On the Decoding Complexity of Cyclic Codes Up to the BCH Bound

Davide Schipani
Mathematics Institute
University of Zürich
CH-8057 Zürich
davide.schipani@math.uzh.ch

Michele Elia
Dipartimento di Elettronica
Politecnico di Torino
IT-10129, Torino
elia@polito.it

Joachim Rosenthal
Mathematics Institute
University of Zürich
CH-8057 Zürich
http://www.math.uzh.ch/aa

***Abstract*—The standard algebraic decoding algorithm of cyclic codes $[n,k,d]$ up to the BCH bound t is very efficient and practical for relatively small n while it becomes unpractical for large n as its computational complexity is $O(nt)$. Aim of this paper is to show how to make this algebraic decoding computationally more efficient: in the case of binary codes, for example, the complexity of the syndrome computation drops from $O(nt)$ to $O(t\sqrt{n})$, and that of the error location from $O(nt)$ to at most $\max\{O(t\sqrt{n}), O(t^2 \log(t)\log(n))\}$.**

Keywords: Cyclic codes, Syndrome computation, Decoding complexity, Reed Solomon codes, Root search

Mathematics Subject Classification (2010): 94B15, 94B35

I. Introduction

The algebraic decoding of cyclic codes up to the BCH bound, as obtained early in the sixties with the contribution of many people, was considered very efficient for the needs of that time ([1], [2], [15], [16], [18], [19]). However, today we can and need to manage error correcting codes of sizes that require more efficient algorithms, possibly at the limit of their theoretical minimum complexity. We are proposing here an algorithm that goes in this direction.

Let us summarize, for easy reference, the algebraic decoding of cyclic codes: let $\mathcal{C}$ be an $[n,k,d]$ cyclic code over a finite field $\mathbb{F}_q$, $q=p^s$ for a prime p, with generator polynomial of minimal degree $r=n-k$

$$g(x) = x^r + g_1x^{r-1} + \ldots + g_{r-1}x + g_r \quad ,$$

$g(x)$ dividing x^n-1, and let α be a primitive n-th root of unity lying in a finite field $\mathbb{F}_{p^m}$, where the extension degree is the minimum integer m such that n is a divisor of p^m-1. Assuming that $\mathcal{C}$ has BCH bound t, then $g(x)$ has $2t$ roots with consecutive power exponents, so that the whole set of roots is

$$\mathfrak{R} = \{\alpha^{\ell+1}, \alpha^{\ell+2}, \ldots, \alpha^{\ell+2t}, \alpha^{s_{2t+1}}, \ldots, \alpha^{s_r}\} \quad ,$$

where it is not restrictive to take $\ell=0$ as it is usually done.

Let $R(x) = g(x)I(x) + e(x)$ be a received code word such that the error pattern $e(x)$ has no more than t errors. The Gorenstein-Peterson-Zierler decoding procedure ([15], [18]), which is a standard decoding procedure for every cyclic code up to the BCH bound, is made up of four steps:

- Computation of $2t$ syndromes: $S_j = R(\alpha^j)$, $j=1,\ldots,2t$.
- Computation of the error-locator polynomial $\sigma(z) = \sigma_t z^t + \sigma_{t-1}z^{t-1} + \cdots + \sigma_1 z + 1$ (we are assuming the case that exactly t errors occurred; if there are $t_e < t$ errors, this step would output a polynomial of degree t_e).
- Computation of the roots of $\sigma(z)$ in the form α^{-j_h}, $h=1,\ldots,t$, yielding the error positions j_h.
- Computation of the error magnitudes.

Efficient implementations of this decoding algorithm combine the computation of $2t$ syndromes using Horner's rule, the Berlekamp-Massey algorithm to obtain the error-locator polynomial, the Chien search to locate the errors, and the evaluation of Forney's polynomial to estimate the error magnitudes.

The computation of the $2t$ syndromes using Horner's rule requires $2tn$ multiplications in $\mathbb{F}_{q^m}$, which may be prohibitive when n is large. The Berlekamp-Massey algorithm has multiplicative complexity $O(t^2)$ ([2], [11]), is very efficient and will not be discussed further later on. The Chien search requires again $O(tn)$ multiplications in $\mathbb{F}_{q^m}$ and Forney's algorithm $O(t^2)$ ([11]). Notice that this fourth step is not required if we deal with binary codes and that both the first and the fourth steps consist primarily in polynomial evaluations, so they can benefit from any efficient polynomial evaluation algorithm, as we will show.

