

GRBNER BASIS WITH QUEUE APPROACH TO SOLVE WELCH-BERLEKAMP EQUATION

Pongstorn Maidee Somsak Choomchuay
Department of Electronics, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Chalongkrung Road, Ladkrabang, Bangkok 10520, Thailand.

Abstract

The better way to decode RS code, there is not only the hard-decision technique but also soft-decision decoding. Given by Welch-Berlekamp algorithm; the soft-decision gives one-run generalized minimum distance decoding. The additional feature that can further avoid unnecessary computation is elaborated in this paper. The new approach is easier to understand. We show that if we found the satisfied position set to the Welch-Berlekamp equation at each stage, we need to compute only one element of its set, all the left can automatically go to the next stage. At the stage that has no position left unless in the defined set, we need not to work further. This yields better computation performance. The hardware requirement applied to the RS code with moderate average error correctivity can be designed accordingly.

1. Introduction

In RS code system, to achieve better computational work load, the remainder polynomial approach can be used instead of syndrome computing. Subsequently, the method of finding the error location of the corresponding key equation is generalized in the Welch-Berlekamp algorithm[1]. Extend to Patrick Fitzpatrick's work[2], S.M. Jenning[3] had shown the similar algorithm based on the $Gr_{\mathfrak{b}}$ bner basis. The lately modification to WB algorithm was proposed by using the queue technique by Berlekamp[4].

In this paper we show an alternative approach to WB algorithm given in [4] but based on the $Gr_{\mathfrak{b}}$ bner basis and queue. Compared to [4], the proposed technique is more intuitive while the computation count is comparable. The new approach is also more systematic and gives us the chance of future improvement.

The lemma that verifies the improvement is shown in the appendix A and the overall algorithm is demonstrated in the appendix B for clearly understanding.

2. Welch-Berlekamp Equation

An alternative Welch-Berlekamp Equation derived in [1] can be shown as

$$N(\alpha^k) \equiv r_k \gamma_k W(\alpha^k) \quad : \deg(N) < \deg(W) = e \leq 2t/2. \quad (1)$$

Interpolation polynomial can then be obtained by interpolating x_i and y_i , so that $y_i = h(x_i)$ for $i = 1, \dots, 2t$. After we replace x_k, y_k with α^k and $r_k \gamma_k$, respectively, (1) can be rewritten as

$$N(x) = h(x).W(x) \pmod{g(x)} \quad (2)$$

3. $Gr_{\mathfrak{b}}$ bner bases in $F[x]$ to solve WB equation[2,3]

Any pair of polynomials in $F[x] \times F[x]$, (a, b) , can be written in the combination form of $\{(x^i, 0), (0, x^j) : i, j \geq 0\}$. They can be compared to each other by mean of the fixed term order $<$. The greatest one is called the *leading term* of (a, b) and denoted as $Lt(a, b)$. For a given submodule M of $F[x] \times F[x]$ generated by basis $\beta =$

$\{(a_1, b_1), (a_2, b_2)\}$ with $Lt(a_1, b_1) = (x^p, 0)$ and $Lt(a_2, b_2) = (0, x^q)$ for any integer p and q , then β is the $Gr_{\mathfrak{b}}$ bner basis, of which the less is known as the *minimal element*. Part of generator polynomial $g(x)$ given in (2) can be defined as

$$G_k(x) = \prod_{i=0}^{k-1} (x - x_i) ; k = 1, \dots, 2t \text{ and } G_0(x) = 1. \quad (3)$$

Let $M_k = \{(a, b) : a \equiv bh^k \pmod{G_k}\}$ with $k = 0, 1, \dots, 2t$ be a partial solution set of (2). To find the actual solution of (2), iterative method started with submodule M_0 is used. Since $(a - bh^k)$ is multiple of G_k , where h^k is $h(x) \pmod{G_k}$. Then $(a - bh^k)(x)$ can be written in the form

$$(a - bh^k)(x) = q(x)G_{2t}(x) + c_{2t-1}G_{2t-1}(x) + \dots + c_k G_k(x). \quad (4)$$

Note that $q(x)$ can be any polynomial in $F[x]$. Relative to G_{k+1} , (a, b) can be decomposed to get $(a, b) = b(h^{k+1}, 1) + m(G_{k+1}, 0) + (c_k G_k, 0)$ by division technique[2], for any polynomial, m , in $F[x]$. It is clear that $(a, b) \in M_{k+1}$, if the term $(c_k G_k, 0)$ is canceled.

The conditions to find the $Gr_{\mathfrak{b}}$ bner basis $\{(a'_1, b'_1), (a'_2, b'_2)\}$ belonging to M_{k+1} , for the known $Gr_{\mathfrak{b}}$ bner basis $\{(a_1, b_1), (a_2, b_2)\}$ of M_k are summarised below. To maintain the degree constraint according to (1), (a_1, b_1) are chosen to be a minimal elements relative to a fixed term order $<_2$.

