AAECC **13**, 137–162 (2002)

AAECC
Applicable Algebra in Engineering, Communication and Computing
© Springer-Verlag 2002

Groebner Bases and Distance of Cyclic Codes

Massimiliano Sala

Department of Mathematics, University of Pisa, Italy
(e-mail: sala@mail.dm.unipi.it)

Received: April 24, 2001; revised version: February 14, 2002

Abstract. Recently some methods have been proposed to find the distance of cyclic codes using Gröbner bases. We present a similar method, whose computational cost is significantly lower.

Keywords: Gröbner bases, Cyclic codes, Distance.

1 Introduction

Extensive work has been done to find (or estimate) the distance of cyclic codes, particularly in the binary case and for lengths of the kind $n = 2^m - 1$. See for example [2], [5], [6], [15], [18], [20], [26], [28], [31] and [32]. Furthermore, the dual distance of BCH codes plays an important role both to estimate their covering radius (see [19]) and to estimate their weight distribution (see [17] and [30]).

Recently a method has been proposed by Augot, which uses Gröbner bases (see [1]). Unfortunately, the polynomials involved have too many indeterminates to obtain interesting results with a reasonable computational cost. That is, the number of indeterminates is greater than the length of the code.

There are some ways to correct errors using Gröbner bases (see for example [9],[10] and [13]). Also the decoding procedure proposed by Loustaunau and York (see [21]) relies on Gröbner bases, but it is remarkable because it uses polynomials with a few indeterminates (no more than the distance of the code). Their work is based on an earlier paper by Chen, Reed, Helleseth and Truong (see [7]) and has been improved by Caboara and Mora in [4].

We follow the approach of [21], adapting it to find the distance of the code. This way we exhibit an original algorithm to find the distance of cyclic codes via the computation of some Gröbner bases, starting from polynomials with less indeterminates. Focusing on primitive narrow-sense BCH codes, we can exploit the particular structure of the polynomials involved, finding the distance of some of such codes also for large dimensions.

We exhibit also some variations to our method which are faster in some interesting cases.

2 Preliminaries

Now we fix some notation and recall some known facts.

Let C be an $[n, k, d]$ code on a field $\mathbb{F}_q$ such that q is prime to n. The words of C can be represented as $\mathbb{F}_q$-polynomials of degree lower than n. If C is cyclic there exists a **generator polynomial** g of the code C, that is g is a polynomial of degree $r = n - k$ such that the words of C are exactly the polynomials vanishing at the roots of g modulo $x^n - 1$. Let α be a primitive n-root of unity in the splitting field $\mathbb{F}_Q$ of $x^n - 1$ over $\mathbb{F}_q$, that is

$$(x^n - 1) = \prod_{i=0}^{n-1}(x - \alpha^i)$$

(the condition $(n, q) = 1$ assures that there are n distinct n-th roots of unity, corresponding to some powers of α).

A **defining set** $S_C = \{h_1, \ldots, h_v\}$ of C is a set of powers of α such that g is the minimal degree $\mathbb{F}_q$-polynomial vanishing on them. If for C one can choose $S_C = \{1, 2, \ldots, \delta - 1\}$, we say that C has **designed distance** δ and we denote it by **Cyclic**$(n, \delta, \mathbb{F}_q)$. If it happens that $d = \delta$, then we say that C has **coincident distances**.

If n is of the form $n = q^m - 1$ (so that α is a primitive element of the splitting field $\mathbb{F}_{q^m}$), we say that C is a **narrow-sense primitive BCH code** and we denote it by **BCH**$(n, \delta, \mathbb{F}_q)$.

We are mainly interested in the binary case, i.e. when $\mathbb{F}_q = \mathbb{Z}_2$. In this case, we denote by **Cyclic**(n, δ) the binary cyclic code Cyclic$(n, \delta, \mathbb{Z}_2)$ (with designed distance δ) and we denote by **BCH**(n, δ) the binary narrow-sense primitive BCH code **BCH**$(n, \delta, \mathbb{F}_q)$ (with designed distance δ and $n = 2^m - 1$). For the latter we use also the standard notation **BCH**$[n, k, d]$.

Remark 2.1. For any n, the codes $\{\text{Cyclic}(n, \delta, \mathbb{F}_q)\}$ are BCH codes, which are not primitive if n is not of the kind $q^m - 1$.

We can define the **Discrete Fourier Transform** (or **DFT** for short) of a code word $c = (c_0, \ldots, c_{n-1}) \in C$ as the vector:

$$\phi(c) = \text{DFT}(c) = (A_0, \ldots, A_{n-1}), \quad A_i = \sum_{j=0}^{n-1} c_j \alpha^{ij}.$$

Obviously $\phi(c) = (A_0, \dots, A_{n-1}) \in (\mathbb{F}_Q)^n$ (in the case of a $BCH[2^m - 1, k, d]$ code we have $\mathbb{F}_Q = \mathbb{F}_{2^m}$).

The codes with a spectral definition form a large class of codes, which contains in particular all cyclic codes (see [1]):

Definition 2.1. *Let C be a code in $(\mathbb{F}_Q)^n$ (or $(\mathbb{F}_q)^n$). If there exist l multivariate polynomials $P_1, \dots, P_l$ in n variables such that for all $c \in (\mathbb{F}_Q)^n$ (or for all $c \in (\mathbb{F}_q)^n$) c belongs to C if and only if*

$$P_1(A_0, \dots, A_{n-1}) = \cdots = P_l(A_0, \dots, A_{n-1}) = 0$$

where as before $\phi(c) = (A_0, \dots, A_{n-1})$, then the code C is called **a code with a spectral definition** *and the polynomials $P_1, \dots, P_l$ define the* **code spectral equations**.

Let $c = (c_0, \dots, c_{n-1}) \in (\mathbb{F}_Q)^n$, w be its weight and $i_1, \dots, i_w$ be the positions of its nonzero components. For any $h \in \{1, \dots, w\}$ we define:

- the **h-th locators** of c as $\mathbf{X_h} = \alpha^{i_h}$
- the **h-th elementary symmetric function** of C as

$$\sigma_h = \quad (-1)^h \sum_{1 \le k_1 \le \cdots \le k_h \le w} X_{k_1} \dots X_{k_h}$$

The elementary symmetric functions of c and the DFT of c are linearly related (see [1]).

Augot collected these linear relations in a system $Sys(C)(w)$ together with the spectral equations defining the code C and stated the main result of his paper [1]:

Theorem 2.1. *Let C be a cyclic $\mathbb{F}_q$ $[n, k, d]$ code defined by the spectral equations $P_1 = \cdots = P_l = 0$ and such that q is prime to n. For any $1 \le w \le n - 1$ there exists a solution of $Sys(C)(w)$ if and only if there exists a word of C of weight w.*

Corollary 2.1. *Let C be a cyclic $\mathbb{F}_q$ $[n, k, d]$ code (q prime to n) defined by the spectral equations $P_1 = \cdots = P_l = 0$. If $Sys(C)(w)$ has solutions and $Sys(C)(w')$ has not for any $1 \le w' \le w - 1$, then $\mathbf{d} = \mathbf{w}$.*

Apart from some very simple cases, it is quite difficult to find the solutions of the systems $Sys(C)(w)$. The most efficient way is often to compute a Gröbner basis G for $Sys(C)(w)$ with the lexicographic order (see [11] and [12]). From G it is immediate to see if there are solutions or not, but the computation of G is quite expensive, as in $Sys(C)(w)$ there are many indeterminates (actually $w + n$) and the complexity of the computation is exponential in the number of indeterminates.

3 The Method by Loustaunau and York

Let C be an $[n, k, d]$ cyclic code on a field $\mathbb{F}_q$ such that q is prime to n. Let g be its generator polynomial and $\alpha^{i_1}, \dots, \alpha^{i_{n-k}}$ be its roots. Then a parity check matrix for C is (see [27], p. 214):

$$H = \begin{pmatrix} 1 & \alpha^{i_1} & \alpha^{2i_1} & \dots & \alpha^{(n-1)i_1} \\ 1 & \alpha^{i_2} & \alpha^{2i_2} & \dots & \alpha^{(n-1)i_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{i_{n-k}} & \alpha^{2i_{n-k}} & \dots & \alpha^{(n-1)i_{n-k}} \end{pmatrix}$$

Let $c \in (\mathbb{F}_q)^n$ be a code word and let $\tilde{c}$ be the received message, then $\tilde{c} = c + e$, where $e = (e_1, \dots, e_{n-k})$ is the error vector.

