Abstract—In this letter, Zech logarithmic decoding method is proposed for triple-error-correcting binary cyclic codes, which have not been developed before, whose generator polynomials have at most three irreducible factors.

Index Terms—Cyclic codes, Generator polynomial, Zech logarithm.

I. INTRODUCTION

THE knowledge of finite fields is a very useful tool for constructing and decoding multiple-error-correcting cyclic codes. In 1849, Zech published a table of the addition logarithms that all field elements were represented in the logarithm domain. Such a logarithm is called Zech logarithm, which allows for solving equations over finite fields [1]-[2] and decoding quadratic residue codes [3]-[4]. In particular, a decoding method to use Zech logarithms [5] was proposed for binary two-error-correcting cyclic codes whose generator polynomials have at most two irreducible factors. However, no attempt has been made here to develop Zech logarithmic decoder more than two errors.

In this letter, Zech logarithmic decoding method is proposed for triple-error-correcting binary cyclic codes, which have not been developed before. Three decoding algorithms for triple-error-correcting cyclic codes whose generator polynomials have distinct irreducible factors are given. All algorithms have been verified by software simulation using C++ language. An example demonstrates the decoding algorithms validly.

The notations are first introduced throughout this letter. Let \( n \) be a odd number of the form \( n = 1 \mod 2 \). If \( m \) is the smallest positive integer such that \( n \) divides \( 2^m - 1 \) and \( \alpha \) is a primitive element of the finite field \( GF(2^m) \), then the element \( \beta = \alpha^q \) with \( q = (2^m - 1)/n \) is a primitive \( n \)th root of unity in \( GF(2^m) \). In a \((n, k)\) binary cyclic code \( C \), the generator polynomial \( g(x) \) of degree \( n - k \) over \( GF(2) \) is a divisor of \( x^n - 1 \) and is also expressed as the product of the distinct polynomials \( m_i(x) \), where \( m_i(x) \) is the minimal polynomial of \( \beta^i \) over \( GF(2) \). A codeword of \( C \) is a binary vector \( \mathbf{c} = (c_0, c_1, ..., c_{n-1}) \) so that its associated polynomial \( c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \) is a multiple of \( g(x) \). If the codeword \( \mathbf{c} \) is transmitted through a noisy communication channel, and if the vector \( \mathbf{r} = (r_0, r_1, ..., r_{n-1}) \) is received, then the polynomial \( r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1} \) corresponding to \( \mathbf{r} \) can be expressed as a sum of the code polynomial \( c(x) \) and the error polynomial \( e(x) = e_0 + e_1 x + \cdots + e_{n-1} x^{n-1} \). Let \( Z(u) \) denote the Zech logarithm of the form

\[
Z(u) = \log_{\alpha}(1 + u^w),
\]

where \( u \in \{0, 1, ..., 2^m - 2\} \cup \{\infty\}. \) The arithmetic operations on \( \{0, 1, ..., 2^m - 2\} \cup \{\infty\} \) are defined in the obvious way as follows: \( 1 + \alpha^{u} = \alpha^{2^{j}u} \), \( Z(\infty) = 0 \), and \( Z(0) = \infty \). For more details we refer the reader to [6].

Let \( wt(f(x)) \) denote the weight of \( f(x) \) which is number of nonzero coefficients in the polynomial \( f(x) \). For a cyclic code with minimum distance \( d \), an error polynomial \( e(x) \) is said to be correctable if its weight less than or equal to the error-correcting capacity, \( t = [(d - 1)/2] \), where \( [l] \) denotes the greatest integer less than or equal to \( l \). If the minimum distance of an arbitrary cyclic code is larger than or equal to \( 7 \), then the following lemma holds.

Lemma 1: There is exactly one correctable error polynomial \( e(x) \) determined by its syndromes if \( wt(e(x)) \leq 3 \).

Proof: This follows from Theorem 1 in [7].

Consider a triple-error-correcting binary cyclic code \( C \) whose the length is \( n \) and its generator polynomial has three irreducible factors, i.e., \( g(x) = m_i(x)m_j(x)m_k(x) \), where the numbers \( i, j, k, n \) must be mutually relatively prime. We follow the assumption in [5] that if \( b \) is a common divisor of \( i, j, k \), then \( b > 1 \), one may select \( i', j', k' \) making \( i' + j' + k' = 1 \mod (n) \). Then Lemma 2 below is required for deriving the decoding algorithms as shown in the next section.