Condition 1 If $(a_1, b_1) \in M_{k+1}$ then
 $(a'_1, b'_1) = (a_1, b_1)$ and $(a'_2, b'_2) = (x - x_k)(a_2, b_2)$.

So that $(a'_2, b'_2) \in M_{k+1}$.

Condition 2 If $(a_1, b_1) \notin M_{k+1}$ then
 $(a'_1, b'_1) = (x - x_k)(a_1, b_1)$,

and $(a'_2, b'_2) = (a_2, b_2) - \frac{d_k}{c_k}(a_1, b_1)$,

where c_k and d_k are derived from (4),

$$c_k = \frac{(a_1 - b_1 h^k)(x_k)}{G_k(x_k)},$$

and $d_k = \frac{(a_2 - b_2 h^k)(x_k)}{G_k(x_k)}$.

Condition 3 For $(a_1, b_1) \notin M_{k+1}$, (a_2, b_2) is the adjacent higher leading term to (a_1, b_1) . (a'_1, b'_1) is no longer

the minimal element according to the condition 2. Then it is required to swap those two terms to get

$$(a_1', b_1') = (a_2, b_2) - \frac{d_k}{c_k}(a_1, b_1),$$

and $(a_2', b_2') = (x-x_k)(a_1, b_1).$

4. The Modified Algorithm

In section 3, one can notice that a zero denoted hereby as x_i is added to $G_k(x)$ once x_i and y_i have entered corresponding to (3). In our modification, detailed below, we will choose the proper zero from a list and add to $G_k(x)$. To do this we need to rewrite (3) as

$$G_k(x) = \prod_{i \in R_k} (x - x_i), \quad (5)$$

where $R_k = \{i : i \in \{1, \dots, 2t\}, |R_k| = k, \text{ if } i, j \in R_k \text{ then } i \neq j\}$.

For given $Gr_{\mathfrak{b}}$ bner basis belong to M_k of which (a_1, b_1) is the minimal element. We also need to define a set Q that collects indexes of which position resulting c_k to 0.

$$Q = \{j : c_k = \frac{(a_1 - b_1 h^k)(x_j)}{G_k(x_j)} = 0, j \notin R_k\}. \quad (6)$$

By the definition of Q given in (6), we automatically get $(a_1, b_1) \in M_{k+1}$, when we add the zero with the position corresponding to member of Q into G_k , according to the condition 1 in the previous section. On the other hand, if we add other zero position outside Q to $G_k(x)$, i.e. position j , according to condition 3, defer the element in Q , we can always have (a_1, b_1) lower adjacent to (a_2, b_2) . We also get $(a_1, b_1), (a_2, b_2) \in M_{k+1}$, with $R_{k+1} = R_k + j$ and $j \notin Q, R_k$, by

$$(a_1', b_1') = (a_2, b_2) - \frac{d_k}{c_k}(a_1, b_1), \quad (7.1)$$

$$(a_2', b_2') = (x-x_j)(a_1, b_1). \quad (7.2)$$

If we add one zero according to element in Q into G_{k+1} , we then get

$$c_{k+1} = \frac{(a_1' - b_1' h^{k+1})(x_{j_q})}{G_{k+1}(x_{j_q})} ; j_q \in Q. \quad (8)$$

We can proof that c_{k+1} never be 0. It means that adding one more zero index from Q never result that the member of M_{k+1} to be member of M_{k+2} . We also have d_{k+1} corresponding to (8) which is always zero. According to the condition 3 in section 3,

$$(a_1'', b_1'') = (a_2', b_2'), \quad (9.1)$$

$$(a_2'', b_2'') = (x-x_{j_q})(a_1', b_1') ; j_q \in Q, \quad (9.2)$$

are the member of M_{k+2} . In addition, (a_1'', b_1'') can be further decomposed into the form similar to (4). Thus

$$c_{k+2} = \frac{(a_1'' - b_1'' h^{k+2})(x_j)}{G_{k+2}(x_j)} ; j \notin R_{k+2}.$$

With further reduction

$$c_{k+2} = \frac{(x_j - x_i)(a_1 - b_1 h^k)(x_j)}{(x_j - x_i)(x_j - x_{j_q})G_k(x_j)} ; j \notin R_{k+2}, j_q \in Q \text{ and } R_{k+2}, i \in R_{k+1}.$$

Obviously, $c_{k+2} = 0$ when j belongs to Q but not j_q . Every zero corresponding to the indexes in Q will take (a_1'', b_1'') into M_{k+3} . Therefore, Q can be re-defined as

$$Q = \{j : c_{k+2} = \frac{(a_1'' - b_1'' h^{k+2})(x_j)}{G_{k+2}(x_j)} = 0, j \notin R_{k+2}\}, \quad (10)$$

and former set Q are covered.