The syndrome equations are:

$$(3.1) \quad e_0 + e_1\alpha^{i_j} + \cdots + e_{n-1}\alpha^{(n-1)i_j} = s_j, \quad j = 1, \dots, r = n-k\ ,$$

where $s = (s_1, \dots, s_r) = Hc = H\tilde{c}$ is the syndrome vector.

We recall that, if there are no more than t errors with $d = 2t+1$, then we can correct them just solving the syndrome equations.

Now we present the algorithm proposed by Loustaunau and York to solve these equations. The idea (see Remark 3.1) is to express the solutions of (1) as points in an algebraic variety. Consider the variables

$$\{x_j, z_i, y_i \mid 1 \le j \le n-k,\ 1 \le i \le t\}$$

and the following system:

$$(I) \begin{cases} y_1 z_1^{i_j} + y_2 z_2^{i_j} + \cdots + y_t z_t^{i_j} - x_j = 0, & j = 1, \dots, r \\ z_i^{n+1} - z_i = 0, & i = 1, \dots, t \\ y_i^{q-1} - 1 = 0, & i = 1, \dots, t \end{cases}$$

Then it can be shown (see [21]) that the solutions are of the form

$$(s_1, \dots, s_{n-k}, 0, \dots, 0, \alpha^{l_1}, \dots, \alpha^{l_\tau}, *, \dots, *, \beta_1, \dots, \beta_\tau)$$

where τ is the number of the errors (so $\tau \le t$), the first $n-k$ coordinates represent the x_j, the following t coordinates represent the z_i and the last t coordinates represent the y_i.

The location of the errors is given by the powers $l_1, \dots, l_\tau$ and their value is given by $\beta_1, \dots, \beta_\tau$. A $*$ indicates that this coordinate can be any nonzero element of $\mathbb{F}_q$ and there are $t-\tau$ such y-coordinates corresponding to the $t-\tau$ zero z-coordinates.

So it is possible to recover from errors just solving the system (I) via the computation of a Gröbner basis with the lexicographic order.

Remark 3.1. System (I) first appeared in [8] by X. Chen, I.S. Reed, T. Helleseth and T.K. Truong, where a similar method was proposed (the variety described by system (I) is called the **syndrome variety** or, more exactly, the **CRHT variety**). A modification of their algorithm has lead to the decoding algorithm by Caboara and Mora (see [4]), which exploits the structure of the Gröbner basis of system (I) as it is described by the Gianni-Kalkbrenner Theorem (see [14] and [16]).

4 Exploiting the CRHT Variety for the Distance

If we write system (I) in the case of a null syndrome $s = \mathbf{0}$, we obtain a simplified version:

$$\begin{cases} y_1 z_1^{i_1} + y_2 z_2^{i_1} + \cdots + y_t z_t^{i_1} & = 0 \\ \ldots & = 0 \\ y_1 z_1^{i_{n-k}} + y_2 z_2^{i_{n-k}} + \cdots + y_t z_t^{i_{n-k}} & = 0 \\ z_1^{n+1} - z_1 & = 0 \\ \ldots & = 0 \\ z_t^{n+1} - z_t & = 0 \\ y_1^{q-1} - 1 & = 0 \\ \ldots & = 0 \\ y_t^{q-1} - 1 & = 0 \end{cases}$$

For any cyclic code C we can consider a defining set $S_C = \{h_1, \ldots, h_v\}$. So we can denote by $\mathbf{J_C(t)}$ a reduced version of the previous system:

$$J_C(t) \begin{cases} y_1 z_1^{h_1} + y_2 z_2^{h_1} + \cdots + y_t z_t^{h_1} & = 0 \\ \ldots & = 0 \\ y_1 z_1^{h_v} + y_2 z_2^{h_v} + \cdots + y_t z_t^{h_v} & = 0 \\ z_1^{n+1} - z_1 & = 0 \\ \ldots & = 0 \\ z_t^{n+1} - z_t & = 0 \\ y_1^{q-1} - 1 & = 0 \\ \ldots & = 0 \\ y_t^{q-1} - 1 & = 0 \end{cases}$$

As we supposed a zero syndrome, there are no errors to correct. Anyway we can suppose to have sent the word $\mathbf{0}$ and to have received a word of the code of weight w, so that the syndrome is **zero** and the system becomes $J_C(w)$. We denote by $\mathbf{ns}(J_C(w))$ the number of solutions of $J_C(w)$.

Example 4.1. Let D be the $\mathbb{Z}_3$ primitive BCH code of designed distance 7 and length $n = 3^m - 1$ (with $m \geq 2$), i.e. $D = \text{Cyclic}(n, 7, \mathbb{Z}_3)$. Then, as defining set we can take $S_D = \{1, 2, 4, 5\}$ and its associated system $J_D(t)$ is:

$$J_D(t)\begin{cases} y_1z_1 + y_2z_2 + \cdots + y_tz_t &= 0 \\ y_1z_1^2 + y_2z_2^2 + \cdots + y_tz_t^2 &= 0 \\ y_1z_1^4 + y_2z_2^4 + \cdots + y_tz_t^4 &= 0 \\ y_1z_1^5 + y_2z_2^5 + \cdots + y_tz_t^5 &= 0 \\ z_1^{n+1} - z_1 &= 0 \\ \ldots &= 0 \\ z_t^{n+1} - z_t &= 0 \\ y_1^2 - 1 &= 0 \\ \ldots &= 0 \\ y_t^2 - 1 &= 0 \end{cases}$$

There are some solutions of system $J_C(w)$ which are important in our framework:

Definition 4.1. *A solution of the system $J_C(w)$ is called* **spurious** *if it does not correspond to a code word.*

From previous definition we immediately get:

Proposition 4.1. *Let C be an $\mathbb{F}_q$ $[n, k, d]$ cyclic code with $(n, q) = 1$. There is at least one solution of $J_C(w)$ which is not spurious if and only if there is at least one word of weight w in the code.*

Now we present an original method to find the minimal distance of an arbitrary cyclic code:

Proposition 4.2. *Let C be an $\mathbb{F}_q$ $[n, k, d]$ cyclic code with $(n, q) = 1$. If we know that $d \geq \delta$, we can proceed this way:*

- *set $w = \delta$ and construct $J_C(w)$;*
- *count both the number of the spurious solutions of $J_C(w)$ and the number of all its solutions;*
- *if all solutions are spurious, then increase w to $w + 1$, construct $J_C(w)$ and come back to previous step;*
- *if on the contrary there are non-spurious solutions, then return w.*

The value returned by this algorithm is the distance d of the code.

Proof. Until only spurious solutions are found, there are no code words of weight w. As soon as any non-spurious solutions are found, then they correspond to words of minimal weight w. □

So the problem of finding the distance of an arbitrary cyclic code reduces to the problem of counting the solutions of $J_C(w)$, both the spurious ones and the non-spurious ones:

- the number of spurious solution can be found via combinatoric arguments (and in Section 5 we show how the spurious solutions can be easily calculated in the binary case);
- the number of all solutions can be obtained starting from a Gröbner basis of the ideal generated by the polynomials of the system (see [3]).

The disadvantage of our method is that we need a Gröbner basis, whose computation is expensive. Anyway our system $J_C(w)$ has no more than $2d$ indeterminates (and only d in the binary case), which are significantly less than the indeterminates of the system $Sys(C)(w)$ previously proposed in [1], which has $n + d$ indeterminates.

Remark 4.1. In the binary case the system $J_C(t)$ needs the variables $\{y_i\}$ no more, so it can be written in the simpler form:

$$J_C(t)\begin{cases} z_1^{h_1} + z_2^{h_1} + \cdots + z_t^{h_1} &= 0 \\ \ldots &= 0 \\ z_1^{h_v} + z_2^{h_v} + \cdots + z_t^{h_v} &= 0 \\ z_1^{n+1} - z_1 &= 0 \\ \ldots &= 0 \\ z_t^{n+1} - z_t &= 0 \end{cases}$$

In the binary case we can precisely characterize all spurious solutions of system $J_C(t)$:

Lemma 4.1. *Let $\{z_1, \ldots, z_t\}$ be a spurious solution of system $J_C(t)$ of Remark 4.1 Then either there are h, l such that $z_h = z_l$ or there is an h such that $z_h = 0$.*

Proof. It is obvious as both the condition $z_h = 0$ and the condition $z_h = z_l$ are nonsense for a code word. □

In Section 6 we make an explicit computation for a simple non-binary case.

