Derivative Descendants of Cyclic Codes and Derivative Decoding

Qin Huang*, Senior Member, IEEE, Bin Zhang

Abstract

This paper defines cyclic and minimal *derivative descendants* (DDs) of an extended cyclic code from the derivative of the Mattson-Solomon polynomials, respectively. First, it demonstrates that the cyclic DDs are the same extended cyclic code. It allows us to perform soft-decision decoding for extended cyclic codes based on their cyclic DDs. Then, it proves that the minimal DDs are equivalent codes. It also allows us to perform soft-decision decoding based on the minimal DDs with permutations. Simulation results show that our proposed derivative decoding can be close to the maximum likelihood decoding for certain extended cyclic codes, including some extended BCH codes.

Index Terms

cyclic codes, Mattson-Solomon polynomial, soft-decision, derivative decoding

I. INTRODUCTION

Cyclic codes, first studied in 1957 [1], form a large class of error-control codes which include many well-known codes, e.g., *Bose-Chaudhuri-Hocquenghem* (BCH) codes, *Reed-Solomon* codes, finite geometry codes, punctured *Reed-Muller* (RM) codes etc. [2]–[5]. Due to the cyclic structure, their encoding and hard-decision decoding can be implemented efficiently. Moreover, their inherent algebraic structure and soft-decision decoding [6]–[15] have always attracted a lot of attention.

This paper starts from the derivative of Mattson-Solomon (MS) polynomials [2], [16]. We define two types of derivative descendants (DDs) of an extended cyclic code C, cyclic DDs and minimal DDs, respectively. The first type is defined as the smallest extended cyclic codes containing the derivatives of all the codewords in C. The second type is defined as the smallest subspaces consisting of the derivatives of all codewords in C.

First, we demonstrate that the cyclic DDs of an extended binary cyclic code in different directions result in the same extended cyclic code. It can be specified by analyzing the exponent set of MS polynomials. Based on their cyclic DDs, we propose a soft-decision derivative decoding algorithm for extended binary cyclic codes. It consists of three steps: calculating log-likelihood ratios (LLRs) of cyclic DDs, decoding cyclic DDs and voting for decision. Simulation results show that the performance of the proposed derivative decoding is close to that of the maximum

Part of this article was presented at GlobeCom 2022. This work was supported by the National Natural Science Foundation of China under Grant 62071026. (Qin Huang and Bin Zhang contributed equally to this work.) (Corresponding author: Qin Huang.)

likelihood decoding (MLD) for cyclic codes, e.g., (64, 45) and (64, 24) extended BCH (eBCH) codes. Besides we conversely introduce cyclic derivative ascendant (DA) of an extended cyclic code C as well as their decoding.

Then, we prove that the minimal DDs of an extended binary cyclic code in different directions are equivalent. Moreover, it reveals that the cyclic shift of a codeword in a minimal DD is a codeword in another minimal DD. As a result, the derivative decoding can be carried out with only one decoder for the minimal DD in one direction and cyclic shifting. Due to the small dimension of minimal DDs, it is attractive to perform derivative decoding based on ordered statistics decoding (OSD) [17]. Simulation results show that the derivative decoding based on the OSD with order-1 can outperform the higher order OSD.

The rest of the paper is organized as follows. Section II gives a brief review of cyclic codes and MS polynomials. In Section III, we define the cyclic DDs and cyclic DAs of extended cyclic codes. Section IV presents the derivative decoding algorithm. In Section V, we define the minimal DDs and present the derivative decoding based on decodings of minimal DDs. Section VI concludes this paper.

II. CYCLIC CODES AND THEIR DECOMPOSITION

A. Cyclic codes and Mattson-Solomon polynomials

A linear code C of length n is *cyclic* if a cyclic shift of any codeword is also a codeword, i.e. whenever $a = [a_i, i \in [n]]$ is in C then so is $[a_{i+1}, i \in [n]]$. Here, $[n] \triangleq \{0, 1, 2, ..., n-1\}$ and subscripts are reduced modulo n.

Let m be a positive integer. A binary cyclic code C of length $n = 2^m - 1$ and dimension $0 < k \le n$ is an ideal in the ring $\mathbb{F}_2[x]/(x^n - 1)$, which is generated by a generator polynomial g(x) with degree n - k such that g(x)divides $x^n - 1$.

Let α denote a primitive element of \mathbb{F}_{2^m} . For a codeword $a = [a_0, a_1, ..., a_{n-1}]$ corresponding to a code polynomial $a(x) = \sum_{i=0}^{n-1} a_i x^i$, the associated Mattson-Solomon polynomial is defined over \mathbb{F}_{2^m} as follows

$$A(z) \triangleq \sum_{j=0}^{n-1} A_j z^j,$$

where

$$A_j = a(\alpha^{-j}) = \sum_{i=0}^{n-1} a_i \alpha^{-ij}$$

The codeword a can be recovered from A(z) by

$$a = [a_i, i \in [n]] = [A(\alpha^i), i \in [n]].$$

The coefficient A_j is fixed to 0 if and only if α^{-j} is a zero of g(x). Moreover, the MS polynomial of the cyclic shift of **a** is $A(\alpha z)$.

We define the *exponent set* of all the MS polynomials associated with C as follows

$$S_{\mathcal{C}} \triangleq \{ j \in [n] : g(\alpha^{-j}) \neq 0 \}.$$
⁽¹⁾

For brevity, we call it the exponent set of C. Please note that its size is the same as the dimension k. We can express C as

$$\mathcal{C} = \{ [A(\alpha^i), i \in [n]] : A(z) = \sum_{j \in S_{\mathcal{C}}} A_j z^j \},$$

where $A_j \in \mathbb{F}_{2^m}$. The conjugacy constraint [4, Ch. 6], i.e. $A_{2j} = A_j^2$ is required to keep $[A(\alpha^i), i \in [n]]$ binary.

The cyclic code C can be extended by adding an overall parity-check bit to each codeword. The overall paritycheck bit of a codeword a is the evaluation of the corresponding MS polynomial at 0, i.e., A(0) [2, Ch. 8]. Therefore, the extended cyclic code of C can be also identified by S_C . We denote 0 in \mathbb{F}_{2^m} by α^{∞} and define $I \triangleq \{\infty\} \cup [n]$. The extended cyclic code C with exponent set S_C can be expressed as

$$\mathcal{C} = \{ [A(\alpha^i), i \in I] : A(z) = \sum_{j \in S_{\mathcal{C}}} A_j z^j \}.$$

Please note that $\alpha^{\infty} \alpha = \alpha^{\infty}$. Thus we make the agreement $\infty + 1 = \infty \mod n$. Then C is an extended cyclic code if whenever $a = [a_i, i \in I]$ is in C then its cyclic shift $[a_{i+1}, i \in I]$ is also in C. In the following, we mainly focus on extended cyclic codes, and may use a and A(z) to denote the codeword interchangeably.