With the same procedure, $Gr_{\mathfrak{b}}$ bner basis that belongs to M_{k+5} can be easily obtained. In brief, we add zero that not correspond to the new defined Q to (a_1'', b_1'') according to condition 3, then (a_1'', b_1'') belongs to M_{k+3} and after adding one more zero from Q . (a_1^3, b_1^3) can be taken into M_{k+4} . It is clear that all left zero in Q will take $(a_1^4, b_1^4) \in M_{k+4}$ into M_{k+5} . We can proof this if all the left elements in the $2t$ location list are transferred to Q . There is no need to update the $Gr_{\mathfrak{b}}$ bner basis since it is already belong to M_{2t} , as shown in lemma 1 (see appendix A).

5. Conclusion

Detailed in the previous sections and with the support of the algorithm given in appendix B, the elements in Q containing zeroes take the basis in M_k to M_{k+1} will become ones that take the basis in M_{k+2} to M_{k+3} after adding one zero outside Q . The elements in Q are decreased one each. In general, all the left elements in the set Q after we have tested all the list are the ones which no further needed for the computed of the basis's member of M_{2t} . Theoretically, we are free to choose any elements in Q . However, the firstly arrived one is preferable. We can call the set Q as a kind of particular queue. One may notice that the computation count of this approach is depended on the amount of error which can vary. This results in the improvement of the average decoding speed. To smoothen the data flow, input and output buffers can be added.

References

- [1] M. Morii and M. Kasahara, "Generalized Key-Equation of Remainder Decoding Algorithm for Reed-Solomon Codes," *IEEE Trans. Inform. Theory*, vol. 38, No.6, Nov. 1992, p.1801-1807.
- [2] P. Fitzpatrick, "On the Key Equation," *IEEE Trans. Inform. Theory*, vol.41, No.5, Sep. 1995, p.1290-1302.
- [3] S.M. Jennings, "Gr_b bner basis view of Welch-Berlekamp algorithm for Reed-Solomon codes," *IEE Proc.-Commun.*, vol.142, No. 6, Dec. 1995, p. 349-351.
- [4] E.R. Berlekamp, "Bounded Distance+1 Soft-Decision Reed-Solomon Decoding," *IEEE Trans. Inform. Theory*, vol.42, No.3, May 1996, p. 704-720.

Appendix A

Lemma 1 For the $Gr_{\mathfrak{b}}$ bner basis belongs to M_k which (a, b) is its minimal element, together with the n element set Q defined in (6). Then $(a, b) \in M_k, M_{k+1}, \dots, M_{k+n}$

while zeros that have their indexes in a set Q into G_k is added, one at a time, finally all.

Proof

If we add i zeros with their indexes in Q into G_k , we get

$$G_{k+i} = G_k \prod_{u \in v} (x - x_u) \quad ; v \subset Q, |v| = i,$$

Suppose that $(a, b) \in M_{k+i}$, similar to (4) we have

$$c_{k+i} = \frac{(a - bh^{k+i})(x_j)}{G_{k+i}(x_j)} \quad ; j \notin R_{k+i}$$

According to condition 1, we get

$$c_{k+i} = \frac{(a - bh^k)(x_j)}{G_k(x_j) \prod_{u \in v} (x_j - x_u)} \quad ; v \subset Q, |v| = i$$

With the constrain that $j \notin R_{k+i}$ and $Q, v \subset Q, j \notin v$. It is clear that c_{k+i} shrinks to 0 by the definition of Q given by (6) if j is also the element of Q . Since the number of element in Q is n then the maximum value of i that take c_{k+i} to 0 is $|Q|-1 = n-1$, correspond to $c_{k+n-1} = 0$. So that $(a, b) \in M_{k+n}$. \square

Appendix B

Algorithm

```

input :  $(x_i, y_i) ; i = 1, \dots, 2t$ 
initialize:  $(a_1, b_1) = (1, 0); (a_2, b_2) = (0, 1);$ 
            $k=1; q_{\text{index}} = 0;$ 
WHILE (  $k < 2t+1$ ) DO
   $c = a_1(x_k) - b_1(x_k) \cdot y_k$ 
  IF (  $c = 0$  ) THEN  $q[q_{\text{index}}=q_{\text{index}}+1] = x_k$ 
  ELSE  $d = a_2(x_k) - b_2(x_k) \cdot y_k$ 
     $(a_1, b_1) = (a_2, b_2) - \frac{d}{c} (a_1, b_1)$ 
     $(a_2, b_2) = (x - x_k)(a_1, b_1)$ 
    IF ( $q_{\text{index}} > 0$ ) THEN  $x_k = q[q_{\text{index}}=q_{\text{index}}-1]$ 
     $(a_1, b_1) = (a_2, b_2)$ 
     $(a_2, b_2) = (x - x_k)(a_1, b_1)$ 
  END IF
END IF
END WHILE

```