5 Spurious Solutions in the Binary Case

In this section we always suppose that C is a binary cyclic code such that n is odd and $1 \in S_C$, i.e. among the roots of the generator polynomial of C there is a primitive n-root of unity α of the splitting field of $x^n - 1$ over $\mathbb{Z}_2$. An important class of such codes is the class of the binary primitive codes $BCH[n, k, d]$ (see [22], p. 257–268).

Remark 5.1. Let C be an arbitrary binary $[n, k, d]$ cyclic code such that n is odd and let g be its generator polynomial. As g divides $x^n - 1$, we can consider the cyclic code generated by $h = (x^n - 1)/g$. It is sometimes called the **dual** of C (see [27], p. 208) and we denote it by $\mathbf{C}^{\perp}$. Note that $C^{\perp}$ is not the dual code of C, which has a different generator, but nonetheless they are equivalent codes, so their distance is the same.

We state that either C satisfies our hypothesis $1 \in S_C$ or $C^{\perp}$ does. In fact both h and g divide $x^n - 1$. But $(x - \alpha)$ is an irreducible factor of $x^n - 1$, so that either $(x - \alpha)$ divides h or $(x - \alpha)$ divides g.

In conclusion, for any odd n one half of the binary cyclic codes (with length n) satisfies our hypothesis $1 \in S_C$.

Remark 5.2. The hypothesis $1 \in S_C$ implies that equation

$$(*) \qquad z_1 + z_2 + \cdots + z_w = 0$$

is in the system $J_C(w)$ for any w.

Definition 5.1. *Let C be a binary $[n, k, d]$ cyclic code and let w be an integer such that $1 \le w \le n - 1$. We denote by*

- **ns** $[w, n]$ *the number of all solutions of $J_C(w)$;*
- **spz**$[w, n]$ *the number of the spurious solutions of $J_C(w)$ which have at least one zero coordinate;*
- **spd**$[w, n]$ *the number of the spurious solutions of $J_C(w)$ which have no zero coordinates;*
- **sp** $[w, n]$ *the number of all spurious solutions of $J_C(w)$.*

It is clear that $\mathrm{sp}[w, n] = \mathrm{spz}[w, n] + \mathrm{spd}[w, n]$ for any w and for any n. For later purposes we set

$$\mathrm{ns}[0, n] = \mathrm{sp}[0, n] = \mathrm{spd}[0, n] = 1, \qquad \mathrm{spz}[0, n] = 0$$

From now on, we implicitly use the characterization of spurious solutions given by Lemma 4.1.

Lemma 5.1. *Let C be a binary $[n, k, d]$ cyclic code such that n is odd and $1 \in S_C$. Then:*

$$\mathrm{spz}[1, n] = 1, \qquad \mathrm{spd}[1, n] = 0, \qquad \mathrm{ns}[1, n] = \mathrm{sp}[1, n] = 1$$

$$\mathrm{spz}[2, n] = 1, \qquad \mathrm{spd}[2, n] = n, \qquad \mathrm{ns}[2, n] = \mathrm{sp}[2, n] = n + 1$$

Proof. In the case $w = 1$ equation $(*)$ reduces to $z_1 = 0$, which clearly satisfies also the other equations.

In the case $w = 2$ equation $(*)$ reduces to $z_1 = z_2$, which has as solutions the couples $\{(z, z) \mid z \in \mathbb{F}_{2^m}\}$, where $\mathbb{F}_{2^m}$ is the splitting field of $x^n - 1$. These couples satisfy the equations of the system $z^i = z^i$ for any i. The equations $z^{n+1} = z$ are satisfied if either z is an n-root of unity or $z = 0$, so the solutions are the couples

$$\{(z, z) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} \mid z^n = 1 \text{ or } z = 0\}.$$

The couple $(0, 0)$ has a zero component and no other couple has, then $\mathrm{spz}[2, n] = 1$ and $\mathrm{ns}[2, n] = \mathrm{sp}[2, n] = n + 1$, so that $\mathrm{spd}[2, n] = n$. $\square$

To compute the number of spurious solutions of $S_C(w)$, it is easier to compute separately the two values $\mathrm{spz}[w, n]$ and $\mathrm{spd}[w, n]$ and then sum them to get $\mathrm{sp}[w, n]$.

We need a recursive definition:

Definition 5.2. *Let $v = (v_0, \ldots, v_{w-1})$ be a w-vector with w even. We say that v has* **double coordinates** *if:*

- *each component of v is nonzero;*
- *there exist $i \neq j$ such that $v_i = v_j$;*
- *the truncated $(w - 2)$-vector obtained by v truncating the two coordinates $\{i, j\}$ has again double coordinates;*
- *in the case $w = 2$ there holds $v_0 = v_1$.*

Example 5.1. Let $v = (\alpha^2, 1, \alpha, 1)$, $v' = (\alpha, 1, \alpha, 1)$ and $v'' = (1, 0, 1, 0)$ be three 4-vectors in $\mathbb{F}_{2^m}$ for some m.

Examining v, we observe that $v_1 = v_3 = 1$, so that $i = 1$ and $j = 3$ with the notation of previous definition. Truncating these coordinates we obtain the 2-vector (α^2, α), which has not double coordinates. This means by definition that v itself has not double coordinates.

Examining v' we observe that $v'_1 = v'_3 = 1$, so that $i = 1$ and $j = 3$. Truncating these coordinates we obtain the 2-vector (α, α) which has double coordinates. So by definition v' has double coordinates.

The vector v'' has not double coordinates, because it has a zero component.

Lemma 5.2. *Let C be a binary $[n, k, d]$ cyclic code such that $1 \in S_C$ and n is odd. Let $w \geq 1$ be such that for any $w' \leq w - 1$ the system $J_C(w')$ has only spurious solutions. Then*

- *if w is even, any spurious solution with no zero components has double coordinates;*
- *if w is odd,every spurious solution has at least one zero component.*

Proof. By induction on w starting from 2 for the even case and from 1 for the odd case.

Let $w = 2$, then from the proof of Lemma 5.1, the spurious solutions with no zero components are the couples $\{(z, z) \mid z \in \mathbb{F}_{2^m}, z^n = 1\}$, which have obviously double coordinates.

Suppose now that $w \geq 4$ is even. If Z is a spurious solution of $J_C(w)$ which has no zero components, then (see Lemma 4.1) there are h, l such that $Z_h = Z_l$ and $l > h$. Let $\tilde{Z}$ be the $(w-2)$-vector obtained by Z truncating the coordinates h and l. We state that $\tilde{Z}$ is a solution of $J_C(w-2)$. Consider a generic equation of $J_C(w-2)$, say $z_1^i + \cdots + z_{w-2}^i = 0$, then we can take the corresponding equation of $J_C(w)$ $\quad z_1^i + \cdots + z_w^i = 0$, which is satisfied by Z, and use the fact that $Z_h = Z_l$ to get

$$Z_1^i + \cdots + Z_{h-1}^i + Z_{h+1}^i + \cdots + Z_{l-1}^i + Z_{l+1}^i + \cdots + Z_w^i = Z_l + Z_h = 0$$

Renaming $\tilde{Z}_s$ the remaining Z_s, we obtain $\tilde{Z}_1^i + \cdots + \tilde{Z}_{w-2}^i = 0$, which is the thesis.

Let $w = 1$, then there is only one spurious solution $z = 0$, which has a zero component (see Lemma 5.1).

We are left with the case w odd and $w \geq 3$. Suppose there is a spurious solution Z of $S_C(w)$ with no zero coordinates, then reasoning in a manner similar to the even case, we obtain that its truncated $(w-2)$-vector $\tilde{Z}$ is a spurious solution of the system $J_C(w-2)$.

As Z has no zero components, also $\tilde{Z}$ has no zero components. We know by induction that there are no spurious solutions of $J_C(w-2)$ with no zero components, leading to a contradiction. So all spurious solutions of $S_C(w)$ have at least one zero component. □

An immediate consequence of the previous lemma is the following corollary:

Corollary 5.1. *Let C be a binary $[n, k, d]$ cyclic code such that $1 \in S_C$ and n is odd. Let $w \geq 1$ be such that for any $w' \leq w - 1$ the system $J_C(w')$ has only spurious solutions. If w is odd then*

$$\mathrm{sp}[w, n] = \mathrm{spz}[w, n]$$

There exists an explicit formula for the values $\mathrm{spz}[w, n]$:

Lemma 5.3. *Let C be a binary $[n, k, d]$ cyclic code such that $1 \in S_C$ and n is odd. Let w be an integer such that $2 \leq w \leq n - 1$. Then*

$$(**) \qquad \mathrm{spz}[w, n] = \sum_{i=1}^{w} (-1)^{i+1} \binom{w}{i} \mathrm{ns}[w - i, n]$$

Proof. Let Z be a spurious solution with zero components and $\{h_1, \dots, h_s\}$ be the set of its zero components. We can truncate Z to the $(w - s)$-vector $\tilde{Z}$, dropping the zero components. It can be easily shown that $\tilde{Z}$ is a solution of the system $J_C(w - s)$.