Lemma 2: Let \( e(x) \) be the error polynomial and let \( I = \log_{\alpha}(e(\beta^i)), J = \log_{\alpha}(e(\beta^j)), K = \log_{\alpha}(e(\beta^k)) \) be the syndromes. If \( wt(e(x)) \leq 3 \), then

1) \( e(x) = 0 \) if and only if \( I = J = K = \infty \).
2) \( e(x) = x^h \) for some \( h \) if and only if
   (i) None of \( I, J, K \) is \( \infty \).
   (ii) \( q[I, q]J, q[K] \), and
   (iii) \( jk(Z(I)/q) = ik(Z(J)/q) = ij(Z(K)/q) \mod n \), where
         \[
         h \equiv i'((I)/q) + j'(J)/q) + k'(K)/q) \mod n. \]
3) \( e(x) = 1 + x^h \) for some \( 1 \leq h < n \) if and only if
   (i) None of \( I, J, K \) is \( 0 \) and \( \infty \).
   (ii) \( q[Z(I)], q[Z(J)], q[Z(K)] \), and
   (iii) \( jk(Z(I)/q) = ik(Z(J)/q) = ij(Z(K)/q) \mod n \), where
         \[
         h \equiv i'(Z(I)/q) + j'(Z(J)/q) + k'(Z(K)/q) \mod n. \]
4) \( e(x) = 1 + x^w(1 + x^h) \) for some \( 1 \leq w < n, 1 \leq h < n \) if and only if
   (i) None of \( I, J, K \) is \( \infty \).
   (ii) \( Z(I) - qiw \neq 0, \infty, Z(J) - qjw \neq 0, \infty \),

C.-D. Lee is with the Department of Communication Engineering, I-Shou University, Dashu Township, Kaohsiung Country 84001, Taiwan, R.O.C. (e-mail: chongdao@isu.edu.tw).
there are two errors at the locations (iv) 

Z(K) − qkw \neq 0, \infty (\text{mod } 2^m − 1),

(iii) \(q(Z(I) − qiw), q(Z(J) − jqw), q(Z(K) − kqw),\)

(iv) \(jk(Z(I) − qiw)/q) \equiv ik(Z(J) − jqw)/q,\)

\[ h \equiv i'(Z(I) − qiw)/q \]

\[ + j'(Z(J) − jqw)/q \]

\[ + k'(Z(K) − kqw)/q) \text{ (mod } n). \] (4)

**Proof:** The proof of (1) (respectively 2), 3)) is analogous to the proof of (0) (respectively 1), 2)) in Lemma 2 of [5] by the congruence \(i'i + j'j + k'k = 1 \text{ (mod } n).\) It remains to prove (4).

If \(e(x) = 1 + x^w(1 + x^h),\) then \(I \equiv Z(Q(h) + qiw)\) and thus \(Z(I) \equiv Z(qih) + qiw.\) Subtracting \(qiw\) to both sides yields \(Z(I) − qiw \equiv Z(qih).\) Finally, we obtain \(Z(I) − qiw \equiv qiw\) by Zech logarithms. Similar arguments hold for \(J\) and \(K.\)

It is the above congruences that make the conditions from (i) to (iv) allowable.

On the other hand, according to Lemma 1 and the congruence \(i'i + j'j + k'k = 1 \text{ (mod } n),\) we have only one correctable error polynomial \(e(x) = 1 + x^w(1 + x^h)\) that follows by a similar proof in Lemma 2 of [5]. The proof of (4) is complete. Lemma 2 is proved.

II. Decoding Algorithm

This section consists of three subsections. Section III-A (respectively III-B, III-C) describes Zech logarithmic decoding algorithm for triple-error-correcting binary cyclic codes whose generator polynomials have three (respectively two, one) irreducible factors.