B. Decomposing cyclic codes as a direct sum of minimal cyclic codes

For an integer $s \in [n]$, the cyclotomic coset modulo n containing s is $C_s \triangleq \{s, 2s, 2^2s, ..., 2^{m_s-1}s\}$, where m_s is the smallest positive integer such that $2^{m_s}s = s \mod n$. The smallest entry of C_s is called the *coset representative*. For a subset S of [n], we denote the smallest and the largest element of S by $\min(S)$ and $\max(S)$, respectively. We denote the union of all the cyclotomic cosets which have nonempty intersections with S as $\operatorname{cc}(S) \triangleq \bigcup_{s \in S} C_s$. And we denote the set consisting of all the coset representatives in $\operatorname{cc}(S)$ as $\operatorname{cr}(S) \triangleq \bigcup_{s \in S} \{\min(C_s)\}$.

The extended minimal cyclic code associated with the cyclotomic coset C_s is

$$\mathcal{M}_s = \{ [A(\alpha^i), i \in I] : A(z) = T_{m_s}(A_s z^s)$$
for all $A_s \in \mathbb{F}_{2^{m_s}} \},$

where $T_{m_s}(z)$ is the trace function

$$T_{m_s}(z) \triangleq \sum_{j \in [m_s]} z^{2^j},$$

and $\mathbb{F}_{2^{m_s}}$ is a subfield of \mathbb{F}_{2^m} . It is clear that the exponent set of \mathcal{M}_s is C_s .

An extended cyclic code with exponent set $S_{\mathcal{C}}$ can be expressed as a direct sum of the extended minimal cyclic codes, i.e.,

$$\mathcal{C} = \bigoplus_{s \in \operatorname{cr}(S_{\mathcal{C}})} \mathcal{M}_s$$
$$= \{ [A(\alpha^i), i \in I] : A(z) = \sum_{s \in \operatorname{cr}(S_{\mathcal{C}})} T_{m_s}(A_s z^s)$$
for all $A_s \in \mathbb{F}_{2^{m_s}} \},$

where \bigoplus is the direct sum operator. We call the set $cr(S_C)$ as the *representative set* of S_C . We end this section with the following example.

Example 1. Let α denote a primitive element in \mathbb{F}_{2^4} . Consider the (16,7) extended cyclic code C associated with the generator polynomial $g(x) = 1 + x^4 + x^6 + x^7 + x^8$. The zeros of g(x) are $\alpha^1, \alpha^2, \alpha^4, \alpha^8, \alpha^3, \alpha^6, \alpha^{12}, \alpha^9$. From (1), the exponent set of C is $S_C = \{0, 1, 2, 4, 8, 5, 10\}$. Then C can be identified by the set

$$\{[A(\alpha^i), i\in I]: A(z) = \sum_{j\in S^{\mathcal{C}}} A_j z^j\},$$

where $A_j \in \mathbb{F}_{2^4}$ and satisfies $A_{2j} = A_j^2$. The representative set of S_c is $\{0, 1, 5\}$. Note that $m_0 = 1$, $m_1 = 4$, $m_5 = 2$. Then C can be expressed as

$$\mathcal{C} = \{ [A(\alpha^i), i \in I] : A_0 + T_4(A_1z) + T_2(A_5z^5) \},\$$

where $A_0 \in \mathbb{F}_2$, $A_1 \in \mathbb{F}_{2^4}$, $A_5 \in \mathbb{F}_{2^2}$.

III. CYCLIC DERIVATIVE DESCENDANTS AND ASCENDANTS

This section introduces cyclic DDs and cyclic DAs of extended cyclic codes. Their dimensions and distances are also investigated.

A. Cyclic derivative descendants

Let β be a power of α . Consider a codeword a and its MS polynomial A(z). The derivative of A(z) in the direction β is defined as

$$\Delta_{\beta}A(z) \triangleq A(z+\beta) - A(z).$$
⁽²⁾

With the above definition, we define the cyclic DDs of extended cyclic codes.

Definition 1. For an extended cyclic code C, its cyclic derivative descendant in the direction β denoted by $\mathcal{D}(C, \beta)$ is the extended cyclic code with the smallest dimension which contains

$$\{[\Delta_{\beta}A(\alpha^{i}), i \in I] : A(z) \in \mathcal{C}\}\$$

In the following, we may use $\Delta_{\beta}A(z)$ to denote the vector $[\Delta_{\beta}A(\alpha^i), i \in I]$ if the context is clear. For simplicity, we may call the vector the derivative of a.

In fact, the cyclic DDs in all the directions are the same. To prove this, we start with the extended minimal cyclic codes. For an integer $s \in [n]$, we denote its binary expansion by $\overline{s} = [s_0, s_1, ..., s_{m-1}]$ such that $s = \sum_{j=0}^{m-1} s_j 2^j$. We define the support set of the binary expansion of s as $W_s \triangleq \{j \in [m] : s_j \neq 0\}$. And we say the binary expansion of s' is properly covered by that of s if $W_{s'} \subsetneq W_s$. Let P(s) denote the set consisting of all the nonnegetive intergers whose binary expansion is properly covered by that of s, i.e.

$$P(s) \triangleq \{\sum_{j \in V} 2^j : V \subsetneqq W_s\}.$$
(3)

The exponent set of the cyclic DDs of an extended minimal cyclic code is given by the following lemma.

Lemma 1. Consider the extended minimal cyclic code \mathcal{M}_s . The exponent set of $\mathcal{D}(\mathcal{M}_s,\beta)$ for any β is cc(P(s)).

4



Cyclic derivative descendants: (16, 5) Hadamard code

Fig. 1. The cyclic DD of the (16,7) extended cyclic code is a (16,5) extended cyclic code.

Proof. For any $A(z) \in \mathcal{M}_s$, its derivative in direction β is

$$\Delta_{\beta}A(z) = T_{m_s} \left(A_s (z+\beta)^s \right) - T_{m_s} (A_s z^s)$$
$$= T_{m_s} \left(A_s \left((z+\beta)^s - z^s \right) \right).$$

Note that

$$(z+\beta)^s - z^s = \prod_{j \in W_s} (z^{2^j} + \beta^{2^j}) - \prod_{j \in W_s} z^{2^j}$$
$$= \sum_{V \subsetneq W_s} (z^{\sum_{j \in V} 2^j} \beta^{\sum_{j \in W_s/V} 2^j})$$
$$= \sum_{k \in P(s)} z^k \beta^{s-k}.$$

Then

$$\Delta_{\beta}A(z) = T_{m_s}(A_s \sum_{k \in P(s)} z^k \beta^{s-k}).$$

From the above equation, we see that the exponents of z must be a subset of cc(P(s)). Note that the coefficients of $\Delta_{\beta}A(z)$ must satisfy the conjugacy constraint, because $\Delta_{\beta}A(\alpha^i) = A(\alpha^i + \beta) - A(\alpha^i)$, and $A(\alpha^i)$ and $A(\alpha^i + \beta)$ are in \mathbb{F}_2 for all $i \in I$. Therefore we can write $\Delta_{\beta}A(z)$ in the form

$$\Delta_{\beta} A(z) = \sum_{s' \in \operatorname{cr} \left(P(s) \right)} T_{m_{s'}}(A'_{s'} z^{s'})$$

where

$$A'_{s'} = \sum_{\substack{i \in [m_s], k \in P(s), \\ k2^i [\mod n] = s'}} \beta^{(s-k)2^i} A_s^{2^i}$$

Treat $A'_{s'}$ as a function of A_s , i.e. $A'_{s'}(A_s)$. Note that the degree of $A'_{s'}(A_s)$ is at most 2^{m_s-1} which implies $A'_{s'}(A_s)$ has at most 2^{m_s-1} roots. Therefore, $A'_{s'}$ is not always zero. As a result, the representative set of the exponent set of $\mathcal{D}(\mathcal{C},\beta)$ is exactly $\operatorname{cr}(P(s))$ while the exponent set is $\operatorname{cc}(P(s))$.