On the converse, if $\tilde{Z}$ is a solution of $J_C(w - s)$ (spurious or not) and $\{h_1, \dots, h_s\}$ is a subset of $\{0, \dots, w - 1\}$, then we can build the w-vector Z filling with 0 its components in $\{h_1, \dots, h_s\}$ and filling the other components of Z with the components of $\tilde{Z}$. Obviously Z is a spurious solution with zero components of $J_C(w)$ and hence any such solution can be built in this way starting from a generic solution of $J_C(w - s)$.

We are now ready to prove $(**)$. We can consider the solutions with a fixed zero component and the remaining $w - 1$ components such that they form a solution of $J_C(w - 1)$. Varying through the components we collect apparently $w * \mathrm{ns}[w - 1, n]$ solutions, but there are some w-tuples counted more than once. So we have to subtract the solutions with two zeros fixed and the remaining $w - 2$ components such that they form a solution of $J_C(w - 2)$. Fixing two zeros is equivalent to choose a couple in a set of n numbers, which are $\binom{n}{2}$.

Our partial result is then

$$\binom{n}{1} \mathrm{ns}[w - 1, n] - \binom{n}{2} \mathrm{ns}[w - 2, n]$$

This time we have subtracted too many solutions, in fact we are losing the solutions with three zeros and the remaining $w - 3$ components such that they form a solution of $J_C(w - 3)$. The new partial result is

$$\binom{n}{1} \mathrm{ns}[w - 1, n] - \binom{n}{2} \mathrm{ns}[w - 2, n] + \binom{n}{3} \mathrm{ns}[w - 3, n]$$

It is clear that continuing this way we get the desired result. □

In next lemma we show an iterative formula for $\mathrm{spd}[w, n]$, whose proof is reported in the Appendix.

Lemma 5.4. *Let C be a binary $[n, k, d]$ cyclic code (with n odd) such that $1 \in S_C$ and let w be an even integer such that $2 \le w \le n - 1$. Then*

$$\mathrm{spd}[w, n] = n \left(\sum_{i=1}^{w/2} \binom{w - 1}{2i - 1} \mathrm{spd}[w - 2i, n - 1] \right)$$

We can collect our previous results in the following theorem:

Theorem 5.1. *Let C be a binary $[n, k, d]$ cyclic code (with n odd) such that $1 \in S_C$ and let w be an integer such that $2 \le w \le n - 1$. Suppose that for $2 \le w' < w$ the system $J_C(w')$ has only spurious solutions. Then*

$$\mathrm{sp}[w, n] = \mathrm{spd}[w, n] + \mathrm{spz}[w, n],$$

with

$$\mathrm{spd}[w,n] = \begin{cases} n\left(\sum_{i=1}^{w/2}\binom{w-1}{2i-1}\mathrm{spd}[w-2i,n-1]\right), & \textit{for } w \textit{ even} \\ 0, & \textit{for } w \textit{ odd} \end{cases}$$

$$\mathrm{spz}[w,n] = \sum_{i=1}^{w}(-1)^{i+1}\binom{w}{i}\mathrm{sp}[w-i,n]$$

Next corollary is a direct consequence of previous theorem:

Corollary 5.2. *Let C be a binary $[n,k,d]$ cyclic code such that $1 \in S_C$ and n is odd. If w is such that also $J_C(w')$ has only spurious solutions for $1 \leq w' < w$, then the number of the spurious solutions of $J_C(w)$ does not depend on the other equations of the system,*

Remark 5.3. Previous corollary can be applied to all binary narrow-sense primitive $BCH[n,k,d]$ codes, with $w \leq \delta$.

In the hypothesis of Theorem 5.1 we can compute for some small w the values of $\mathrm{sp}[w,n]$, $\mathrm{spz}[w,n]$ and $\mathrm{spd}[w,n]$. They are just some simple univariate polynomials with variable n. Previous corollary is quite important because it shows that the first step of our method is trivial: it is enough to evaluate these polynomials in the desired n in order to obtain the number of spurious solutions, in a manner completely independent from the designed distance of the code. So the only non-trivial part is the computation of the Gröbner basis of the ideal generated by the polynomials of the system $J_C(w)$. Clearly this basis *does* depend on the designed distance: as large the designed distance is, as many polynomials of the form $z_1^i + \cdots + z_w^i$ are in the system.

In the list below there are some values of $\mathrm{sp}[w,n]$, $\mathrm{spz}[w,n]$ and $\mathrm{spd}[w,n]$ for small w:

$$\begin{aligned} \mathrm{spz}[3,n] &= & 3n+1 \\ \mathrm{spd}[3,n] &= & 0 \\ \mathrm{sp}[3,n] &= & 3n+1 \\ \\ \mathrm{spz}[4,n] &= & 6n+1 \\ \mathrm{spd}[4,n] &= & 3n^2-2n \\ \mathrm{sp}[4,n] &= & 3n^2+4n+1 \\ \\ \mathrm{spz}[5,n] &= & 15n^2+1 \\ \mathrm{spd}[5,n] &= & 0 \\ \mathrm{sp}[5,n] &= & 15n^2+1 \end{aligned}$$

$$\begin{aligned}
\mathrm{spz}[6,n] &= & 45n^2 - 15n + 1\\
\mathrm{spd}[6,n] &= & 15n^3 - 30n^2 + 16n\\
\mathrm{sp}[6,n] &= & 15n^3 + 15n^2 + n + 1
\end{aligned}$$

$$\begin{aligned}
\mathrm{spz}[7,n] &= 105n^3 - 105n^2 + 63n + 1\\
\mathrm{spd}[7,n] &= 0\\
\mathrm{sp}[7,n] &= 105n^3 - 105n^2 + 63n + 1
\end{aligned}$$

6 Some Applications and Alternative Strategies

Now we present two direct applications of our method to binary narrow-sense primitive BCH codes, presenting a simple example and proving a classical general result.

Example 6.1. Let $C = \mathbf{BCH(15, 5)}$. The system associated to C is:

$$J_C(5)\begin{cases} x+y+z+t+u &= 0\\ x^3+y^3+z^3+t^3+u^3 &= 0\\ x^{16}+x &= 0\\ y^{16}+y &= 0\\ z^{16}+z &= 0\\ t^{16}+t &= 0\\ u^{16}+u &= 0 \end{cases}$$

As the designed distance is 5, we know that for lower values of w the system $J_C(w)$ has only spurious solutions (see Remark 5.3). Then the number of spurious solutions is easily computed:

$$\mathrm{sp}[5, 15] = 15 * 15^2 + 1 = 3376$$

The leading terms of the Gröbner basis (with lexicographic order) of $J_C(5)$ are:

$$\{u^{16}, t^{16}, z^3t^{14}u, z^9t^2u, z^{16}, y^2t^2u, y^2z, y^{16}, x\}$$

To compute $\mathrm{ns}(J_C(5))$ we can do standard operations ([3]) and obtain:

$$\mathrm{ns}(J_C(5)) = 5536 > 3376 = \mathrm{sp}[5, 15]$$

So $BCH(15, 5)$ has coincident distances.

The simplest case of a narrow-sense $BCH[n, k, d]$ code, whose distance is still unknown, has length $n = 511$ and designed distance 59 (see [5]). Unfortunately, the computation of a Gröbner basis associated to the corresponding system $J_{BCH(511,59)}(59)$ is impossible with nowadays computers using known algorithms (see Remark 6.2).

To apply these methods to unsolved problems, one needs either new quicker algorithms for the computation of Gröbner bases or general results on the Gröbner bases of the ideals associated to the systems $J_C(w)$. The following fact is well-known (see for example [22], p. 260), but we think that the proof here proposed is interesting due to the method employed.

Fact 6.1. *For any $n \geq 7$, $BCH(n,3)$ has coincident distances.*

Proof. We follow the algorithm of Proposition 5.2.

