1^{24} + z_2^2 x_1 + z_2 z_1 x_1^{24} + z_2 z_1 x_1 + z_2 x_1^{25} + z_2 x_1^2 + z_1^2 x_1^{24} + z_1^2 x_1 \\ & + z_1 x_1^{25} + z_1 x_1^2 + x_1^{1337} + x_1^{1291} + x_1^{1176} + x_1^{1153} + x_1^{1061} + x_1^{1038} \\ & + x_1^{808} + x_1^{785} + x_1^{555} + x_1^{532} + x_1^{440} + x_1^{394} + x_1^{348} + x_1^{325} + x_1^{187} \\ & + x_1^{164} + x_1^{95} + x_1^{26} \} \end{aligned}$$

$$G_3 = G_2 \cup \{z_3 + z_2 + z_1 + x_1\}$$

Let $\boldsymbol{s}$ be the syndrome corresponding to a received vector with 3 or fewer errors. According to Theorem 3.3 and to Corollary 3.4, if exactly one or 3 errors occur in $\boldsymbol{s}$, then G_1 and G_3 will give rise to the error locator, and it is clear which polynomials these will be. However, if exactly two errors occur in $\boldsymbol{s}$ then one of the two non-trivial polynomials listed in G_2 will give rise to the error locator, these are:

$$z_2^2 z_1 + z_2^2 x_1 + z_2 z_1^2 + z_2 x_1^2 + z_1^2 x_1 + z_1 x_1^2 + x_1^{256} + x_1^3,$$

and

$$\begin{aligned} & z_2^2 x_1^{24} + z_2^2 x_1 + z_2 z_1 x_1^{24} + z_2 z_1 x_1 + z_2 x_1^{25} + z_2 x_1^2 + z_1^2 x_1^{24} \\ & + z_1^2 x_1 + z_1 x_1^{25} + z_1 x_1^2 + x_1^{1337} + x_1^{1291} + x_1^{1176} + x_1^{1153} + x_1^{1061} \\ & + x_1^{1038} + x_1^{808} + x_1^{785} + x_1^{555} + x_1^{532} + x_1^{440} + x_1^{394} \\ & + x_1^{348} + x_1^{325} + x_1^{187} + x_1^{164} + x_1^{95} + x_1^{26}. \end{aligned}$$

Since, in this case, an error does occur, we must have $x_1 \neq 0$. If we evaluate the first polynomial at the point $(\boldsymbol{s}, 0, z)$ we get a degree 2 polynomial with leading term $\boldsymbol{s}z^2$, and hence this is the error locator polynomial for the received word corresponding to $\boldsymbol{s}$. Therefore we only need to consider the first polynomial when checking for two errors. With this in mind, let:

$$\begin{aligned} g_1 = & z_1^3 x_1^{24} + z_1^3 x_1 + z_1^2 x_1^{25} + z_1^2 x_1^2 + z_1 x_1^{1337} + z_1 x_1^{1291} + z_1 x_1^{1176} \\ & + z_1 x_1^{1153} + z_1 x_1^{1061} + z_1 x_1^{1038} + z_1 x_1^{808} + z_1 x_1^{785} + z_1 x_1^{555} \\ & + z_1 x_1^{532} + z_1 x_1^{440} + z_1 x_1^{394} + z_1 x_1^{348} + z_1 x_1^{325} + z_1 x_1^{187} \end{aligned}$$

$$+ z_1x_1^{164} + z_1x_1^{95} + z_1x_1^{26} + x_1^{1338} + x_1^{1292} + x_1^{1177} + x_1^{1154} + x_1^{1062}$$
$$+ x_1^{1039} + x_1^{809} + x_1^{786} + x_1^{556} + x_1^{533} + x_1^{441} + x_1^{395} + x_1^{349} + x_1^{326}$$
$$+ x_1^{280} + x_1^{257} + x_1^{188} + x_1^{165} + x_1^{96} + x_1^4$$

$$g_2 = z_2^2z_1 + z_2^2x_1 + z_2z_1^2 + z_2x_1^2 + z_1^2x_1 + z_1x_1^2 + x_1^{256} + x_1^3$$

$$g_3 = z_3 + z_2 + z_1 + x_1.$$

Now we provide an algorithm for decoding the [23, 12, 7] Golay code which uses only polynomial evaluation. Suppose $\tilde{\boldsymbol{c}}$ is our received message. To find the error locator polynomial we proceed as follows.

Step 1: Compute the syndrome $\boldsymbol{s} = H\tilde{\boldsymbol{c}}$. If $\boldsymbol{s} = 0$ then stop.
Step 2: Compute $g_1(\boldsymbol{s}, 0)$, if $g_1(\boldsymbol{s}, 0) = 0$ then go to Step 3; Otherwise, the error locator polynomial is $g_1(\boldsymbol{s}, z)$, and stop.
Step 3: Compute $g_2(\boldsymbol{s}, 0, 0)$, if $g_2(\boldsymbol{s}, 0, 0) = 0$ then go to Step 4; Otherwise, the error locator polynomial is $g_2(\boldsymbol{s}, 0, z)$, and stop.
Step 4: The error locator polynomial is $g_3(\boldsymbol{s}, 0, 0, z)$, stop.