The standard decoding procedure is satisfactory when the code length n is not too large (say $< 10^3$) and efficient implementations are set up taking advantage of the particular structure of the code. The situation changes dramatically when n is of the order of 10^6 or larger. In this case a complexity $O(tn)$, required by the syndrome evaluations and by the Chien search, is not acceptable anymore.

This paper describes some methods to make these steps more efficient and practical even for large n. We will follow the usual approach of focusing as above in computing the number of multiplications, as they are more expensive than sums (see also [5]).

The paper is structured as follows: Section II concerns the computation of syndromes. Section III deals with the computation of the roots of the error-locator polynomial as well as the corresponding error positions; the error locator polynomial is supposed to be given (being computed by Berlekamp-Massey algorithm). Finally, Section IV gives a numerical example illustrating the whole procedure.

II. Syndrome Evaluation

Let β be any element of $\mathfrak{R}$, the standard Horner's rule ([12],[13]) allows us to compute $R(\beta)$ in at most n products, thus for the computation of $2t$ syndromes we have the estimate $O(tn)$. However, in [21], [6] we showed that polynomials over a finite field of characteristic p can be evaluated more efficiently by exploiting the Frobenius automorphism, i.e. the mapping $\sigma(\beta) = \beta^p$, with a significant computational cost reduction.

Briefly, to evaluate a polynomial $r(x)$ of degree n over $\mathbb{F}_{p^s}$, in β, an element of $\mathbb{F}_{p^m}$, we write $r(x)$ as a linear combination of s polynomials $r_i(x)$ over $\mathbb{F}_p$

$$r(x) = r_0(x) + \gamma r_1(x) + \cdots + \gamma^{s-1} r_{s-1}(x) \quad ,$$

where $\{1, \gamma, \ldots, \gamma^{s-1}\}$ is basis for $\mathbb{F}_{p^s}$. Thus $r(\beta)$ is obtained as a liner combination of s field elements $r_i(\beta)$. To evaluate a polynomial $R(x)$ over $\mathbb{F}_p$ in β, one can profit by writing

$$R(x) = R_{1,0}(x^p) + xR_{1,1}(x^p) + \cdots + x^{p-1}R_{1,p-1}(x^p) \quad ,$$

where $R_{1,0}(x^p)$ collects the powers of x with exponent a multiple of p and in general $x^i R_{1,i}(x^p)$ collects the powers of the form x^{ap+i}, with $a \in \mathbb{N}$ and $0 \le i \le p-1$. Thus $R(\beta)$ can be computed by evaluating β^p, then computing every $R_{1,i}(\beta^p)$ and finally computing the linear combination. This procedure requires, for example, nearly $n/2$ multiplications in the binary case, but the big advantage is that it can be iterated. After L steps, we need to evaluate p^L polynomials $R_{L,i}(x)$ of degree at most $\lfloor \frac{n}{p^L} \rfloor$. By a convenient number L of iterations, and with a smart arrangement of the multiplications ([6]), one can achieve an overall complexity of approximately $2s\sqrt{n(p-1)}$. Outstanding is in particular the case of binary codes, where the complexity is $2\sqrt{n}$.

It should be remarked that in hardware implementations, the proposed algorithm allows a strong parallelism, while Horner's rule is inherently serial. In fact, if L is the number of iterations, the evaluation of the p^L polynomials $R_{L,i}(x)$ can be done in parallel. Moreover an additional gain may be given by the pre-computation of the powers of β, especially when the number of syndromes to be computed is big. Furthermore, like in Horner's rule, multiplication by β or its powers can be performed using Linear Feedback Shift Registers ([10], [15], [17]) with a further speed up at a very small cost, while the p-power operations would benefit from the use of a normal basis ([12], [14]).

Lastly, it should be also remarked that, in particular situations, a better cost reduction can be obtained by means of a different use of the Frobenius automorphism and a careful choice of the number of iterations. As an example, in [21] we described a method of computing the syndromes for the famous Reed-Solomon code $[255, 223, 33]$ over $\mathbb{F}_{2^8}$ used by NASA ([24]), that employs 6735 multiplications to evaluate 32 syndromes, versus 8159 multiplications that are necessary using Horner's rule. The direct application of the method outlined above would not be convenient in this situation because of the particular parameters involved.

III. Roots of the error-locator polynomial

Once the error locator polynomial $\sigma(z)$ is computed from the syndromes using the Berlekamp-Massey algorithm, its roots, represented in the form $\alpha^{-\ell_i}$, correspond to the error positions ℓ_i, $i = 1, \ldots, t$, which are generally found by testing $\sigma(\alpha^{-i})$ for all n possible powers α^{-i} with an algorithm usually referred to as the Chien search. In this approach, if $\sigma(\alpha^{-j}) = 0$ an error in position j is recognized, otherwise the position is correct. However, this simple mechanism can be unacceptably slow when n is large since its complexity is $O(tn)$: aim of this Section is to describe a less costly procedure.