The system associate to the code is:

$$J_{BCH(n,3)}(3) \begin{cases} x+y+z &= 0, \\ x^{(n+1)}+x &= 0, \\ y^{(n+1)}+y &= 0, \\ z^{(n+1)}+z &= 0, \end{cases}$$

As $\mathbb{F}_q = \mathbb{Z}_2$ and $n+1 = 2^m$ for some m, we have

$$(x+y+z)^{(n+1)} = (x+y+z)^{(2^m)} = x^{2^m} + y^{2^m} + z^{2^m}$$

so the system can be reduced to

$$J'_{BCH(n,3)}(3) \begin{cases} x+y+z &= 0, \\ y^{(n+1)}+y &= 0, \\ z^{(n+1)}+z &= 0, \end{cases}$$

This system is triangular, so the number of all its solutions is $\mathrm{ns}[3,n] = (n+1)^2$. The number of its spurious solutions is easily found $\mathrm{sp}[3,n] = 3n+1$. As for any $n \geq 7$

$$(n+1)^2 = \mathrm{ns}[3,n] \ > \ \mathrm{sp}[3,n] = 3n+1$$

we can conclude (see Proposition 5.1) that there are words of weight 3 in $BCH(n,3)$, showing that $BCH(n,3)$ has coincident distances. □

Next example is important because it shows that our method can also be applied to non-binary codes.

Example 6.2. Let C be the $BCH(8,2,\mathbb{Z}_3)$ code. The system associated to C is:

$$J_C(3) \begin{cases} xa+yb+cz = 0 \\ x^9 - x \qquad = 0 \\ y^9 - y \qquad = 0 \\ z^9 - z \qquad = 0 \\ a^2 - 1 \qquad = 0 \\ b^2 - 1 \qquad = 0 \\ c^2 - 1 \qquad = 0 \end{cases}$$

The leading terms of a Gröbner basis are

$$\begin{cases} x^2y^7z^8c, xy^8z^8c, x^3yz^8c, xy^2z^8bc, y^3z^8bc, x^2y^8zc, x^4y^5z^2c, x^3y^6z^2c, y^2z^8ac, \\ yz^8a^2c, x^2z^8bc, xyz^7b^2c, xz^8b^2c, xy^8bc^2, xy^8zb, xy^8b^2, y^3z^6ac, x^6y^2zc, \\ x^9, y^9, z^9, y^5z^2bc, y^4z^2ac, x^4z^2bc, z^5abc, y^2z^3ab, yz^3abc, y^3zb^2c, z^3abc^2, \\ y^2za^2c, y^2zabc, z^2a^2bc, x^2zb^2c, yzab^2c, z^2ab^2c, za^2b^2c, y^2abc^2, y^3ab, \\ y^2ab^2, x^2yb, ya^2b, a^3, b^3, c^3, xa \end{cases}$$

Doing standard operations we obtain:

$$\mathrm{ns}(J_C(3)) = 648$$

The computation of the number of spurious solutions is summerized as follows. We denote by I_x the system obtained by $J_C(3)$ putting $x = 0$ and we adopt the same notation for I_y and I_z. We denote by I_{xy} the system obtained by $J_C(3)$ putting $x = y = 0$ and analogously for I_{xz} and I_{yz}. We use also the notation $I_{x=y}$ for the system obtained putting both $x = y$ and $\{x \neq 0,\ z \neq 0\}$ (analogously for $I_{x=z}$ and $I_{y=z}$) and the notation $I_{x=y=z}$ for the system obtained with $x = y = z$ and $\{x \neq 0\}$. We use the short notation L_S for the number of solutions of I_S (for example $L_x = \mathrm{ns}(I_x)$).

The spurious solutions of $J_C(3)$ can be divided in two parts: spz and spnz, where spz are the spurious solutions with at least a null variable among the three variables x, y, z and spnz are the remaining ones. With reasonings similar to the arguments used in Section 5, one can see that:

$$\mathrm{spz} \;=\; L_x + L_y + L_z - (L_{xy} + L_{xz} + L_{yz}) + L_{xyz}$$

with Gröbner basis techniques or direct calculations one can get:

$$\mathrm{spz} \;=\; 72 + 72 + 72 - (8 + 8 + 8) + 8 \;=\; 200$$

Using again similar arguments one obtains:

$$\mathrm{spnz} \;=\; L_{x=y} + L_{x=z} + L_{y=z} - 2L_{x=y=z}$$

that is

$$\mathrm{spnz} \;=\; 32 + 32 + 32 - 2 * 16 \;=\; 64$$

Finally we obtain:

$$\mathrm{sp} \;=\; \mathrm{spz} + \mathrm{spnz} \;=\; 200 + 64, \;=\; \mathbf{264} \;<\; \mathbf{648} = \mathrm{ns}(J_C(3))$$

so that we can conclude that C has coincident distances.

Sometimes it is convenient to consider other polynomial systems.

Definition 6.1. *Let C be a binary $[n, k, d]$ cyclic code such that $1 \in S_C$ and n is odd. Let $S_C = \{h_1, \dots, h_v\}$ be a defining set of C and let $1 \le w \le n - 1$. We denote by* $\tilde{\mathbf{J}}_{\mathbf{C}}(\mathbf{w})$ *the system:*

$$\tilde{J}_C(w) \begin{cases} z_1^{h_1} + z_2^{h_1} + \cdots + z_w^{h_1} &= 0 \\ \dots &= 0 \\ z_1^{h_v} + z_2^{h_v} + \cdots + z_w^{h_v} &= 0 \\ z_1^n - 1 &= 0 \\ \dots &= 0 \\ z_w^n - 1 &= 0 \end{cases}$$

The systems $J_C(w)$ and $\tilde{J}_C(w)$ are closely related:

Lemma 6.1. *Let C be a binary $[n, k, d]$ cyclic code such that $1 \in S_C$ and n is odd. Let $S_C = \{h_1, \dots, h_v\}$ be a defining set of C and let $1 \le w \le n - 1$. Then the solutions of $\tilde{J}_C(w)$ are exactly the solutions of $J_C(w)$ which have no zero coordinates.*

Proof. Let $\mathbb{F}$ be the splitting field of $x^n - 1$ over $\mathbb{Z}_2$. If $Z = (Z_1, \dots, Z_w)$ is a solution of $\tilde{J}_C(w)$, then it satisfies the first v equations of $J_C(w)$, as they are the same first v equations of $\tilde{J}_C(w)$. Moreover, for any $z \in \mathbb{F}$ the equation $z^n = 1$ implies the equation $z^{n+1} = z$, so also the other equations are satisfied.

If $Z = (Z_1, \dots, Z_w)$ is a solution of $J_C(w)$ which has no zero coordinates, then for $1 \le i \le w$ there holds $Z_i^{n+1} = Z_i \Longrightarrow Z_i^n = 1$. $\square$

We denote by $\mathbf{ns}(\tilde{J}_C(w))$ the number of all solutions of $\tilde{J}_C(w)$.

Remark 6.1. As all solutions of $\tilde{J}_C(w)$ are the solutions of $J_C(w)$ which have no zero coordinates, the number of spurious solutions of $\tilde{J}_C(w)$ is exactly $\mathrm{spd}[w, n]$.

Let w be odd and such that $\tilde{J}_C(w')$ has only spurious solutions for $1 \le w' \le w$. As $\mathrm{spd}[w, n] = 0$, there are no spurious solutions of $\tilde{J}_C(w)$, leading to the following alternative:

- either the polynomials of $\tilde{J}_C(w)$ generate the whole ring $\mathbb{Z}_2[z_1, \dots, z_w]$ (and so $d > w$)
- or the polynomials of $\tilde{J}_C(w)$ generate a proper ideal of $\mathbb{Z}_2[z_1, \dots, z_w]$ (and so $d = w$)

Previous lemma and previous remark suggest an alternative strategy to find the distance of cyclic codes:

Proposition 6.1. *Let C be a binary $[n, k, d]$ cyclic code (with n odd). If we know that $d \ge \delta$, we can proceed this way:*

- *set $w = \delta$ and construct $\tilde{J}_C(w)$;*
- *count both the number of the spurious solutions of $\tilde{J}_C(w)$ and the number of all its solutions;*
- *if all solutions are spurious, then increase w to $w+1$, construct $\tilde{J}_C(w)$ and come back to previous step*
- *if on the contrary there are non-spurious solutions, then return w.*

The value returned by this algorithm is the distance d of the code.