The above lemma shows that the cyclic DDs of an extended minimal cyclic code in different directions are the same. Using the fact that the extended cyclic code C is a direct sum of extended minimal cyclic codes, we obtain the following theorem immediately.

Theorem 1. For an extended cyclic code C with exponent set S_C , its cyclic DDs in different directions are the same code denoted by $\mathcal{D}(C)$, whose exponent set S_D is $\bigcup_{s \in cr(S_C)} cc(P(s))$ and the corresponding representative set is $\bigcup_{s \in cr(S_C)} cr(P(s))$.

Example 2. Continuation of Example 1. The representative set of the exponent set of the (16,7) extended cyclic code C is $\{0, 1, 5\}$. According to (3), $P(0) = \emptyset$, $P(1) = \{0\}$, $P(5) = \{0, 1, 4\}$. From Theorem 1, we conclude that the cyclic DD of C denoted by $\mathcal{D}(C)$ is the (16,5) extended cyclic code with the exponent set $S_{\mathcal{D}} = \{0, 1, 2, 4, 8\}$, *i.e.*,

$$\mathcal{D}(\mathcal{C}) = \{ [A(\alpha^{i}), i \in I] : A(z) = A_0 + T_4(A_1 z) \}$$

It shows that $\mathcal{D}(\mathcal{C})$ is the (16,5) Hadamard code. For code codeword $\boldsymbol{a} = [A(\alpha^i), i \in I] \in \mathcal{C}$, all its derivatives,

$$\begin{split} & [\Delta_{\alpha^0} A(\alpha^i), i \in I] \\ & [\Delta_{\alpha^1} A(\alpha^i), i \in I] \\ & \dots \\ & [\Delta_{\alpha^{14}} A(\alpha^i), i \in I], \end{split}$$

are codewords in $\mathcal{D}(\mathcal{C})$ as illustrated in Fig. 1.

For any nontrivial extended binary cyclic code C, i.e. $S_C \neq \{0\}$, we give the following propositions to characaterize the dimension and distance of their cyclic DDs.

For any binary vector v, we denote its Hamming weight by wt(v). For a subset S of [n], we define $deg(S) \triangleq max(\bigcup_{s \in S} \{wt(\overline{s})\})$. Let d denote the minimum Hamming distance of C. And let k_D and d_D denote the dimension and the minimum Hamming distance of D(C), respectively.

Proposition 1. $k_{\mathcal{D}} \leq \sum_{i=0}^{\deg(S_{\mathcal{C}})-1} \binom{m}{i}.$

Proof. The dimension of $\mathcal{D}(\mathcal{C})$ satisfies

$$k_{\mathcal{D}} = |S_{\mathcal{D}}| \le \sum_{i=0}^{\deg(S_{\mathcal{D}})} \binom{m}{i}.$$

From Theorem 1,

$$\deg(S_{\mathcal{D}}) = \deg\left(\bigcup_{s \in \operatorname{cr}(S_{\mathcal{C}})} \operatorname{cc}(P(s))\right).$$

Note that the binary expansion of 2s modulo n is a cyclic shift of \overline{s} . Thus, $\deg(C_s) = \operatorname{wt}(\overline{s})$ and $\deg\left(\operatorname{cc}(P(s))\right) = \deg(P(s))$. Then,

$$\deg(S_{\mathcal{D}}) = \deg\Big(\bigcup_{s \in \operatorname{cr}(S_{\mathcal{C}})} P(s)\Big).$$

Note that $\bigcup_{s\in {\rm cr}(S_{\mathcal C})} P(s) \subseteq \bigcup_{s\in S_{\mathcal C}} P(s),$ then

$$\deg(S_{\mathcal{D}}) \leq \deg\Big(\bigcup_{s \in S_{\mathcal{C}}} P(s)\Big).$$

For s = 0, we have $W_s = \emptyset$ and $P(s) = \emptyset$. Then $\deg(P(s)) = 0$. For any positive $s \in S_c$, from (3), we have $\deg(P(s)) = \operatorname{wt}(\overline{s}) - 1$. Then $\deg(\bigcup_{s \in S_c} P(s)) = \deg(S_c) - 1$. As a result,

$$k_{\mathcal{D}} \leq \sum_{i=0}^{\deg(S_{\mathcal{D}})} \binom{m}{i} \leq \sum_{i=0}^{\deg(S_{\mathcal{C}})-1} \binom{m}{i}.$$

Proposition 2. $d_{\mathcal{D}} \leq 2d$.

Proof. Consider the codeword $A(z) \in C$ with wt(A(z)) = d. The Hamming weight of its derivative $\Delta_{\beta}A(z)$ satisifies

$$\operatorname{wt}\left(\Delta_{\beta}A(z)\right) = \operatorname{wt}\left(A(z+\beta) - A(z)\right)$$
$$\leq \operatorname{wt}\left(A(z+\beta)\right) + \operatorname{wt}\left(A(z)\right)$$
$$= 2d$$

As a result, $d_{\mathcal{D}} \leq 2d$.

B. Cyclic derivative ascendant

Conversely to the cyclic DDs, we define cyclic DAs of an extended cyclic code as follows.

Definition 2. For an extended cyclic code C, we define its cyclic derivative ascendant denoted by A(C) as the extended cyclic code with the largest dimension such that $D(A(C)) \subseteq C$.

We give a proposition to characterize the exponent set of the cyclic DA of C.

Proposition 3. Let S_A denote the exponent set of $\mathcal{A}(\mathcal{C})$. A nonnegative integer s smaller than n is in S_A if and only if

$$cc(P(s)) \subseteq S_{\mathcal{C}}.$$

Proof. Because the conjugacy constraint is required, $s \in S_A$ if and only if $C_s \subseteq S_A$ which is equivalent to $\mathcal{M}_s \subseteq \mathcal{A}(\mathcal{C})$.

If $\mathcal{M}_s \subseteq \mathcal{A}(\mathcal{C})$, from (2) and Definition 2, we have $\mathcal{D}(\mathcal{M}_s) \subseteq \mathcal{D}(\mathcal{A}(\mathcal{C})) \subseteq \mathcal{C}$. From Lemma 1, the exponent set of $\mathcal{D}(\mathcal{M}_s)$ is $\operatorname{cc}(P(s))$. Then $\operatorname{cc}(P(s)) \subseteq S_{\mathcal{C}}$.

If $cc(P(s)) \subseteq S_{\mathcal{C}}$, then $\mathcal{D}(\mathcal{M}_s) \subseteq \mathcal{C}$. From Definition 2, $\mathcal{A}(\mathcal{C})$ is the extended cyclic code with the largest dimension such that $\mathcal{D}(\mathcal{A}(\mathcal{C})) \subseteq \mathcal{C}$. As a result, $\mathcal{M}_s \subseteq \mathcal{A}(\mathcal{C})$.