The Cantor-Zassenhaus probabilistic factorization algorithm ([3]) is very efficient in factoring a polynomial and consequently in computing the roots of a polynomial ([9, Chapter 14.5, Corollary 14.16]). Since $\sigma(z)$ is the product of t linear factors $z + \rho_i$ over $\mathbb{F}_{q^m}$ (i.e. ρ_i is a q-ary polynomial in α of degree $m-1$), this factoring algorithm can be directly applied to separate these t factors. The error positions ℓ_i are then obtained by computing the discrete logarithm of $(\rho_i)^{-1} = \alpha^{\ell_i}$ to base α. This task can be performed by Shank's algorithm ([22]), which we revisit below. The overall expected complexity of finding the error positions with this algorithm is $O(mt^2 \log t)$ ([9, Chapter 14]), plus $O(t\sqrt{n})$, where the second addend comes from Shank's algorithm. It is evident how this complexity is better than $O(n)$ in most cases, in particular when t is small in comparison to n. Moreover, considering the recently improved version in [7], sketched briefly in the remark below, it is possible to obtain better computational estimates.

a) Cantor-Zassenhaus algorithm: The algorithm of Cantor-Zassenhaus ([3]) is described here for easy reference (see also the improvement in [7]). We describe only the case of characteristic 2, which is by far the most common in practice; the interested reader can find the general situation in [3], [7]. Assume that $p(z)$ is a polynomial over $\mathbb{F}_{2^m}$ that is a product of t polynomials of degree 1 over the same field $\mathbb{F}_{2^m}$, m even (when m is odd it is enough to consider a quadratic extension and proceed as in the case of even m). Suppose that α is a known primitive element in $\mathbb{F}_{2^m}$, and set $\ell_m = \frac{2^m-1}{3}$, then $\rho = \alpha^{\ell_m}$ is a primitive cubic root in $\mathbb{F}_{2^m}$, so that ρ is a root of z^2+z+1. The algorithm consists of the following steps:

1) Generate a random polynomial $b(z)$ of degree $t-1$ over $\mathbb{F}_{2^m}$
2) Compute $a(z) = b(z)^{\ell_m} \bmod p(z)$;
3) IF $a(z) \neq 0, 1, \rho, \rho^2$, THEN at least a polynomial among

$$\gcd\{p(z), a(z)\}, \gcd\{p(z), a(z)+1\},$$
$$\gcd\{p(z), a(z)+\rho\}, \gcd\{p(z), a(z)+\rho^2\}$$

will be a non trivial factor of $p(z)$, ELSE repeat from point 1.
4) Iterate until all linear factors of $p(z)$ are found.

b) Remark 1: As shown in [7], the polynomial $b(z)$ can be chosen of the form $z+\beta$, using $b(z) = z$ as initial choice. Let θ be a generator of the cyclic subgroup of $\mathbb{F}^*_{2^m}$ of order ℓ_m. If $z^{\ell_m} = \rho^i \bmod \sigma(z)$, $i \in \{0,1,2\}$, then each root ζ_h of $\sigma(z)$ is of the form $\alpha^i\theta^j$. If this is the case, which does not allow us to find a factor, we repeat the test with $b(z) = z+\beta$ for some β and we will succeed as soon as the elements $\zeta_h+\beta$ are not all of the type $\alpha^i\theta^j$ for the same $i \in \{0,1,2\}$. This can be shown to happen probabilistically very soon, expecially when the degree of $\sigma(z)$ is high.

c) Shank's algorithm: Shank's algorithm can be applied to compute the discrete logarithm in a group of order n generated by the primitive root α. The exponent ℓ in the equality

$$\alpha^\ell = b_0 + b_1\alpha + \cdots + b_{s-1}\alpha^{s-1} \quad .$$

is written in the form $\ell = \ell_0 + \ell_1\lceil\sqrt{n}\rceil$. A table $\mathcal{T}$ is constructed with $\lceil\sqrt{n}\rceil$ entries $\alpha^{\ell_1\lceil\sqrt{n}\rceil}$ which are sorted in some well defined order, then a cycle of length $\lceil\sqrt{n}\rceil$ is started computing

$$A_j = (b_0+b_1\alpha+\cdots+b_{s-1}\alpha^{s-1})\alpha^{-j} \quad j = 0,\ldots,\lceil\sqrt{n}\rceil-1 \ ,$$

and looking for A_j in the Table; when a match is found with the κ-th entry, we set $\ell_0 = j$ and $\ell_1 = \kappa$, and the discrete logarithm ℓ is obtained as $j+\kappa\lceil\sqrt{n}\rceil$.
This algorithm can be performed with complexity $O(\sqrt{n})$ both in time and space (memory). In our scenario, since we need to compute t roots, the complexity is $O(t\sqrt{n})$.