Proof. The proof is analogous to the proof of Proposition 4.2 □

There is no theoretical differences between the former algorithm and the latter, but it is important to be able to use both. In particular, the computation of the Gröbner basis with $J_C(w)$ can be quite faster that with $\tilde{J}_C(w)$ (and vice-versa), which is the most expensive part of the two algorithms. Below there is a table where we have collected the computation time needed to derive the Gröbner bases of the ideals associated to the two polynomial systems for some simple codes:

Table 1. Comparison between J_C and $\tilde{J}_C$

n	k	S_C	d_{BCH}	d	$\tilde{J}_C$	J_C
15	7	1,3	5	5	3.37	3.08
15	5	1,3,5	7	7	2336.27	482.15
17	9	1	4	5	5.37	59.68
21	12	1,9	3	3	2.11	1.72
21	10	1,7,9	4	4	1.74	1.58
21	10	1,3,7	5	5	2.38	2.27
21	9	1,5	3	3	1.45	1.61
21	9	1,3,9	5	6	2.03	2.27

The columns of Table 1 contain:

- the length n of the code;
- the dimension k of the code;
- a defining set S_C of the code;
- the value of the BCH bound for the code;
- the actual distance d of the code;
- the computation time needed to obtain a Gröbner basis for the ideal of the system $\tilde{J}_C$;
- the computation time needed to obtain a Gröbner basis for the ideal of the system J_C;

The correspondence among the first five columns is taken from [27], p. 494.

Remark 6.2. The computations involving Gröbner bases have been made with the software package ALPI and using a PIII 500. ALPI ("A Lisp POSSO Interface") has been developed by the group of Computational Algebra at the University of Pisa (Italy) and exploits the PoSSo C++ Library. This library provides tools for symbolic solving of polynomial systems and more advanced features ([29]).

All computations have been checked using the software package MAGMA ([23]) at the MEDICIS computational cluster ([24]). Some of the more time-consuming jobs have been done also with other software packages, such as Maple and Macaulay.

The cyclic code $C = \text{Cyclic}(23, 5)$ is called the **binary Golay code** and is a [23, 12, 7] code (see [22], p. 60–70).

Remark 6.3. The binary Golay code is quite studied, in particular because it is a non-trivial example of perfect code. We note that its actual distance is different from its designed distance, proving that the hypothesis $n = 2^m - 1$ is necessary in Fact 6.1

Dealing with binary Golay code, we mix both methods (see Proposition 6.1 and Proposition 4.2) to find the distance of C. The BCH bound assures that $d \geq d_{BCH} = 5$.

First we consider the system $\tilde{J}_C(5)$:

$$\tilde{J}_C(5) \begin{cases} x + y + z + t + u & = 0 \\ x^3 + y^3 + z^3 + t^3 + u^3 & = 0 \\ x^{23} + 1 & = 0 \\ y^{23} + 1 & = 0 \\ z^{23} + 1 & = 0 \\ t^{23} + 1 & = 0 \\ u^{23} + 1 & = 0 \end{cases}$$

The Gröbner basis (Reverse Graded Order) turns out to be $\{1\}$, so there are no codewords of weight 5 in C. The execution time is 5 seconds.

Then we study the system $J_C(6)$:

$$J_C(6) \begin{cases} x + y + z + t + u + v & = 0 \\ x^3 + y^3 + z^3 + t^3 + u^3 + v^3 & = 0 \\ x^{24} + x & = 0 \\ y^{24} + y & = 0 \\ z^{24} + z & = 0 \\ t^{24} + t & = 0 \\ u^{24} + u & = 0 \\ v^{24} + v & = 0 \end{cases}$$

The leading terms of an associated Gröbner basis are:

$$\{x, y^2z, y^2t^2u, z^3t^2u, v^{24}, u^{24}, t^{24}, z^{24}, y^{24}\}$$

The execution time is 112 seconds. Counting the solutions one gets the number: 190464. Using our formulas we can easily compute:

$$\mathrm{spd}[6, 23] = 190464$$

which is the same value. So C has no codewords of weight 6.

Finally, we consider the system $\tilde{J}_C(7)$:

$$\tilde{J}_C(7)\begin{cases} x+y+z+t+u+v+w &= 0\\ x^3+y^3+z^3+t^3+u^3+v^3+w^3 &= 0\\ x^{23}+1 &= 0\\ y^{23}+1 &= 0\\ z^{23}+1 &= 0\\ t^{23}+1 &= 0\\ u^{23}+1 &= 0\\ v^{23}+1 &= 0\\ w^{23}+1 &= 0 \end{cases}$$

The computation lasted more than three days. As the Gröbner basis turns out to be different from $\{1\}$, we have shown that *there are* codewords of weight 7 in C and hence that the actual distance of C is $\mathbf{d} = \mathbf{7}$.

Remark 6.4. As a minimal defining set of C, one can consider $S = \{1\}$. This means that we can find the distance of C also considering systems of the form:

$$H(w)\begin{cases} x_1+x_2+\cdots+x_w &= 0\\ x_1^{24}+x_1 &= 0\\ x_2^{24}+x_2 &= 0\\ v\ldots & \vdots\ \vdots\\ x_w^{24}+x_w &= 0 \end{cases}$$

The Gröbner basis associated to $H(w)$ is the same as the Gröbner basis associated to $J(w)$, so any of the two computations gives the same result. But the execution time are quite differents. For example the computation for $H(6)$ lasted 4848 seconds, while the computation for $J(6)$ lasted only 112 seconds!

Large difference in the computation times can be observed also considering the same variation $\tilde{H}(w)$ to the $\tilde{J}(w)$ system:

$$\tilde{H}(w)\begin{cases} x_1+x_2+\cdots+x_w &= 0\\ x_1^{23}+1 &= 0\\ x_2^{23}+1 &= 0\\ v\ldots & \vdots\ \vdots\\ x_w^{23}+1 &= 0 \end{cases}$$

For example the computation for $\tilde{H}(5)$ lasted 81 seconds, while the computation for $\tilde{J}(5)$ lasted only 5 seconds!

7 Accelerator Polynomials in the Case $\delta = 5$

Now we focus on the case of Cyclic$(n, 5)$ codes, trying to establish when they have coincident distances. In this section we denote by G a Gröbner bases of the ideal generated by the polynomials of $\tilde{J}_C(5)$.

When $n = 2^m - 1$, it is known that Cyclic$(n, 5) = BCH[n, k, 5]$ has coincident distances (see [22], p. 260). But neither other general statements are known, nor efficient algorithms are. Generally, the codimension of such codes is low, even for medium length codes. For example, when $n = 1023$ we have $BCH[1023, 1003, 5]$, which has codimension 20. In these cases, one can find the distance with a brute force check. If n grows large, the problem becomes subtler. Moreover, codes that can correct one or two errors are appropriate for transmission with modern telecommunication physical means (this is the advantage that the telecommunication standard AAL5 has with respect to AAL3/4, see http://www.atmforum.com).

The system $\tilde{J}_C(5)$ is in this case:

$$\tilde{J}_C(5) \begin{cases} x + y + z + t + u & = 0 \\ x^3 + y^3 + z^3 + t^3 + u^3 & = 0 \\ x^n + 1 & = 0 \\ y^n + 1 & = 0 \\ z^n + 1 & = 0 \\ t^n + 1 & = 0 \\ u^n + 1 & = 0 \end{cases}$$

Due to Remark 6.1, either $G = \{1\}$ (and so $d > 5$) or $G \neq \{1\}$ (and so $d = 5$).

In Table 2 we have collected some data:

- the first three columns contain the dimension, the length and the actual distance of $C = $ Cyclic$(n, 5)$;
- the fourth column contains the leading term of the lowest degree polynomial in G (if it is 1, then $G = \{1\}$);
- the fifth column contains the computation time needed to get G;
- the sixth column contains the computation time necessary to get a Gröbner basis for $J_C(5)$;
- we denote by **accelerated time** the value of the sixth column and we postpone the explanation of this concept.

Examining Table 2, we observe that system J_C is almost always faster than system $\tilde{J}_C$, except when $G = \{1\}$ (this happens when $d = 5$). In these cases the difference between the two times is not large and in one case ($n = 49$) $\tilde{J}_C$ performs better than J_C. Note that a large difference is found for the case $n = 51$.