A. Three Factors

Let \(g(x) = m_1(x)m_2(x)m_3(x)\) and \(i'i + j'j + k'k = 1 \text{ (mod } n).\)

**Algorithm I:**

1) Compute \(I = \log_\alpha r(\beta^i), J = \log_\alpha r(\beta^j)\) and \(K = \log_\alpha r(\beta^k).\)

2) If \(I = J = K = \infty,\) then \(r = e.\)

3) If (i) \(I \neq \infty, J \neq \infty, K \neq \infty,\)

(ii) \(q[I, qJ, qK],\) and

(iii) \(jk(I)/q) \equiv ik(J)/q) \equiv ij(K)/q) \text{ (mod } n),\) then there is a single error at the location \(h, i.e., e(x) = x^h,\)

\[ h \equiv i'(I)/q) + j'(J)/q) + k'(K)/q) \text{ (mod } n). \] (5)

4) If \(0 \leq p \leq n − 1\) and

(i) \(I \neq \infty, J \neq \infty, K \neq \infty,\)

(ii) \(I - qjp \neq 0, J - qjp \neq 0, K - qkp \neq 0 \text{ (mod } 2^m - 1),\)

(iii) \(q[Z(I) - qip), q[Z(J) - jqp), q(Z(K) - kqp),\) and

(iv) \(jk(Z(I) - qip)/q) \equiv ik(Z(J) - jqp)/q,\)

\[ \equiv ij(Z(K) - kqp)/q) \text{ (mod } n),\) then there are two errors at the locations \(p \) and \(p + h, i.e., e(x) = x^p(1 + x^h),\)

\[ h \equiv i'(Z(I) - qip)/q) \]

\[ + j'(Z(J) - jqp)/q) \]

\[ + k'(Z(K) - kqp)/q) \text{ (mod } n). \] (6)

5) If \(0 \leq p \leq n - 2, 1 \leq w \leq n - 2\) and

(i) \(I \neq \infty, J \neq \infty, K \neq \infty,\)

(ii) \(I - qip \neq 0, J - jqp \neq 0, K - qkp \neq 0 \text{ (mod } 2^m - 1),\)

(iii) \(I' - qiqp \neq 0, J' - jqwp \neq 0, K' - qkw \neq 0 \text{ (mod } 2^m - 1),\)

(iv) \(q[Z(I') - qiw), q[Z(J') - jqw), q(Z(K') - kqw),\) and

(v) \(jk(Z(I') - qiw)/q) \equiv ik(Z(J') - jqw)/q,\)

\[ \equiv ij(Z(K') - kqw)/q) \text{ (mod } n),\) then there are three errors at the locations \(p, p + w\) and \(p + w + h, i.e., e(x) = x^p(1 + x^w(1 + x^h)),\)

\[ h \equiv i'(Z(I') - qiw)/q) \]

\[ + j'(Z(J') - jqw)/q) \]

\[ + k'(Z(K') - kqw)/q) \text{ (mod } n). \] (7)

**Example 1:** Let \(\beta = \alpha^5\) be a primitive 51st root of unity in \(GF(2^8),\) where \(\alpha\) is a root of the primitive polynomial \(1 + x^2 + x^3 + x^4 + x^8.\) The generator polynomial of (51, 27, 8) cyclic code is the product of the minimal polynomials \(m_1(x), m_3(x)\) and \(m_4(x), i.e., g(x) = m_1(x)m_3(x)m_4(x) = 1 + x^4 + x^6 + x^8 + x^{10} + x^{11} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16} + x^{18} + x^{19} + x^{20} + x^{22} + x^{24}.\)

We take \(i = 1, j = k = 9, i' = -2, j' = -2,\) and \(k' = 1\) that have satisfied the assumption \(i'i + j'j + k'k = 1 \text{ (mod } 51).\) Now, if \(e(x) = x + x^3 + x^6,\) then the decoding procedures are as follows. First, we calculate

\[ r(\beta) = e(\beta) = \beta + \beta^3 + \beta^6 = \alpha^{126}, \]

\[ r(\beta^3) = e(\beta^3) = \beta^3 + (\beta^3)^3 + (\beta^3)^6 = \alpha^{201}, \]

\[ r(\beta^9) = e(\beta^9) = \beta^9 + (\beta^3)^9 + (\beta^3)^9 = \alpha^{34}. \]