d) Remark 2: The Cantor-Zassenhaus algorithm finds the roots $X_j = \alpha^{\ell_j}$ of the reciprocal of the error locator polynomial, then the baby-step giant-step algorithm of Shank's finds the error positions ℓ_js. As said in the introduction, this is the end of the decoding process for binary codes. For non-binary codes, Forney's polynomial $\Gamma(x) = \sigma(x)(S(x)+1) \bmod x^{2t+1}$, where $S(x) = \sum_{i=1}^{2t} S_i x^i$ ([23]), yields the error values

$$Y_j = -X_j \frac{\Gamma(X_j^{-1})}{\sigma'(X_j^{-1})} \quad .$$

Again we remark that this last step can benefit from an efficient polynomial evaluation algorithm, such as the one discussed in Section 2.

e) Remark 3: We observe that the above procedure can be used to decode beyond the BCH bound, up to the minimum distance, whenever the error locator polynomial can be computed from a full set of syndromes ([4], [8], [20], [23]).

IV. A NUMERICAL EXAMPLE

In the previous sections we presented methods to compute syndromes and error locations in the GPZ decoding scheme of cyclic codes up to their BCH bound, which are asymptotically better than the classical algorithms. The following example illustrates the complete new procedure.
Consider a binary BCH code $[63, 45, 7]$ with generator polynomial

$$g(x) = x^{18}+x^{17}+x^{14}+x^{13}+x^9+x^7+x^5+x^3+1$$

whose roots are

$$\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^{32}, \alpha^3, \alpha^6, \alpha^{12},$$
$$\alpha^{24}, \alpha^{48}, \alpha^{33}, \alpha^5, \alpha^{10}, \alpha^{20}, \alpha^{40}, \alpha^{17}, \alpha^{34},$$

thus the BCH bound is 3. Let $c(x) = g(x)I(x)$ be a transmitted code word, and the received word be

$$r(x) = x^{57}+x^{56}+x^{53}+x^{52}+x^{50}+x^{48}+x^{46}+x^{44}+x^{42}+$$
$$x^{39}+x^{31}+x^{18}+x^{17}+x^{14}+x^{13}+x^7+x^5+x^3+1$$

where 3 errors occurred. The 6 syndromes are

$$\begin{cases} S_1 = \alpha^5 + \alpha^2 + \alpha \\ S_2 = S_1^2 \\ S_3 = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha \\ S_4 = S_1^4 \\ S_5 = \alpha^5 + \alpha^2 + 1 \\ S_6 = S_3^2 \end{cases} .$$

For example, S_1 has been computed considering $r(x)$ as

$[r_{3,0}+zr_{3,1}+y(r_{3,2}+zr_{3,3})]+x[r_{3,4}+zr_{3,5}+y(r_{3,6}+zr_{3,7})]$,

with $y = x^2$, $z = x^4$, $w = x^8$ and

$$\begin{cases} r_{3,0} = w^7 + w^6 + 1 \\ r_{3,1} = w^6 + w^5 \\ r_{3,2} = w^6 + w^5 + w^2 \\ r_{3,3} = w^5 + w \\ r_{3,4} = w^7 + w^2 \\ r_{3,5} = w^6 + w + 1 \\ r_{3,6} = 1 \\ r_{3,7} = w^4 + w^3 + 1 \end{cases}$$

with only 16 products, namely 3 to compute α^2, α^4 and α^8, 6 for the powers of w up to w^7 and 7 multiplications by x, y and z.

The coefficients of the error locator polynomial turn out to be

$$\begin{cases} \sigma_1 = \alpha^5 + \alpha^2 + \alpha \\ \sigma_2 = \alpha^3 + \alpha^4 + \alpha \\ \sigma_3 = \alpha^4 + \alpha^5 + \alpha^2 \end{cases} .$$

The roots of $\sigma^*(z) = z^3\sigma(z^{-1}) = \prod_{i=1}^{3}(z - \alpha^{\ell_i})$ are computed as follows using the Cantor-Zassenhaus algorithm.