We could conclude that, if we want to investigate the class of Cyclic$(n, 5)$ codes with our methods, it is a good idea to try both methods, hoping that at least one ends in a reasonable time. But even supposing to always choose

Table 2. Computation times for Cyclic$(n, 5)$ with $n \leq 57$

n	k	d	base	time $\tilde{J}_C$	time J_C	acc. time
15	7	5	x	3.37	2.69	**1.37**
21	12	5	x	5.78	3.76	-3.07
27	3	9	1	7.57	7.01	
31	21	5	x	40.47	12.26	-5.23
33	13	10	1	44.18	50.39	
35	11	5	x	469.63	19.43	**10.02**
39	15	10	1	45.51	32.37	
43	15	13	1	35.82	29.50	
45	29	5	x	461.38	249.66	**19.23**
49	7	7	1	140.00	245.31	
51	35	5	x	3799.45	104.11	-26.46
55	15	5	x	305.03	143.32	**29.28**
57	21	14	1	132.30	106.29	**26.41**

the faster of the two methods, it is clear that the growth of the dimension n soon leads to unaffordable computation times (at least with nowadays medium class computers). We are going to propose an idea which can highly reduce computation time in some cases.

Definition 7.1. *We denote by p_1, p_2 and p_3 the following polynomials in $\mathbb{Z}_2[x, y, z, t, u]$:*

$$\begin{aligned} p_1 &= y^2 + yt + t^2 + yu + tu + u^2 + yz + tz + uz + z^2 \\ p_2 &= z^3 + z^2t + zt^2 + t^3 + z^2u + ztu + t^2u + zu^2 + tu^2 + u^3 \\ p_3 &= t^4 + t^3u + t^2u^2 + tu^3 + u^4 \end{aligned}$$

We call these three polynomials the **accelerator polynomials**.

For any n we can form a new system $\mathbf{A_C(5)}$ adding the accelerator polynomials to $\tilde{J}_C(5)$:

$$A_C(5) \begin{cases} x + y + z + t + u & = 0 \\ x^3 + y^3 + z^3 + t^3 + u^3 & = 0 \\ x^n + 1 & = 0 \\ y^n + 1 & = 0 \\ z^n + 1 & = 0 \\ t^n + 1 & = 0 \\ u^n + 1 & = 0 \\ y^2 + yt + t^2 + yu + tu + u^2 + yz + tz + uz + z^2 & = 0 \\ z^3 + z^2t + zt^2 + t^3 + z^2u + ztu + t^2u + zu^2 + tu^2 + u^3 & = 0 \\ t^4 + t^3u + t^2u^2 + tu^3 + u^4 & = 0 \end{cases}$$

Proposition 7.1. *Let C be the $[n, k, d]$ Cyclic$(n, 5)$ code, with n odd. Let $\bar{G}$ be a Gröbner basis for the ideal $\bar{J}$ associated to $A_C(5)$ (with any ordering). If $\bar{G}$ is not $\{1\}$, then C has coincident distances:*

$$d = \delta = 5$$

Proof. Denote as usual by G a Gröbner basis for the ideal $\tilde{J}$ associated to $\tilde{J}_C(5)$. Due to Remark 6.1, either $G = \{1\}$ (and so $d > 5$) or $G \neq \{1\}$ (and so $d = 5$).

If $\bar{G}$ is not $\{1\}$, then $\bar{J}$ is not $Z_2[x, y, z, t, u]$. As $\tilde{J} \subset \bar{J}$, also $\tilde{J}$ is not $\{1\}$, so that $G \neq \{1\}$ and $d = 5$. □

This proposition is important, because it allows a different strategy to check whether C has coincident distances or not:

- compute a Gröbner basis $\bar{G}$ for $A_C(5)$;
- if $\bar{G} \neq \{1\}$, conclude that $d = 5$;
- else, proceed as usual.

The last column of Table 2 contains the times needed to compute $\bar{G}$. We use the convention that it has a negative sign if $\bar{G} = \{1\}$ and $G \neq \{1\}$. The computation of $\bar{G}$ is notably faster than the other two computation, making this strategy much effective when C turns out to have coincident distances.

We have tested our strategy with the $[75, 61, 5]$ Cyclic$(75, 5)$ code: the computation of a Gröbner basis for A_C takes 143.71 seconds, while the computation of a Gröbner basis for $\tilde{J}_C$ took more than 18.000 seconds!

Remark 7.1. The accelerator polynomials used in this section are present in the Gröbner bases of some symmetric systems, see [25] for more details.

8 Conclusions

The method proposed in Section 4 is the first one able to get the distance of non-trivial cyclic codes. But more computational arguments are needed to approach recent results in cyclic code theory. The use of accelerator polynomials is an interesting possibility and further research should address the study of accelerator polynomials for other designed distances. A slightly different approach can be found in [33], where the use of the concentrated solutions has led to the computation of all binary narrow-sense BCH codes with $n = 31$ and coincident distances.

Acknowledgments. The core ideas in this paper are from the Ph.D. thesis of the author (see [34]), so the author wishes to thank Carlo Traverso, the advisor of the thesis, for many fruitful suggestions and hints. A special thank to Federico Ponchio and Simona Settepanella, who gave some contributions to the combinatoric study of the spurious solutions in the binary case. The author is particularly indebted to Marc Giusti, director of UMS MEDICIS ([24]), who let him use remotely the computational resources of the MEDICIS cluster.

Appendix

In this section we prove the following useful lemma:
Lemma 6.4. *Let C be a binary $[n, k, d]$ cyclic code (with n odd) such that $1 \in S_C$ and let w be an even integer such that $2 \le w \le n-1$. Then*

$$\mathrm{spd}[w, n] = n \left(\sum_{i=1}^{w/2} \binom{w-1}{2i-1} \mathrm{spd}[w-2i, n-1] \right)$$

We need some definitions:

Definition 8.1. *Let C be a binary $[n, k, d]$ cyclic code (with n odd) such that $1 \in S_C$. Let w be an even integer such that $2 \le w \le n-1$ and let s be an integer such that $1 \le s \le \frac{w}{2}$. We adopt the following notation:*

- $\mathrm{SPD}[w, n]$ *is the set of the spurious solutions of $J_C(w)$ which have no zero coordinates;*
- $SPD_s[w, n]$ *is the set of the spurious solutions of $J_C(w)$ which satisfy the following conditions:*
  - *they belong to* $\mathrm{SPD}[w, n]$,
  - *there is a $(2s-1)$-element subset A of $\{1, \ldots, w-1\}$ such that $x_j = x_w$ for any $x \in \mathrm{SPD}_s[w, n]$ and any $j \in A$,*
  - *$x_j \ne x_w$ for any $x \in \mathrm{SPD}_s[w, n]$ and any $j \notin A$;*
- *if $x \in \mathrm{SPD}_s[w, n]$, we say that x has $2s$* **coincident coordinates***;*
- $\mathrm{spd}_s[w, n]$ *is the number of elements of* $\mathrm{SPD}_s[w, n]$.

As each element in $\mathrm{SPD}[w, n]$ has double coordinates, the (disjoint) union of the $\mathrm{SPD}_s[w, n]$ gives exactly $\mathrm{SPD}[w, n]$, so that:

$$(a) \qquad \mathrm{spd}[w, n] = \#(\mathrm{SPD}[w, n]) = \sum_{1 \le s \le \frac{w}{2}} \mathrm{spd}_s[w, n]$$

Thanks to (a), Lemma 6.4 follows directly from next fact:

Fact 8.1. *Let C be a binary $[n, k, d]$ cyclic code with n odd such that $1 \in S_C$ and let w be an even integer such that $2 \le w \le n-1$. Then*

$$\mathrm{spd}[w, n]_s = n \binom{w-1}{2s-1} \mathrm{spd}[w-2s, n-1]$$

Proof. For any $x \in \mathrm{SPD}[w, n]_s$ we denote by

$$C(x) = \{j_1 = j_1(x), \ldots, j_{2s} = j_{2s}(x) = w\}$$

the set of ordered coincident coordinates of x, that is for any $1 \le k \le 2s-1$ we have

$$j_k < j_{k+1} \qquad \text{and} \qquad x_{j_k} = x_{j_{k-1}} = x_w \ .$$

Choose a $(2s-1)$-element subset of $\{1,\dots,w-1\}$ and denote it by A. We can associate to A a suitable subset of $\mathrm{SPD}[w,n]_s$ this way:

$$\mathrm{SPD}[w,n]_s(A) = \{x \in \mathrm{SPD}[w,n]_s \mid C(x) = A \cup w\}$$

It is clear that $\mathrm{SPD}[w,n]_s$ is the union of all $\mathrm{SPD}[w,n]_s(A)$, where A varies among all $(2s-1)$-element subsets of $\{1,\dots,w-1\}$. If $A \neq B$ and $x \in \mathrm{SPD}[w,n]_s(A) \cap \mathrm{SPD}[w,n]_s(B)$, then x would have a number of coincident coordinates strictly greater than $2s$, which is in contradiction with $x \in SPD[w,n]_s$. So $\mathrm{SPD}[w,n]_s$ is the *disjoint* union of all $\mathrm{SPD}[w,n]_s(A)$, i.e.