Further, we have \(I = \log_\alpha r(\beta) = 126, J = \log_\alpha r(\beta^3) = 201,\)

and \(K = \log_\alpha r(\beta^9) = 34.\) It is clear that steps 2) and 3) skip since \(I\) is not equal to \(\infty\) and is not divided by \(q = 5,\) respectively. A full computer search shows that, for \(0 \leq p \leq 50,\) step 4) also skips because the conditions of (iii) and (iv) are always not sufficed. We continue in step 5) of this algorithm obtaining Table I. As you can see, this algorithm stops when \(p = 1 \) and \(w = 2;\) furthermore, the congruence \(h \text{ in (7)} \) becomes \(h \equiv (-2) \cdot 3 + (-2) \cdot 9 + 1 \cdot 27 = 3.\)

Finally, these values imply that \(e(x) = x^p(1 + x^w(1 + x^h)) = x(1 + x^2(1 + x^3)) = x + x^3 + x^6,\) as required.

<table>
<thead>
<tr>
<th>(p)</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w)</td>
<td>1</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>(I - qip)</td>
<td>126</td>
<td>...</td>
<td>126</td>
</tr>
<tr>
<td>(J - jqp)</td>
<td>201</td>
<td>...</td>
<td>201</td>
</tr>
<tr>
<td>(K - qkp)</td>
<td>34</td>
<td>...</td>
<td>34</td>
</tr>
<tr>
<td>(I' = Z(I - qip))</td>
<td>110</td>
<td>...</td>
<td>110</td>
</tr>
<tr>
<td>(J' = Z(J - jqp))</td>
<td>154</td>
<td>...</td>
<td>154</td>
</tr>
<tr>
<td>(K' = Z(K - kqp))</td>
<td>136</td>
<td>...</td>
<td>136</td>
</tr>
<tr>
<td>(I' - qiqp)</td>
<td>105</td>
<td>...</td>
<td>115</td>
</tr>
<tr>
<td>(J' - jqwp)</td>
<td>139</td>
<td>...</td>
<td>169</td>
</tr>
<tr>
<td>(K' - qkw)</td>
<td>91</td>
<td>...</td>
<td>181</td>
</tr>
<tr>
<td>(Z(I' - qiw))</td>
<td>248</td>
<td>...</td>
<td>243</td>
</tr>
<tr>
<td>(Z(J' - jqw))</td>
<td>98</td>
<td>...</td>
<td>156</td>
</tr>
<tr>
<td>(Z(K' - kqw))</td>
<td>209</td>
<td>...</td>
<td>29</td>
</tr>
</tbody>
</table>

**TABLE I:**

Decoding of (51, 27, 8) Cyclic Code with Three Errors Using Algorithm I
B. Two Factors

Let \( g(x) = m_j(x)m_k(x) \) and \( j'j + k'k \equiv 1 \) (mod \( n \)).

Algorithm 2:
1) Compute \( J = \log_a r(\beta^j) \) and \( K = \log_a r(\beta^k) \).
2) If \( J = K = \infty \), then \( r = e \).
3) If \( J \neq \infty, K \neq \infty \),
   (i) \( qJ, qK \), and
   (ii) \( k(J/q) \equiv j(K/q) \) (mod \( n \)), then there is a single error
   at the location \( h \), i.e., \( e(x) = x^h \), where
   \[
   h \equiv j'(J/q) + k'(K/q) \pmod{n}.
   \] (8)
4) If \( 0 \leq p \leq n - 1 \) and
   (i) \( J \neq \infty, K \neq \infty \),
   (ii) \( J - qjp \neq 0, K - qkp \neq 0 \) (mod \( 2^m - 1 \)),
   (iii) \( qZ(J - qjp), qZ(K - qkp) \), and
   (iv) \( k(Z(J - qjp)/q) \equiv j(Z(K - qkp)/q) \) (mod \( n \)), then there are two errors at the locations \( p \) and \( p + h \), i.e., \( e(x) = x^p(1 + x^h) \), where
   \[
   h \equiv j'(Z(J - qjp)/q) + k'(Z(K - qkp)/q) \pmod{n}.
   \] (9)
5) If \( 0 \leq p \leq n - 2, 1 \leq w \leq n - 2 \) and
   (i) \( J \neq \infty, K \neq \infty \),
   (ii) \( J - qjp \neq 0, K - qkp \neq 0 \) (mod \( 2^m - 1 \)),
   (iii) \( J' - qjw \neq 0, K' - kw \neq 0 \) (mod \( 2^m - 1 \)),
   (iv) \( qZ(J' - qjw), qZ(K' - kw) \), and
   (v) \( k(Z(J' - qjw)/q) \equiv j(Z(K' - kw)/q) \) (mod \( n \)), then there are three errors at the locations \( p, p + w \) and \( p + w + h \), i.e., \( e(x) = x^{p+w}(1 + x^h) \), where
   \[
   h \equiv j'(Z(J' - qjw)/q) + k'(Z(K' - kw)/q) \pmod{n}.
   \] (10)

