Fourteenth International Workshop on Algebraic and Combinatorial Coding Theory
September 7–13, 2014, Svetlogorsk (Kaliningrad region), Russia *pp. 307–312*

Euclidean algorithm for linearized polynomials

Igor Y. Sysoev igor.sisoev@gmail.com
Moscow Institute of Physics and Technology

Abstract. In this paper a new Euclidean algorithm is proposed. This algorithm has less calculation complexity. The main idea is using simplified inversion in Galois Field. Proposed algorithm reduces calculation complexity decoding for rank codes. One multiplier, one invertor and 2 multiplexors are required for the implementation. In the case of additional multiplier data processing delay may be decreased by 2 times. Also such algorithm is useful when syndrome calculator is based on weak self orthogonal bases.

1 Introduction

A practical rank codes application [1] is network coding. Network coding is useful for sensors networks. Sensor network is a great number of stand-alone devices (sensors) which are connected between each others. These sensors transmit data to the central node. Sensors can transmit not only separate packets but also packet superposition. So any error can be distributed in entire network. Rank codes based on rank metric effectively correct such errors.

A key problem is minimizing resources in such networks. So it is necessary to use algorithms with lower computational complexity. A finding locator polynomial procedure takes over a third part time of decoding operation. Depending on component base, decoder uses Berlekamp-Massey algorithm [2], in software implementation, or Euclidean algorithm for linearized polynomials [3], in hardware implementation. In this paper a new Euclidean algorithm modification is proposed. The algorithm has less calculation complexity. The main idea is using simplified operation (inversion) in Galois Field by using special basis. The similar idea with using non standard basis for decoding rank codes has been proposed by Kschischang and Silva [4].

2 Weak self orthogonal basis

First it is necessary to introduce a definition of weak self orthogonal basis and describe its properties. Let $GF(2)$ is the base field and $GF\left(2^{2^M}\right)$ is an extension field ($M = 0, 1, 2, ...$). It is necessary to define basis from $GF\left(2^{2^M}\right)$

over $GF(2)$

$$g_0(N) = (g_1, g_2, ..., g_N), N = 2^M \tag{1}$$

and special matrix

$$G_N = \begin{Vmatrix} g_1 & g_2 & \cdots & g_N \\ g_1^2 & g_2^2 & \cdots & g_N^2 \\ \cdots & \cdots & \cdots & \cdots \\ g_1^{2^{N-1}} & g_2^{2^{N-1}} & \cdots & g_N^{2^{N-1}} \end{Vmatrix} = \begin{Vmatrix} g_1^{[0]} & g_2^{[0]} & \cdots & g_N^{[0]} \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_N^{[1]} \\ \cdots & \cdots & \cdots & \cdots \\ g_1^{[N-1]} & g_2^{[N-1]} & \cdots & g_N^{[N-1]} \end{Vmatrix}. \tag{2}$$

Matrix (2) is associated with the basis (1). It is worth noting that any matrix (2) can be tied to proper basis (1). For the sake of simplicity the notation $[i]$ is used for 2^i (Frobenius power). It is necessary to define weak self orthogonal basis [5].

Definition 1. *Basis $g_0(N) = (g_1, g_2, ..., g_N)$ is said to be weak self orthogonal bases if*

$$G_N G_N^T = D, \tag{3}$$

where D is some diagonal matrix, which is not a product of unity matrix and some element from the extension field.

The following result from lemma 1 is required.

Lemma 1. *Let $f_0 \in GF\left(2^{2^M}\right)$ is an element that $f_0^{1+2^N} = -1$. If $g_0(N) = (g_1, g_2, ..., g_N)$ from $GF\left(2^{2^M}\right)$ is weak self orthogonal, then*

$$g_0(2N) = (g_1, g_2, ..., g_N, f_0 g_1, f_0 g_2, ..., f_0, g_N) \tag{4}$$

is weak self orthogonal basis from $GF\left(2^{2^{M-1}}\right)$.