Now we investigate the dimension and distance of $\mathcal{A}(\mathcal{C})$. Let $k_{\mathcal{A}}$ and $d_{\mathcal{A}}$ denote the dimension and minimum Hamming distance of $\mathcal{A}(\mathcal{C})$, respectively. We give the following propositions. The proofs are given in the Appendix.



Fig. 2. Derivative decoding for extended cyclic codes.

Proposition 4. $k_{\mathcal{A}} \leq \sum_{i=0}^{\deg(S_{\mathcal{C}})+1} \binom{m}{i}.$

Proposition 5. $d_A \ge d/2$.

IV. DERIVATIVE DECODING FOR EXTENDED BINARY CYCLIC CODES

In this section, we propose a derivative decoding based on the decodings of cyclic DDs. It can efficiently decode the cyclic codes whose cyclic DDs have efficient soft-decision decoding algorithms. In particular, we propose to perform the derivative decoding on those codes whose cyclic DDs are extended Euclidean Geometry (EG) codes [11] [3, Chap. 8] which can be efficiently decoded by the *sum-product algorithm* (SPA) [18], [19]. In addition, we discuss the cyclic DDs and DAs of RM codes, and their decodings.

A. Algorithm description

Let $y = [y_i, i \in I]$ denote the received vector of transmitting a codeword $a = [A(\alpha^i), i \in I]$ of the extended binary cyclic code C over a binary-input memoryless symmetric (BMS) channel. Let W(y|x) denote the probability that y is output by the channel when x is input to the channel. The LLR vector of the channel output y is $L = [L_i : i \in I]$, where L_i is given by

$$L_{i} = \ln(\frac{W(y_{i}|0)}{W(y_{i}|1)}).$$
(4)

The algorithm takes L as an input and runs in an iterative manner. Let B denote a collection of directions, i.e. a subset of $\mathbb{F}_{2^m}^*$, where $\mathbb{F}_{2^m}^*$ consists of all the nonzero elements in \mathbb{F}_{2^m} . As shown in Fig. 2, each iteration has three steps: 1) calculate LLR vectors of cyclic DDs for all $\beta \in B$; 2) decode all the cyclic DDs; 3) vote for the estimated codeword from the decoded descendant codewords.

1) The LLR vector associated with the cyclic DD in the direction β is defined as

$$\boldsymbol{L}^{\beta} \triangleq [L_i^{\beta}, i \in I], \tag{5}$$

where L_i^{β} is the LLR value associated with $\Delta_{\beta}A(\alpha^i) = A(\alpha^i + \beta) - A(\alpha^i)$. We calculate L_i^{β} as

$$L_i^\beta = 2\tanh^{-1}\left(\tanh(\frac{L_i}{2})\tanh(\frac{L_j}{2})\right),\tag{6}$$

Algorithm 1 Derivative Decoding Based on Cyclic Derivative Descendants

Input: The LLR vector L; the maximum iteration number N_{max} ; a collection of directions B; the parity check matrix **H**

Output: The decoded codeword: \hat{a}

1: for $t = 1, 2, ..., N_{max}$ do for $\beta \in B$ do 2: $L^{\beta} \leftarrow \text{derivativeLLR}(L, \beta)$ 3: $\hat{a}^eta \leftarrow ext{decoderDD}(L^eta)$ 4: $\widetilde{L}^{\beta} \leftarrow \texttt{qetVote}(L, \hat{a}^{\beta}, \beta)$ 5: end for 6: $oldsymbol{L} \leftarrow rac{1}{|B|} \sum_{eta \in \mathbb{F}_{2^m}^*} \widetilde{oldsymbol{L}}^eta$ \triangleright Here, \sum denotes the component-wise summation 7: $\hat{a}_i \leftarrow \mathbb{1}[L_i < 0]$ for all $i \in I$ 8: if $\mathbf{H}a^{\mathrm{T}} = \mathbf{0}$ then 9: 10: Break end if 11: 12: end for 13: return \hat{a}

where j satisfies $\alpha^j = \alpha^i + \beta$. We denote the procedure of calculating L^{β} with the input L and β by $L^{\beta} =$ derivativeLLR(L, β).

2) According to Theorem 1, we can use the same decoder, denoted by decoderDD, to decode all the cyclic DDs. The decoding result of L^{β} is given by $\hat{a}^{\beta} = \text{decoderDD}(L^{\beta})$.

3) The final step is to use a soft-voting scheme to obtain a new LLR vector \hat{L} . From (6), the "soft vote" from the estimate \hat{a}_i^{β} to L_i is $\tilde{L}_i^{\beta} \triangleq (1 - 2\hat{a}_i^{\beta})L_j$. For the direction β , the "soft vote" from \hat{a}^{β} to L is given by

$$\widetilde{m{L}}^eta = \texttt{getVote}(m{L}, \hat{m{a}}^eta, eta) riangleq [(1-2\hat{a}_i^eta)L_j, i \in I].$$

Here we have used the natural embedding of \mathbb{F}_2 in \mathbb{R} for the interpretation of \hat{a}_i^{β} in the above equation. Update L as the average of all the "soft votes" from different directions

$$\boldsymbol{L} = \frac{1}{|B|} \sum_{\beta \in \mathbb{F}_{2m}^*} \widetilde{\boldsymbol{L}}^{\beta}.$$

Here, \sum denotes the component-wise summation.

Once we update L, we take $\hat{a} = [\hat{a}_i, i \in I]$ where $\hat{a}_i = \mathbb{1}[L_i < 0]$. If \hat{a} is a codeword in C, i.e. $\mathbf{H}\hat{a}^{\mathrm{T}} = \mathbf{0}$ where \mathbf{H} is the particy-check matrix of C and \hat{a}^{T} denotes the transpose of \hat{a} , we end the iteration and output \hat{a} . Otherwise, proceed into the next iteration unless obtaining a codeword or reaching a maximal iteration number N_{max} . The pseudo code of the above procedure is shown in Algorithm 1.

Remark 1. The proposed decoding works for cyclic codes of length of $2^m - 1$ as well. Set $L_{\infty}=0$. Then we can decode them as their extended cyclic codes.

Remark 2. In Algorithm 1, the functions derivativeLLR, decoderDD and getVote can be implemented separately for each direction. As a result, the proposed algorithm can be implemented in parallel.

B. Computational complexity

We analyze the computational complexity of the proposed algorithm per iteration according to Algorithm 1. Denote the code length by n. It takes 4n floating point operations to perform derivativeLLR. In each iteration, the decoder performs derivativeLLR and decoderDD |B| times. At the end of each iteration, it needs |B|n floating point operations for calculating the average of all the "soft votes". Denote the number of floating point operation of decoderDD by Ω . We get the following proposition.

Proposition 6. The derivative decoding consumes $5|B|n + |B|\Omega$ floating point operations per iteration.

C. Derivative decoding for eBCH codes based on SPA

Consider the (64, 24) eBCH code and the (64, 45) eBCH code. The corresponding generator polynomials in hexadecimal form are 0xF69AC20921 and 0x782CF, respectively. From (1), the corresponding representative sets are $S_1 = \{0, 1, 3, 5, 9, 21\}$ with deg $(S_1) = 3$ and $S_2 = \{0, 1, 3, 5, 7, 9, 11, 13, 21, 27\}$ with deg $(S_2) = 4$. According to Theorem 1, the representative sets associated with their cyclic DDs are $\{0, 1, 5\}$ and $\{0, 1, 3, 5, 9, 11, 13\}$. The corresponding codes are the (64, 13) extended EG code and a (64, 34) extended cyclic code. Please note that the (64, 34) extended cyclic code is a subcode of the (64, 37) extended EG code.