C. One Factor

Let \( q(x) = m_k(x) \) and \( k'k \equiv 1 \) (mod \( n \)).

Algorithm 3:
1) Compute \( K = \log_a r(\beta^k) \).
2) If \( K = \infty \), then \( r = e \).
3) If \( K \neq \infty \) and \( qK \), then there is a single error at
   the location \( h \), i.e., \( e(x) = x^h \), where
   \[
   h \equiv k'(K/q) \pmod{n}.
   \] (11)
4) For \( 0 \leq p \leq n - 1 \), if \( K - qkp \neq 0 \) (mod \( 2^m - 1 \)) and
   \( qZ(K - qkp) \), where \( K \neq 0, \infty \), then there are two errors
   at the locations \( p \) and \( p + h \), i.e., \( e(x) = x^p(1 + x^h) \), where
   \[
   h \equiv k'(Z(K - qkp)/q) \pmod{n}.
   \] (12)
5) For \( 0 \leq p \leq n - 2, 1 \leq w \leq n - 2 \), if \( K \neq \infty, K - qkp \neq 0 \) (mod \( 2^m - 1 \)) and \( K' - kw \neq 0 \) (mod \( 2^m - 1 \)) and
   \( qZ(K' - kw) \), then there are three errors at the locations \( p, p + w \) and \( p + w + h \), i.e., \( e(x) = x^p(1 + x^w(1 + x^h)) \), where
   \[
   h \equiv k'(Z(K' - kw)/q) \pmod{n}.
   \] (13)

It is in a software simulation that Algorithm 3 to decode the binary (23, 12, 7) Golay code has been executed successfully
to check every correctable error pattern. Note that Zech logarithmic decoder is considerably faster in computational time
than the algebraic decoder described in [8]-[10] for binary Golay code.

III. Simulation Results

Three algorithms described in the previous section were implemented on a computer that use C++ language. Although
the generator polynomial mentioned in Algorithm 1 is always
three irreducible factors, it is worth pointing out that the
methods usually work even when its generator polynomial
is more than three irreducible factors. Moreover, Zech logarithmic
decoding algorithms developed here can also be used to
correct at most three errors for binary cyclic codes whose
the error-correcting capacity is larger than 3. Similar investigations
for binary two-error-correcting cyclic codes were previously
announced in [11].

It is possible for some cyclic codes that the syndrome values
appear \( \infty \) when the errors occurred. In such cases, Algorithm 1
fails. If one (respectively two) of three syndromes \( I, J, \) and
\( K \) has the value \( \infty \), then Algorithm 2 (respectively 3) executes instead of Algorithm 1.

IV. Conclusion

This letter presents a method of using Zech logarithmic

techniques for decoding of binary cyclic codes. We have actually shown that the results originally developed in [5] can
be extended to decode triple-error-correcting cyclic codes.

REFERENCES

the structure of GF(p^n),” IEEE Trans. Inform. Theory, vol. 38, no. 3,
of (41, 21, 9) quadratic residue code,” IEEE Trans. Inform. Theory,
quadratic residue code,” Mathematical Proc. of the Royal Irish Academy,
codes with zech logarithms,” IEEE Trans. Commun., vol. 40, no. 12,
for the algebraic decoding of cyclic codes,” IEEE Trans. Inform. Theory,
“Algebraic decoding of quadratic residue codes using Berlekamp-Massey
of (71, 36, 11), (79, 40, 15), and (97, 49, 15) quadratic residue codes,”IEEE Trans.
codes—an analysis,” Singapore ICCS’94 Conference Proceedings, vol. 3,