Proof. See [5]. $\square$

The lemma 1 help us to develop algorithm for construction of a higher dimension weak self orthogonal basis from a lower dimension weak self orthogonal basis. Furthermore, one can construct a basis of the dimension from $GF\left(2^{2^M}\right)$ with such properties using weak self orthogonal basis from $GF\left(2^{2^0}\right)$. Hence the multiplication operation in new basis from a higher dimension field is simplified by replacing all multiplications in a higher dimension field by the multiplications in a lower dimension field. In the work [6] efficiency of using weak self orthogonal basis for computing rank code syndrome has been proved.

3 Inversion complexity

For weak self orthogonal basis the following condition is satisfied

$$f_0^2 = \alpha + \beta f_0, f_0^{-1} = \alpha^{-1}\beta + \alpha^{-1} f_0, \tag{5}$$

where $\alpha, \beta \in GF\left(2^{[M-1]}\right)$ are known constants. One can find inverse for f_0 using the following formula. For the checking the equation (5)

$$f_0^{-1} f_0 = \alpha^{-1}\beta f_0 + \alpha^{-1} f_0^2 = \alpha^{-1}\beta f_0 + \alpha^{-1}\alpha + \alpha^{-1}\beta f_0 = 1. \tag{6}$$

In the simple case ($\alpha = 1$) $f_0 + f_0^{-1} = f_0 + f_0 + \beta = \beta$. Now consider any element d_0 and derive a inversion formula d_0^{-1}

$$d_0 = m + l f_0, d_0^{-1} d_0 = 1. \tag{7}$$

It is necessary to find a special form for products d_0 and d_1

$$d_1 = m + l f_0^{-1}, d_0 d_1 = m^2 + ml(f_0 + f_0^{-1}) + n^2 = (m+n)^2 + ml\beta = \phi, \tag{8}$$

where $\psi \in A$ [6]. One can express an inverse d_0^{-1} through d_1 and ϕ $d_0^{-1} = d_1 \phi^{-1}$. So lets construct formula for inversion for any extension field element

$$d_0^{-1} = (m + l f_0^{-1}) \left((m+n)^2 + ml\beta\right)^{-1} = (d_0 + l\beta)\left((m+l)^2 + ml\beta\right)^{-1}. \tag{9}$$

Thus, inversion in weak self orthogonal basis $GF(2^N)$ requires 3 multiplications in $GF\left(2^{N/2}\right)$, 1 squaring and 1 inversion in $GF\left(2^{N/2}\right)$. Write squaring complexity in weak self orthogonal basis

$$d_0^2 = m^2 + l^2 f_0^2 = m^2 + \alpha l^2 + \beta l^2 f_0. \tag{10}$$

So squaring complexity is evaluated

$$\mathcal{C}_{sqr}(N) = 2\mathcal{C}_{sqr}(N/2) + \mathcal{C}_{mult}(N/2). \tag{11}$$

If, in case $\mathcal{C}_{sqr}(1) = 2^0 = 1$, then

$$\mathcal{C}_{sqr}(N) = \sum_{i=1}^{\log_2^N} 2^{i-1} \mathcal{C}_{mult}(N/2^i). \tag{12}$$

In such algorithm inversion is represented through multiplication in a lower dimension fields. The the entire operation complexity is evaluated as

$$\mathcal{C}_{inv}(N) = 3\mathcal{C}_{mult}(N/2) + \mathcal{C}_{sqr}(N/2) + \mathcal{C}_{inv}(N/2), \tag{13}$$

where $\mathcal{C}_{mult}(N)$ are multiplication complexity operation in $GF\left(2^N\right)$. Denote $M = \log_2 N$. Then the final inversion complexity are expressed

$$\mathcal{C}_{inv} = \sum_{j=0}^{M-1} 3\mathcal{C}_{mult}(2^j) + \mathcal{C}_{sqr}(2^j) + \mathcal{C}_{inv}(1), \tag{14}$$

where $\mathcal{C}_{inv}(1)$ stands for inversion complexity in a small dimension field, for example $GF(2^2)$. The inversion in a lower dimension field may be implemented by means of look up table. Multiplication may be implemented using Karatsuba-Ofman method, that is $\mathcal{C}^K_{mult}(N) = \mathcal{O}\left(N^{\log_2 3}\right)$, one got $\mathcal{C}_{inv}(N) \approx \frac{3}{2}\mathcal{O}(3^M) + 2\mathcal{O}\left(3^M\right) \approx 3.5 \cdot \mathcal{O}\left(3^M\right)$. Inversion complexity in weak self orthogonal basis depends on multiplication complextiy in $GF\left(2^N\right)$