Consider *additive white Gaussian noise* (AWGN) channels. We decode the two eBCH codes by our proposed derivative decoding algorithm. The decoder for their cyclic DDs is the SPA decoder. The parity-check matrix used for decoding the (64, 13) extended EG code is a 336×64 matrix with row weight 4 and column weight 21, and the one used for decoding the (64, 34) extended cyclic code is a 72×64 matrix with row weight 8 and column weight 17. We set the maximum iteration numbers for SPA and DD as $N_{SPA,max} = 20$ and $N_{DD,max} = 3$, respectively. We perform derivative decoding in all the 63 directions and denote the procedure by DD(63)-SPA. In addition, we perform derivative decoding in 16 directions at random and denote the procedure by DD(16)-SPA. We compare with the performance of decoding BCH codes using the Berlekamp-Massey (BM) algorithm [5] [20].

The simulation results are shown in Fig. 3 and the performance of the MLD is also provided. We see that at the block error ratio (BLER) of 10^{-4} , DD(63)-SPA for the (64, 24) eBCH code and the (64, 45) eBCH code outperforms BM for the corresponding BCH codes about 2.9 dB and 1.9 dB, respectively. Moreover, the gaps between the MLD and DD(63)-SPA are 0.5 dB and 0.3 dB in, respectively. In addition, for decoding the (64, 45) eBCH code, DD(16)-SPA only performs about 0.2 dB away from DD(63)-SPA at the BLER of 10^{-4} .

Besides, we compare with the performance of decoding 5G CA-polar codes [21] [22] with the same length and dimension. The decoder used for the CA-polar codes is the Successive Cancellation List (SCL) decoder [23] with list size 32. And the CRC length is set to 6. At the BLER of 10^{-5} , DD(63)-SPA for the (64, 24) eBCH code and the (64, 45) eBCH code outperforms SCL for the CA-polar codes with the same length and dimension about 0.4 dB and 0.2 dB, respectively.



Fig. 3. Performance of decoding eBCH codes using DD-SPA, decoding BCH codes using BM algorithm and decoding CA-polar codes using SCL decoder. The list size of the SCL decoder is 32 and the CRC length is 6.

We discuss the computational complexity of decoding the (64, 45) eBCH code using DD(63)-SPA. Denote the floating point operation cost of the SPA in one iteration by Ω_{SPA} . From Proposition 6, the number of the floating point operations is $320 * 63 + 63N_{SPA}\Omega_{SPA}$ per iteration where Ω_{SPA} is the average iteration number of the SPA. At the E_b/N_0 of 5.0 dB, the average iteration number of the SPA is 1.16 and the average iteration number of DD(63)-SPA is 1.00. As a result, the cost of DD(63)-SPA is around $20160 + 73\Omega_{SPA}$ floating point operations. Besides, the average iteration number of the SPA in DD(16)-SPA and the average iteration number of DD(16)-SPA is also 1.16 and 1.00, respectively. Then the cost of DD(16)-SPA is around $5120+19\Omega_{SPA}$ floating point operations.

D. Cyclic DDs of RM codes and their decoding

Consider the RM code of length 2^m and order r denoted by RM(r, m). The zero set of the generator polynomial associated with RM(r, m) is $\{\alpha^j : 0 < wt(\overline{j}) < m-r\}$ [4, Chap. 6]. According to (1), the exponent set of RM(r, m)is $\{j : 0 \le wt(\overline{j}) \le r\}$. According to Theorem 1, we conclude that the cyclic DD of RM(r, m) is RM(r - 1, m). Note that the dimension and minimum Hamming distance of RM(r, m) are $\sum_{i=0}^r {m \choose i}$ and 2^{m-r} , respectively. The equalities in Proposition 1 and Proposition 2 hold for RM codes and their cyclic DDs. Similarly, according to Proposition 3, we conclude that the cyclic DA of RM(r, m) is RM(r + 1, m). The equalities in Proposition 4 and Proposition 5 hold for the RM codes and their cyclic DAs.

As a result, we can decode $\operatorname{RM}(r, m)$ codes using derivative decoding based on the decodings of $\operatorname{RM}(r-1, m)$ codes. Let T denote a subset of \mathbb{F}_{2^m} such that $(\beta + T) \cup T = \mathbb{F}_{2^m}$. From (2), we have $[\Delta_\beta A(\alpha^i), \alpha^i \in T] = [\Delta_\beta A(\alpha^i), \alpha^i \in \beta + T]$. Considering the $|\boldsymbol{u}|\boldsymbol{u} + \boldsymbol{v}|$ -construction and the automorphism groups of RM codes [2, Chap. 13], we conclude that $[\Delta_\beta A(\alpha^i), \alpha^i \in T]$ is a codeword of $\operatorname{RM}(r-1, m-1)$. The proof is given in the Appendix. From (6), we have $[L_i^\beta, \alpha^i \in T] = [L_i^\beta, \alpha^i \in \beta + T]$, where $[L_i^\beta, \alpha^i \in T]$ is the LLR vector associated

For a subset T of \mathbb{F}_{2^m} , $\beta + T$ denote the set $\{\beta + \alpha^i : \alpha^i \in T\}$.

with $[\Delta_{\beta}A(\alpha^i), \alpha^i \in T]$. In other words, our derivative decoding derived from the MS polynomials can carry on based on the decodings of RM(r-1, m-1) codes as the state-of-the-art projection decodings [14], [15] derived from the *m*-variate polynomials, and obtain the same performance.

E. Decoding cyclic derivative ascendants of EG codes

If an extended cyclic code can be efficiently soft-decision decoded, then its cyclic DA can be soft-decision decoded by the derivative decoding algorithm. Consider the (256, 175) extended EG code with minimum Hamming distance 18 which can be efficiently decoded by the SPA decoder. The corresponding generator polynomial is 0x11377F7700FA55335BA55. The representative set of its exponent set is $S = \{0, 1, 3, 5, 7, 9, 11, 13, 17, 19, 21, 23, 25, 27, 29, 37, 39, 43, 51, 53, 55, 59, 85, 87, 119\}$ with deg(S) = 6. According to Proposition 3, we can construct its cyclic DA. It is an extended cyclic code of length 256 and dimension $191 \le \sum_{i=0}^{7} {8 \choose i}$ with distance $d \ge 18/2 = 9$, according to Proposition 4 and Proposition 5. In fact, this code, denoted by DA(256, 191) has minimum Hamming distance at least 16 according to the BCH bound [4]. The corresponding generator polynomial of DA(256, 191) is 0x19ACCC1AE68A0CEFF.

In Fig. 4, we provide the simulation result of decoding DA(256, 191) using derivative decoding based on SPA with all the directions in $\mathbb{F}_{2^m}^*$, denoted by DD(255)-SPA. The parity-check matrix used for decoding the (256, 175) extended EG code is a 272 × 256 matrix with row weight 16 and column weight 17. The maximum iteration numbers for derivative decoding and SPA are set to $N_{DD,max} = 4$ and $N_{SPA,max} = 20$, respectively. The performance of the MLD is also provided. We see that at the BLER of 10^{-4} , the gap between the MLD and DD-SPA is about 0.9 dB.