Note that in the above example the design distance of the code is 3, however the actual distance is 7. The algorithm presented is able to correct up to three errors, hence up to the true minimum distance of the code.

References

1. Adams, W. W., Loustaunau, P.: An Introduction to Gröbner Bases, vol. 3. Graduate Studies in Mathematics. Rhode Island: American Mathematical Society 1994
2. Berlekamp E. R.: Algebraic Coding Theory. New York: McGraw-Hill 1968.
3. Blackburn, S. R., Chambers, W. G.: Some Remarks on an Algorithm of Fitzpatrick. IEEE Trans. Inform. Theory, IT- 42(4), 1269–1271 (1996)
4. Caniglia L., Galligo A., Heintz, J.: Some New Effectivity Bounds in Computational Geometry. In: Proceedings of AAECC-6, Lecture Notes in Computer Sciences, vol. 357, pp. 131–151. Berlin Heidelberg New York: Springer 1988
5. Chen X., Reed, I. S., Helleseth, T., Truong, K.: Algebraic Decoding of Cyclic Codes: A Polynomial Ideal Point of View. In: Mullen, G.L., Jau-Shyong Shiue, P., (eds.) Finite Fields. Theory, Applications, and Algorithms, vol. 168. Contemporary Mathematics, pp. 15–22. Rhode Island: American Mathematical Society 1994
6. Chen X., Reed I.S., Helleseth, T., Truong, K.: Use of Gröbner Bases to Decode Binary Cyclic Codes up to the True Minimum Distance. IEEE Trans. Inform. Theory, **40** 1654–1661 (1994)
7. Cooper III., A. B.: Direct solution of BCH decoding equations. Communication, Control and Signal Processing, pages 281–286, 1990
8. Cooper III., A. B.: Finding BCH Error Locator Polynomials in One Step. Electronic Letters **27**, 2090–2091 (1991)
9. Cox, D., Little, J., O'Shea, D. O.: Ideals, Varieties and Algorithms. Undergraduate Text in Mathematics. Berlin Heildelberg New York: Springer 1992
10. de Boer M., Pellikaan, R.: Gröbner Bases for Error-Correcting Codes and Their Decoding. Submitted to the book 'Some Tapas of Computer Algebra' (1996)
11. Faugere, J. C., Gianni, P., Lazard, D., Mora, T.: Efficient Computation of Zero-Dimensional Gröbner Bases by a Change of Ordering. J. Symb. Comput. **16**, 329–344 (1993)
12. Feng, G. L., Tzeng, K. K.: Decoding Cyclic and BCH Codes up to Actual Minimum Distance Using Nonrecurrent Syndrome Dependence Relations. IEEE Trans. Inform. Theory, IT-**37** (6) 1716–1723 (1991)

13. Feng, G. L., Tzeng, K. K.: A Generalized Euclidean Algorithm for Multisequence Shift-Register Synthesis with Appl. to Decoding Cyclic Codes. IEEE Trans. Inform. Theory, IT-**37** (5), 1274–1287 (1991)
14. Fitzgerald, J.: Applications of Gröbner Bases to Linear Codes. PhD thesis, Louisiana State University (1996)
15. Fitzgerald, J., Lax, R. F.: Decoding Affine Variety Codes Using Gröbner Bases. Preprint (1996)
16. Fitzpatrick, P.: On the Key Equation. IEEE Trans. Inform. Theory, IT-**41**(5), 1290–1302 (1995)
17. Gianni, P.: Properties of Gröbner Bases under Specializations. In: EUROCAL'87, Lecture Notes in Computer Sciences, vol. **378**, 293–297. Berlin Heidelberg New York: Springer 1989
18. Kalkbrenner, M.: Solving Systems of Algebraic Equations Using Gröbner Bases. In: EUROCAL'87, Lecture Notes in Computer Sciences, vol. **378**, 282–292. Berlin Heidelberg New York: Springer 1989
19. MacWilliams, F. J., Sloane, N. J. A.: The Theory of Error-Correcting Codes. Amsterdam: North Holland 1977
20. Massey, J. L.: Shift-Register Synthesis and BCH Decoding. IEEE Trans. Inform. Theory, IT-**15** 122–127 (1969)
21. Peterson, W. W.: Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes. IEEE Trans. Inform. Theory IT-**6** 459–470 (1960)
22. Poli, A., Huguet, L.: Error Correcting Codes: Theory and Applications. Paris: Masson and Prentice Hall 1992