$$\mathcal{C}^K_{inv} \approx 3.5 \cdot \mathcal{C}^K_{mult}(N). \tag{15}$$

4 New Euclidean algorithm

The base of proposed method is the algorithm described in Loan's paper [7]. It is necessary to describe basic operations and subprograms before the algorithm definition. $\odot$ is multiplication of element of extended field by linearized polynomial, $\oplus$ is the sum of two linearized polynomials, $\otimes$ is multiplication of two linearized polynomials [1]. Lets denote $A[x]$ as coefficient of term of degree x. $Norm(DegA, A(z), B(z))$ is polynomial $A(z)$ normalization. $Norm(DegA, A(z), B(z))$ returns the normalization coefficient and multiplies this coefficient by $B(z)$. The function $Check(DegA, B(z))$ checks exit condition and calculates final program result if condition is met. The function $Decr(DegA, DegB, A(z), B(z))$ decrements variable for polynomial power (through one iteration). So describe proposed Euclidean algorithm. It implements next-step multiplication powers of the polynomials $Q(z)$ and $R(z)$ until one of the powers is lower then $\frac{d-1}{2}$ (exit condition).

```
Listing 1 INPUT: $d$ (rank distance), $S(z)$ (syndrome);
OUTPUT: $OUT(z)$ (algorithm result). $Q(z) = z^{[d-1]}$; (start initialization)
1: $R(z) = S(z)$; $\mu(z) = z^{[0]}$; $\lambda(z) = 0$; $DeqQ = d-1$; $DegR = degS(z)$;
2: $DeqStop = \frac{d-1}{2}$; (stop initialization)
3: LOOP IF $DegR < DegQ$ OR
4: ($DegR == DegQ$ AND $R[DegR] == I$ AND $Q[DegQ]! = 0$) THEN
5:     $l = DeqQ - DeqR$;
6:     IF $R[DegR] == 0$ THEN $DeqR = Check(DegR, \lambda(z))$;
7:     ELSE $(R(z), \lambda(z)) = Norm(R(z), \lambda(z))$;
8:         $(Q(z), \mu(z)) = DecDeg(DegQ, DegR, Q(z), R(z), \mu(z), \lambda(z), l)$;
9:         $DegQ = Check(DegQ, \mu(z))$; END IF
10: ELSE $l = DegR - DegQ$;
```

```
11:     IF $Q[DegQ] == 0$ THEN $DegQ = Check(DegQ, \mu(z))$
12:     ELSE $(Q(z), \mu(z)) = Norm(Q(z), \mu(z))$;
13:         $(R(z), \lambda(z)) = DecDeg(DegR, DegQ, R(z), Q(z), \lambda(z), \mu(z), l)$;
14:         $DegR = Check(DegR, \lambda(z))$; END IF END IF END LOOP
```

To evaluate modified Euclidean algorithm complexity it is necessary to consider worst case. In worst case either power of $R(z)$ polynomial are decreased by 2, either power of $Q(z)$ polynomial are decreased by 2. Thus complexity of the pair of iteration (of d) is evaluated as

$$\mathcal{C}_{2iter}(d, N) = \mathcal{C}_{inv}(N) + 3(d-2) \cdot \mathcal{C}_{mult}(N). \tag{16}$$

Maximum number of pair of such iteration until stop condition equals

$$\mathcal{L}(d) = (d-1) - \frac{d-1}{2} = \frac{d-1}{2}. \tag{17}$$

In the view of (16) Euclidean complexity is evaluated

$$\mathcal{C}_E(d, N) = \frac{d-1}{2}\mathcal{C}_{inv}(N) + \left(\frac{3}{2}(d-1)(d-2)\right)\mathcal{C}_{mult}(N). \tag{18}$$

Using (15) modify (18)

$$\mathcal{C}_E^K(d, N) = \frac{3}{2}(d-1)\left(d + \frac{1}{2}\right) \cdot \mathcal{C}_{mult}^K(N). \tag{19}$$