Fig. 4. Performance of decoding DA(256, 191) using DD(255)-SPA.

V. MINIMAL DERIVATIVE DESCENDANTS AND DERIVATIVE DECODING

This section investigates the minimal subspace which contains all the derivatives of an extended cyclic code in one direction. We prove that all these subspaces, denoted as the minimal DDs, are equivalent. Similarly, we can decode an extended cyclic code based on the decodings of its minimal DDs. Simulation results show that the derivative decoding based on the OSD with order-1 can outperform the OSD with higher order. In the following, we denote the OSD with order-*l* by OSD(l).

A. Minimal derivative descendants of extended cyclic codes

Definition 3. Consider an extended cyclic code C and the direction β . We denote the minimal subspace which contains the derivatives of all the codewords of C in β as the minimal derivative descendant in β

$$\mathcal{D}_{\beta}(\mathcal{C}) = \{ [\Delta_{\beta} A(\alpha^{i}), i \in I] : A(z) \in \mathcal{C} \}.$$

Two codes are said to be *equivalent* [2, Chap. 1] if there is a permutation of the coordinates together with permutations of the coordinate values for each of the coordinates, that map the codewords of one code into those of the other. The following theorem proves that the minimal DDs in different directions are equivalent.

For an integer b, we make the agreement $\infty + b = \infty \mod n$. For a codeword $a = [a_i, i \in I] \in C$, we define the b-cyclic shift of a as $a^{(b)} \triangleq [a_{i+b}, i \in I]$.

Theorem 2. The minimial DDs of an extended cyclic code C in different directions are equivalent.

Proof. Let β_1 and β_2 be powers of α , i.e. $\beta_1 = \alpha^{b_1}$ and $\beta_2 = \alpha^{b_2}$. For any codeword $\Delta_{\beta_1} A(z) \in \mathcal{D}_{\beta_1}(\mathcal{C})$, there is $\Delta_{\beta_2} A(\beta_2^{-1}\beta_1 z) \in \mathcal{D}_{\beta_2}(\mathcal{C})$ such that

$$\Delta_{\beta_2} A(\beta_2^{-1} \beta_1 z) = A\left(\beta_2^{-1} \beta_1(z+\beta_2)\right) - A(\beta_2^{-1} \beta_1 z)$$

= $A(\alpha^{b_1 - b_2} z + \beta_1) - A(\alpha^{b_1 - b_2} z),$ (7)

is the $(b_1 - b_2)$ -cyclic shift of $\Delta_{\beta_1} A(z)$. Please note that this is true for any pair of β_1 and β_2 . As a result, the minimal DDs of C in all the directions are equivalent.

The above theorem shows that the cyclic shift of a codeword in one minimal DD is a codeword in another minimal DD. We obtain the following corollary immediatly from (7) by setting $b_1 = b$ and $b_2 = 0$.

Corollary 1. For any codeword $A(z) \in C$, the b-cyclic shift of $\Delta_{\alpha^b} A(z)$ is equal to $\Delta_1 A(\alpha^b z)$.

According to Definitions 1 and 3, the minimal DDs are the subcodes of the corresponding cyclic DDs. In the next proposition, we show that the cyclic DD of an extended cyclic code C is the summation of all the minimal DDs of C.

Proposition 7. For an extended cyclic code C, the summation of all its minimal DDs is its cyclic DD, i.e.,

$$\mathcal{D}(\mathcal{C}) = \sum_{\beta \in \mathbb{F}_{2m}^*} D_{\beta}(\mathcal{C}).$$
(8)

Proof. We are going to prove that $\sum_{\beta \in \mathbb{F}_{2m}^*} D_{\beta}(\mathcal{C})$ is the smallest extended cyclic code containing all the minimal DDs of \mathcal{C} . Obviously, it is the smallest subspace of \mathbb{F}_2^n containing all the minimal DDs of \mathcal{C} . We only need to prove that $\sum_{\beta \in \mathbb{F}_{2m}^*} D_{\beta}(\mathcal{C})$ is an extended cyclic code.

For any codeword $\Delta_{\beta}A(z) = A(z+\beta) - A(z)$ in $\sum_{\beta \in \mathbb{F}_{2m}^*} D_{\beta}(\mathcal{C})$, its cyclic shift

$$A(\alpha z + \beta) - A(\alpha z) = A\left(\alpha(z + \alpha^{-1}\beta)\right) - A(\alpha z)$$

= $\Delta_{\alpha^{-1}\beta}A(\alpha z),$ (9)

is also a codeword in $\sum_{\beta \in \mathbb{F}_{2m}^*} D_{\beta}(\mathcal{C})$. As a result, $\sum_{\beta \in \mathbb{F}_{2m}^*} D_{\beta}(\mathcal{C})$ is an extended cyclic code.

In general, it is hard to determine the minimal Hamming distance of an arbitrary linear code. With Proposition 7, we can tell that the minimal Hamming distance of a minimal DD of C is lower bounded by the minimal Hamming distance of the cyclic DD of C, which can be lower bounded by the BCH bound [24], [25].

Example 3. Continuation of Example 2. The generator matrix G of the (16,7) extended cyclic code is

1	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0
1	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0
1	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0
1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0
1	0	0	0	0	0	0	1	0	0	0	1	0	1	1	1

The columns are indexed by 0, α^0 , α^1 , ..., α^{14} . Calculate the derivatives of the rows of **G** in α^0 and obtain the following matrix

-															-	
0	0	1	1	0	1	0	1	1	1	1	0	0	0	1	0	
1	1	1	1	0	1	1	0	0	1	0	1	0	0	0	0	
1	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	
1	1	0	0	1	0	1	0	0	0	0	1	1	1	0	1	.
1	1	1	1	0	1	1	0	0	1	0	1	0	0	0	0	
1	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	
1	1	0	0	1	0	1	0	0	0	0	1	1	1	0	1	
															_	

The minimal DD $\mathcal{D}_1(\mathcal{C})$ of \mathcal{C} is spanned by the rows of the above matrix. We can perform Gauss elimination on the above matrix and obtain the generator matrix of $\mathcal{D}_1(\mathcal{C})$,

1	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	
0	0	1	1	0	1	0	1	1	1	1	0	0	0	1	0	.
0	0	0	0	1	0	0	1	1	0	1	0	1	1	1	1	

It shows that $\mathcal{D}_1(\mathcal{C})$ is a (16,3) code. From Proposition 7 and Example 2, $\mathcal{D}_1(\mathcal{C})$ is a subcode of the (16,5) Hadamard code whose minimal Hamming distance is 8. Thus, its minimal Hamming distance is at least 8. In fact, its minimal Hamming distance is exactly 8.

B. Decoding based on decodings of minimal DDs

According to Theorem 2 and Corollary 1, we can perform derivative decoding on C based on the decodings of $\mathcal{D}_1(C)$ with cyclic shiftings.