Let $\rho = \alpha^{21}$ be a cube root of the unity, consider a random polynomial, for instance $z + \rho$, of degree less than 3 and compute $a(z) = (z + \rho)^{21}$ modulo $\sigma^*(z)$ (the exponent of $z + \rho$ is $\frac{2^m-1}{3} = \frac{63}{3} = 21$)

$$(\alpha^5+\alpha^4+\alpha^2+\alpha+1)z^2+(\alpha^3+\alpha+1)z+\alpha^5+\alpha^4+x^3+1 \quad .$$

In this case $a(z)$ has no root in common with $\sigma^*(z)$, while

$$\gcd(a(z)+1, \sigma^*(z)) = z + (\alpha^4 + \alpha^3 + 1) \qquad (\ell_1 = 31),$$
$$\gcd(a(z)+\rho, \sigma^*(z)) = z + (\alpha^5 + \alpha^4 + \alpha^2 + 1) \quad (\ell_2 = 9),$$
$$\gcd(a(z)+\rho^2, \sigma^*(z)) = z + (\alpha^3 + \alpha) \qquad (\ell_3 = 50).$$

The error positions have been obtained using Shank's algorithm with a table of 8 entries, and a loop of length 8 for each root, for a total of 24 searches versus 63 searches of Chien's search.

Acknowledgment

The Research was supported in part by the Swiss National Science Foundation under grant No. 132256

References

[1] E. Berlekamp, *Algebtraic Coding Theory*, McGraw-Hill, New York, 1968.

[2] R.E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge University Press, Cambridge, 2003.

[3] D.G. Cantor, H. Zassenhaus, A new Algorithm for Factoring Polynomials over Finite Fields, *Math. of Coputation*, Vol. 36, N. 154, April 1981, pp.587-592.

[4] M. Elia, Algebraic Decoding of the $(23, 12, 7)$ Golay Code *IEEE Trans. on Information Theory*, vol. IT-33, No.1, January 1981, p.150-151.

[5] M. Elia, M. Leone, On the Inherent Space Complexity of Fast Parallel Multipliers for $GF(2^m)$, *IEEE Trans. on Computer*, vol. 51, N.3, March 2002, p.346-351.

[6] M. Elia, J. Rosenthal, D. Schipani, Polynomial evaluation over finite fields: new algorithms and complexity bounds, *www.arxiv.org*, 2011.

[7] M. Elia, D. Schipani, Improvements on Cantor-Zassenhaus Factorization Algorithm, *www.arxiv.org*, 2010,

[8] G.-L. Feng, K.K. Tzeng, Decoding cyclic and BCH codes up to actual minimum distance using nonrecurrent syndrome dependence relations, *IEEE Trans. on Inform. Th.*, IT-37, No.6, 1991, pp.1716-1723.

[9] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, Cambridge Univ. Press, 1999.

[10] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, Laguna Hills, 1982.

[11] J. Hong, M. Vetterli, Simple Algorithms for BCH Decoding, *IEEE Trans. on Communications*, Vol. 43, No. 8, August 1995, pp.2324-2333.

[12] D. Jungnickel, *Finite Fields, Structure and Arithmetics*, Wissenshaftsverlag, Mannheim, 1993.

[13] D.E. Knuth, *The Art of Computer Programming*, Seminumerical algorithms, vol. II, Addison-Wesley, Reading Massachussetts, 1981.

[14] R. Lidl, H. Niederreiter, *Finite Fields*, Addison-Wesley, Reading, Mass., 1986.

[15] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North Holland, New York, 1977.

[16] J.L. Massey, Shift-Register Synthesis and BCH decoding, *IEEE Trans. on Inform. Th.*, IT-15, 1969, pp.122-127.

[17] R.J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Press, Boston, 1987.

[18] W.W. Peterson, E.J. Weldon, *Error-Correcting Codes*, MIT, New York, 1981.

[19] V.S. Pless, W.C. Huffman, *Handbook of Coding Theory*, vol. I and II, Noth-Holland, Amsterdam, 1998.

[20] Reed, I.S., Truong, T.K., Chen, X., Yin, The algebraic decoding of the $(41, 21, 9)$ quadratic residue code *IEEE Trans. on Inform. Th.*, IT-38, No.3, 1992, pp.974-986.

[21] D. Schipani, M. Elia, J. Rosenthal, Efficient evaluation of polynomials over finite fields, *Proceedings of 2011 Australian Communications Theory Workshop*, 2011, pp.154-157.

[22] D. Shanks, Class number, a theory of factorization and genera, *Proc. Symp. Pure Math.*, AMS, Providence R.I., No.20, 1971, pp.415-440.

[23] S.B. Wicker, *Error control systems for Digital Communication and Storage*, Prentice-Hall, Englewood Cliffs, N.J., 1995.

[24] S.B. Wicker, V.K. Bhargava, eds. *Reed-Solomon codes and their applications*, IEEE Press, Piscataway, N.J., 1994.