In the case of maximum distance rank code ($d = n - k + 1$; $N = n$; $k = n/2$; $d = N/2 + 1$; $t(error\ count) = d/2$) got the following estimates. A Welch-Berlekamp like algorithm [8] has the complexity

$$\mathcal{C}_{WB}(N) \approx \left(\frac{5}{2}N^2 - \frac{3}{8}N^2 + \frac{N}{4}\right)C_{mult}(N) \approx \mathcal{O}(N^{3.585}). \tag{20}$$

A proposed algorithm complexity is evaluated

$$\mathcal{C}_{E_{prop}}^K(N) \approx N^2\mathcal{O}\left(N^{\log_2 3}\right) \approx \mathcal{O}\left(N^{log_2 12}\right) \approx \mathcal{O}\left(N^{3.585}\right). \tag{21}$$

So one can make a conclusion that proposed algorithm complexity in the term of base operations compares well with Welch-Berlekamp algorithm. With additional multiplier given algorithm decreases data processing delay by 2 times.

A fast equivalent of the Extended Euclidean Algorithm for linearized polynomials (LEEA) [9] has the following complexity (provided it's based om Karatsuba-Ofman multiplication algorithm)

$$\mathcal{C}_{LEEA}(d, N) = \mathcal{O}(d^{1.69} \log d \cdot \mathcal{C}_{mult}^K(N)) = \mathcal{O}(N^{3.275} \log N), \tag{22}$$

that is better than proposed algorithm ($\log N$ multiplicator versus $N^{0.31}$). But LEEA has recursive structure and is not suitable for hardware implementations.

5 Conclusion

An optimal Euclidean algorithm for linearized polynomial have been proposed in this paper. One multiplier, one invertor and 2 multiplexors are required for its implementation. In the case of additional multiplier, data processing delay may be decreased by 2 times. Under the condition that d is predetermined (21), asymptotically complexity in the term of base operations equals $\mathcal{C}^{K}_{E_{prop}}(N) \approx \mathcal{O}\left(N^{3.585}\right)$. The algorithm complexity compares well with Welch-Berlekamp algorithm (20). In spite of it has more complexity in comparison with LEEA, its scheme is not recursive and suitable for hardware implementation. Also proposed algorithm is useful when syndrome calculator is based on weak self orthogonal bases.

References

[1] E. M. Gabidulin, Theory of Codes with Maximum Rank Distance, *Probl. Peredachi Inf.*, 1985, **21** (1), 3–16.

[2] G. Richter and S. Plass, Error and erasure decoding of rank-codes with a modified Berlekamp-Massey algorithm, *Proc. ITG Conf. on Source and Channel Coding*, Erlangen, Germany, Jan. 2004, 249–256.

[3] A. Wachter, V. B. Afanassiev and V. R. Sidorenko, Fast Decoding of Gabidulin Codes, *Int. workshop Coding Cryptorg (WCC)*, Paris, France, Apr. 2011, pp. 433–442.

[4] D. Silva, F. R. Kschischang, Fast encoding and decoding of Gabidulin codes, *Proc. of IEEE ISIT*, 2009.

[5] E. M. Gabidulin and N. I. Pilipchuk, Symmetric matrices and codes correcting rank errors beyond the $\lfloor (d-1)/2 \rfloor$ bound, *Disrete Applied Mathematics*, **154** (2), pp. 305–312.

[6] E. M. Gabidulin and S. Y. Sysoev, Rank codes using weak self-orthogonal bases, *Proc. IEEE Region 8 International Conference*, 2010, pp. 70–71.

[7] S. A. Loan, A Novel VLSI Architecture for Euclid Algorithm, *Journal of Active and Passive Electronic Devices*, USA, 2008, V. 3, pp. 281–299.

[8] P. Loidreau, A Welch-Berlekamp Like Algorithm for Decoding Gabidulin Codes, *Coding and Cryptography*, 2005, **36**.

[9] A. Wachter, V. Sidorenko and M. Bossert, A Fast Linearized Euclidean Algorithm for Decoding Gabidulin Codes, *Proc. of Twelfth International Workshop on Algebraic and Combinatorial Coding Theory (ACCT 2010)*, September 2010, Novosibirsk, Russia.