Consider transmitting a codeword $\boldsymbol{a} = [A(\alpha^i), i \in I]$ over a BMS channel, and the recieved vector is \boldsymbol{y} . Let \boldsymbol{L} denote the corresponding LLR vector. For the direction β , we denote the vector $[\Delta_{\beta}A(\alpha^i), i \in I]$ by \boldsymbol{a}^{β} . From Corollary 1, the *b*-cyclic shift of \boldsymbol{a}^{β} , denoted by $\boldsymbol{a}^{\beta,(b)}$, is equal to the derivative of $\boldsymbol{a}^{(b)}$ in α^0 . We denote the *b*-cyclic shift of \boldsymbol{L} as $\boldsymbol{L}^{(b)}$. Then we calculate the LLR vector associated with $\boldsymbol{a}^{\beta,(b)}$ as

$$\boldsymbol{L}^{\beta,(b)} \triangleq [L_i^{\beta,(b)}, i \in \boldsymbol{I}], \tag{10}$$

where

$$L_i^{\beta,(b)} = 2 \tanh^{-1} \Big(\tanh(\frac{L_i^{(b)}}{2}) \tanh(\frac{L_j^{(b)}}{2}) \Big), \tag{11}$$

where j satisifies $\alpha^j = \alpha^i + 1$. Now we can treat $L^{\beta,(b)}$ as a LLR vector associated with an codeword in $\mathcal{D}_1(\mathcal{C})$. Denote a soft-decision decoder for $\mathcal{D}_1(\mathcal{C})$ by decoderDD. The estimate of $a^{\beta,(b)}$ is given by $\hat{a}^{\beta,(b)} = decoderDD(L^{\beta,(b)})$. According to (11), the "soft vote" for $L_i^{(b)}$ from the direction β is $\tilde{L}_i^{\beta,(b)} = (1-2\hat{a}_i^{\beta,(b)})L_j^{(b)}$. And the "soft vote" for $L^{(b)}$ from the direction β is given by

$$\widetilde{m{L}}^{eta,(b)} = ext{getVote}(m{L}^{(b)},m{a}^{eta,(b)},lpha^0)$$

Cyclicly shift it *b* places to the right and obtain the "soft vote" for *L* from the direction β , i.e. \tilde{L}^{β} . The remaining steps mimic to Algorithm 1 in Section IV. We provide the pseudo code in Algorithm 2.

Remark 3. The advantage of calculating $L^{\beta,(b)}$ rather than L^{β} is that we can treat $L^{\beta,(b)}$ as the LLR vector associated with the direction α^0 for all $\beta \in \mathbb{F}_{2^m}$. It allows us to keep using derivative, decoderDD and getVote for the direction α^0 .

Remark 4. For any $A(z) \in C$, there is $\Delta_1 A(\alpha^i) = \Delta_1 A(\alpha^i + 1)$. Denote a subset of \mathbb{F}_{2^m} by T such that $T \cup (1+T) = \mathbb{F}_{2^m}$. For any codeword $\Delta_1 A(z) \in \mathcal{D}_1(C)$, there is $[\Delta_1 A(\alpha^i), \alpha^i \in T] = [\Delta_1 A(\alpha^i), \alpha^i \in 1+T]$. Besides, from (11), there is $[L_i^{\beta,(b)}, \alpha^i \in 1+T] = [L_i^{\beta,(b)}, \alpha^i \in 1+T]$. It can be used to simplify the decoding for minimal DDs in Algorithm 2.

Algorithm 2 Derivative Decoding Based on Minimal Derivative Descendants

Input: The LLR vector L; the maximum iteration number N_{max} ; a collection of directions B; the parity check matrix **H**

Output: The decoded codeword: \hat{a}

1: for $t = 1, 2, ..., N_{max}$ do for b = 1, 2, ..., n do 2: $\pmb{L}^{(b)} \leftarrow \pmb{L}^{(b-1)}$ \triangleright Take L as $L^{(0)}$ 3: if $\alpha^b \in B$ then 4: $\boldsymbol{L}^{\beta,(b)} \leftarrow \text{derivativeLLR}(\boldsymbol{L}^{(b)},1)$ 5: $\hat{a}^{eta,(b)} \leftarrow \texttt{decoderDD}(L^{eta,(b)})$ 6: $\widetilde{L}^{\beta,(b)} \leftarrow \texttt{qetVote}(\hat{a}^{\beta,(b)}, L^b, 1)$ 7: $\widetilde{L}^{\beta} \leftarrow \widetilde{L}^{\beta,(b)}$ \triangleright Cyclicly shift $\widetilde{L}^{\beta,(b)}$ b places to the right 8: end if 9: end for 10: $L \leftarrow rac{1}{|B|} \sum_{eta \in B} \widetilde{L}^eta$ \triangleright Here, \sum denotes the component-wise summation 11: $\hat{a}_i \leftarrow \mathbb{1}[L_i < 0]$ for all $i \in I$ 12: if $\mathbf{H}a^{\mathrm{T}} = \mathbf{0}$ then 13: 14: Break end if 15: 16: end for 17: return \hat{a}

C. Derivative decoding based on OSD

We propose to perform derivative decoding based on OSD. Suppose $A_1(z)$, $A_2(z)$, ..., $A_{k_D}(z)$ forms a basis of $\mathcal{D}_1(\mathcal{C})$. We can perform OSD based on the generator matrix given by this basis,

$$\mathbf{G}_{\mathcal{D}} = \begin{bmatrix} A_1(\alpha^{\infty}) & A_1(\alpha^0) & \dots & A_1(\alpha^{n-1}) \\ A_2(\alpha^{\infty}) & A_2(\alpha^0) & \dots & A_2(\alpha^{n-1}) \\ \dots & \dots & \dots & \dots \\ A_{k_{\mathcal{D}}}(\alpha^{\infty}) & A_{k_{\mathcal{D}}}(\alpha^0) & \dots & A_{k_{\mathcal{D}}}(\alpha^{n-1}) \end{bmatrix}$$
$$\triangleq [\mathbf{g}_{\alpha^{\infty}} \quad \mathbf{g}_{\alpha^0} \quad \dots \quad \mathbf{g}_{\alpha^{n-1}}].$$

In fact, from Remark 4, we can implement the OSD decoder with the matrix $[g_{\alpha^i}, \alpha^i \in T]$ and only take $[L_i^{\beta,(b)}, \alpha^i \in T]$ as the input when decoding $L^{\beta,(b)}$ in Algorithm 2. We denote the derivative decoding based on OSD with by DD-OSD. In particular, we focus on the derivative decoding based on OSD(1) and denote it by DD-OSD(1).

Consider the (128, 36), (256, 37), and (256, 79) eBCH codes. We investigate their cyclic DDs and minimal DDs,

TABLE I CODE PARAMETERS OF EXTENDED BCH CODES AND THEIR DESCENDANTS

$n_{\mathcal{C}}$	$k_{\mathcal{C}}$	$d_{\mathcal{C},\mathbf{BCH}}$	$k_{\mathcal{D}}$	$d_{\mathcal{D},\mathbf{BCH}}$	$k_{\mathcal{D}_1}$
128	36	32	22	48	14
256	37	92	25	96	16
256	79	56	45	64	31



Fig. 5. Performance of decoding eBCH codes using DD-OSD(1) and OSD(3).

and list the code parameters in Table I. In the top line of Table I, $n_{\mathcal{C}}$ and $k_{\mathcal{C}}$ denote the code length and the code dimension of the eBCH codes, respectively; $k_{\mathcal{D}}$ and $k_{\mathcal{D}_1}$ denote the code dimension of the cyclic DDs and minimal DDs, respectively; $d_{\mathcal{C},BCH}$ and $d_{\mathcal{D},BCH}$ denote the BCH bounded distance of these eBCH codes and their cyclic DDs, respectively.