$$\mathrm{SPD}[w,n]_s = \sqcup_{(2s-1)-\text{element subsets of }\{1,\dots,w-1\}} \mathrm{SPD}[w,n]_s(A),$$

which directly implies

$$(b) \quad \mathrm{spd}[w,n]_s = \sum_{(2s-1)-\text{element subsets of }\{1,\dots,w-1\}} \#\big(\mathrm{SPD}[w,n]_s(A)\big).$$

We state that the sets of the form $\mathrm{SPD}[w,n]_s(A)$ have the same number of elements. As a matter of fact for any two $(2s-1)$-element subsets of $\{1,\dots,w-1\}$, say $A = \{j_1,\dots,j_{2s-1}\}$ and $B = \{i_1,\dots,i_{2s-1}\}$, we can consider the permutation $\phi : \{1,\dots,w-1\} \mapsto \{1,\dots,w-1\}$ which switches the coordinates $\{j_h\}$ with the coordinates $\{i_h\}$, inducing a bijection $\Phi : \mathrm{SPD}[w,n]_s(A) \mapsto \mathrm{SPD}[w,n]_s(B)$ such that:

$$\Phi(x_1,x_2,\dots,x_{w-1},x_w) = (x_{\phi(1)},x_{\phi(2)},\dots,x_{\phi(w-1)},x_w).$$

We denote by $\mathbf{L_s}$ the set $\mathrm{SPD}[w,n]_s(\{1,2,\dots,2s-1\})$.

As the $(2s-1)$-element subsets of $\{1,\dots,w-1\}$ are $\binom{w-1}{2s-1}$, one can write (b) in a more explicit form:

$$(c) \qquad \mathrm{spd}[w,n]_s = \binom{w-1}{2s-1} \#(L_s) \ .$$

To finish the proof we have only to count the elements in L_s and to show that they are:

$$n \, \mathrm{spd}[w-2s, n-1] \, .$$

For any x element of L_s, we can consider the $(w-2s)$ vector $\tilde{x}$ obtained by removing from x the first $(2s-1)$ coordinates and the last one. With arguments similar to those used in the proof of Lemma 5.2, one can see that $\tilde{x}$ belongs to $SPD[w-2s,n]$ (but none of its coordinates can have the value x_w) and conversely we can create any element of L_s starting from elements of $SPD[w-2s,n]$. It is easy to show that the "remove operation"

$$\tilde{}: L_s \mapsto \{y \in \mathrm{SPD}[w-2s,n] \mid y_h \neq x_w \text{ for any } h\}, \qquad \tilde{}: x \mapsto \tilde{x}$$

is surjective and that any fibre has cardinality n. As the cardinality of $\{y \in$ SPD[w-2s,n] $\mid y_h \neq x_w$ for any $h\}$ is obviously $\mathrm{spd}[w-2s, n-1]$, we have finished.

□

References

1. Augot, D.: Newton's identities for minimum codewords of a family of alternant codes, *Short Abstract in the Proceedings of IEEE ISIT 95*
2. Augot, D., Levy-dit-Vehel, F.: Bounds on the Minimum Distance of the Duals of BCH Codes, *IEEE Trans. Inf. Theory*, **42**(4), 1257–1260 (1996)
3. Bigatti, A. M., Conti, P., Robbiano, L., Traverso, C.: A Divide and Conquer algorithm for Hilbert-Poincaré Series, Multiplicity and Dimension of Monomial Ideals, *Applied algebra, algebraic algorithms and error-correcting codes (San Juan, PR, 1993)*, pp. 76–88 Berlin Heidelberg New York: Springer (1993)
4. Caboara, M., Mora, T.: The Chen-Reed-Helleseth-Truong Decoding Algorithm and the Gianni-Kalkbrenner Gröbner Shape Theorem, to appear
5. Canteaut, A., Chabaud, F.: A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511, *IEEE Trans. Inf. Theory*, **44**(1), 367–378 (1998)
6. Carlitz, L., Uchiyama, S.: Bounds for exponential sums, *Duke Math. J.* **24**, 37–41 (1957)
7. 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. Inf. Theory*, **40**, 1654–1661 (1994)
8. Chen, X., Reed, I. S., Helleseth, T., Truong, T. K., Algebraic decoding of cyclic codes: A polynomial Ideal Point of View, *Contemporary Mathematics*, **168**, 15–22 (1994)
9. III Cooper, A. B.: Direct solution of BCH decoding equations, *Comm., Cont. and Sign. Proc.*, pp. 281–286 (1990)
10. III Cooper, A. B.: Finding BCH error locator polynomials in one step, *Electronic Letters*, **27**, 2090–2091 (1991)
11. Cox, D., Little, J., O'Shea, D.: *Using Algebraic Geometry*, Berlin Heidelberg New York: Springer 1998
12. Cox, D., Little, J., O'Shea, D.: *Ideals, Varieties and algorithms*, Berlin Heidelberg New York: Springer 1992
13. Fitzpatrick, J.: On the Key Equation, *IEEE Trans. on Inf. Theory*, **41**, 1290–1302 (1995)
14. Gianni, P.: Properties of Gröbner bases under Specialization, *Lect. N. Comp. Sci.*, Vol. **378**, p. 293–297, Berlin Heidelberg New York: Springer 1989
15. Johnson, S. M.: Improved Asymptotic Bounds for Error-Correcting Codes, *IEEE Trans. on Inf. Theory*, **9**, 198–205 (1963)
16. Kalkbrenner, M.: Solving Systems of Algebraic Equations Using Gröbner Bases, *Lect. N. Comp. Sci.*, Vol. **378**, pp. 282–292, Berlin Heidelberg New York: Springer 1989
17. Kasami, T., Fujiwara, T., Shu Lin : An Approximation to the Weight Distribution of Binary Linear Codes, *IEEE Trans. on Inf. Theory*, **31**(6), 769–780 (1985)
18. Krasikov, I., Litsyn, S.: On the Distance Distribution of Duals of BCH Codes, *IEEE Trans. on Inf. Theory*, **45**(1), 247–250 (1999)
19. Laihonen, T., Litsyn, S.: On upper bounds for minimum distance and covering radius for non-binary codes, *Designs, Codes and Cryptography*, **14**(1), 71–80 (1998)

20. Levy-dit-Vehel, F.: Bounds on the Minimum Distance of the Duals of Extended BCH Codes over Fp, *AAECC*, **6**(3), 175–190 (1995)
21. Loustaunau, P., York, E. V.: On the decoding of cyclic codes using Gröbner bases, *AAECC*, **8**(6), 469–483 (1997)
22. MacWilliams, F. J., Sloane, N. J. A.: *The Theory of Error-Correcting Codes*, Amsterdam: North Holland 1977
23. MAGMA, *http://www.maths.usyd.edu.au:8000/u/magma/*
24. MEDICIS, *http://www.medicis.polytechnique.fr*
25. Mora, T., Sala, M.: On the Gröbner bases of some symmetric systems, to appear
26. Moreno, O., Moreno, C. J.: The MacWilliams-Sloane Conjecture on the Tightness of the Carlitz-Uchiyama Bound and the Weights of Duals of BCH Codes, *IEEE Trans. Inf. Theory*, **40**(6), 1894–1907 (1994)
27. Peterson, W. W., Weldon, Jr., E. J.: *Error Correcting Codes*, MIT Press, 1972
28. Plotkin, M.: Binary Codes with Specified Minimum Distance, *IEEE Trans. Inf. Theory*, **6**, 445–450 (1960)
29. POSSO, *http://posso.dm.unipi.it*
30. Sala, M., Tamponi, A.: A Linear Programming Estimate of the Weight Distribution of $BCH(255, k)$, *IEEE Trans. on Inf. Theory*, **46**(6), 2235–2237 (2000)
31. Sala, M., Ponchio, F.: A lower bound on the distance of cyclic codes, to appear
32. Sala, M.: Upper bounds on the dual distance of $BCH(255, k)$, to appear
33. Sala, M.: Gröbner bases, accelerator polynomials and binary cyclic codes with coincident distances, to appear
34. Sala, M.: On Some Algebraic Methods in the Theory of Error Correcting Codes, *Ph.D. Thesis in Mathematics, University of Milan, Italy*, accademic year 1999/2000