In general, DD-OSD(1) outperforms OSD(3) for decoding extended cyclic codes with moderate codelength at high SNR regions. For derivative decoding the (n_c, k_c) eBCH code in Table I, we take *B* as a collection of all the nonzero elements in the corresponding splitting field and denote the procedure by DD(|*B*|)-OSD(1). From Remark 4, we can decode its minimal DD as a $(n_c/2, k_{D_1}/2)$ code with minimal Hamming distance $d_{D,BCH}/2$. In addition, we perform derivative decoding in |B| = 32 directions at random, and denote the procedure by DD(32)-OSD(1). The maximum iteration number $N_{DD,max}$ is 4 in all the cases. We compare with the OSD(3) and provide the simulation results over AWGN channels in Fig. 5.

Following the complexity analysis in [17], we investigate the number of floating point operations of OSD of DD-OSD(1). It consumes $n_{\mathcal{D}}\log_2(n_{\mathcal{D}}) + k_{\mathcal{D}_1}(n_{\mathcal{D}} - k_{\mathcal{D}_1})$ floating point operations to decode the $(n_{\mathcal{D}}, k_{\mathcal{D}_1})$ minimal DD using the OSD(1). From Proposition 6, it consumes $|B|5n + |B|(n_{\mathcal{D}}\log_2(n_{\mathcal{D}}) + k_{\mathcal{D}_1}(n_{\mathcal{D}} - k_{\mathcal{D}_1}))$ floating point operations to perform DD(|B|)-OSD(1) on the (n, k) eBCH code per iteration.

Consider decoding the (256, 79) eBCH code at the E_b/N_0 of 4.0 dB. The average iteration number of DD(255)-OSD(1) is 1.02 and that of DD(32)-OSD(1) is 1.03. As a result, the average cost of DD(255)-OSD(1) is 1.02 * 255 * 1280 + 1.02 * (255 * 8 + 255 * 176 * 79) = 3,951,439 floating point operations and that of DD(32)-OSD(1) is 1.03 * 32 * 1280 + 1.03 * (32 * 8 + 32 * 176 * 79) = 500,728 floating point operations. For comparison, the cost

of OSD(3) is $\binom{79}{3} + \binom{79}{2} + \binom{79}{1} + 176 + 256 + 8 = 14,476,112$ floating point operations.

VI. CONCLUSION

This paper introduces cyclic DDs and minimal DDs for extended cyclic codes and investigates their properties. These properties allow us to decode extended cyclic codes with soft-decision. Besides, it works for cyclic codes of length of $2^m - 1$ as well according to Remark 1. Simulation results verify that they perform very well for some eBCH codes over AWGN channels.

APPENDIX A

PROOF OF PROPOSITION 4

The dimension of $\mathcal{A}(\mathcal{C})$ satisfies

$$k_{\mathcal{D}} = |S_{\mathcal{A}}| \le \sum_{i=0}^{\deg(S_{\mathcal{A}})} \binom{m}{i}.$$

From Proposition 3, for any $s \in S_A$, $P(s) \subseteq S_C$. From (3), $\deg(P(s)) = \operatorname{wt}(\overline{s}) - 1$. Then

$$\mathrm{wt}(\overline{s}) = \mathrm{deg}\Big(P(s)\Big) + 1 \leq \mathrm{deg}(S_{\mathcal{C}}) + 1.$$

This leads $\deg(S_{\mathcal{A}}) \leq \deg(S_{\mathcal{C}}) + 1$. As a result,

$$k_{\mathcal{A}} \leq \sum_{i=0}^{\deg(S_{\mathcal{A}})} \binom{m}{i} \leq \sum_{i=0}^{\deg(S_{\mathcal{C}})+1} \binom{m}{i}.$$

APPENDIX B

PROOF OF PROPOSITION 5

Let $\mathcal{D}(\mathcal{A}(\mathcal{C}))$ denote the cyclic DD of $\mathcal{A}(\mathcal{C})$ with minimum Hamming distance $d_{\mathcal{D}(\mathcal{A})}$. From Proposition 2, we have $d_{\mathcal{D}(\mathcal{A})} \leq 2d_{\mathcal{A}}$. From Definition 2, we have $\mathcal{D}(\mathcal{A}(\mathcal{C})) \subseteq \mathcal{C}$ which indicates $d_{\mathcal{D}(\mathcal{A})} \geq d$. As a result, $d_{\mathcal{A}} \geq d/2$.

APPENDIX C

PROOF OF EQUIVALENCE FOR RM CODES

To begin, we recap the equivalence between representing RM codes by *m*-variate polynomials and representing RM codes by MS polynomials. The exponent set of RM(r, m) is $S = \{s : 0 \le wt(\overline{s}) \le r\}$. For any $A(z) \in RM(r, m)$, we can write it as

$$A(z) = \sum_{s \in S} A_s z^s.$$

Note that we can write z as $z = \sum_{i=0}^{m-1} z_i \alpha^i$ where $z_i \in \mathbb{F}_2$ and we can write s as $s = \sum_{j=0}^{m-1} s_j 2^j$ where $s_j \in \{0, 1\}$. Then

$$A(z) = \sum_{s \in S} A_s \left(\sum_{i=0}^{m-1} z_i \alpha^i\right)^{\sum_{j=0}^{m-1} s_j 2^j}$$
$$= \sum_{s \in S} A_s \prod_{j=0}^{m-1} \left(\sum_{i=0}^{m-1} z_i \alpha^{i 2^j}\right)^{s_j}.$$

Please note that for any $s \in S$, $0 \le wt(\overline{s}) \le r$. Thus

$$A(z) = \sum_{V \subseteq [m], |V| \le r} u_V \prod_{i \in V} z^i,$$

where u_V is a summation of $A_s \alpha^{i2^j}$ over a collection of s, i, j. For |V| = 0, $A(0) = u_\emptyset$, so $u_\emptyset \in \mathbb{F}_2$. For |V| = 1, $A(\alpha^i) = u_\emptyset + u_{\{i\}}$. Therefore, $u_{\{i\}} \in \mathbb{F}_2$ for all $i \in [m]$. Note that for any $V \subseteq [m]$, $A(\sum_{i \in V} \alpha^i) = \sum_{V' \subseteq V} u_{V'} = \sum_{V' \subsetneq V} u_{V'} + u_V$. One can easily prove that $u_V \in \mathbb{F}_2$ for all $V \in [m]$ and $|V| \leq r$ by induction. As a result, we can treat A(z) as a *m*-variate Boolean polynomial $A(z_0, z_1, ..., z_{m-1})$ with degree no larger than *r*. Moreover, $[A(\alpha^i), i \in I]$ is equal to $[A(z_0, z_1, ..., z_{m-1}), [z_0, z_1, ..., z_{m-1}] \in \mathbb{F}_2^m]$. Note that the dimension of RM(r, m) is $\sum_{i=0}^r {m \choose i}$. We conclude that RM(r, m) consists of the evaluation vectors of all the *m*-variate Boolean polynomials with degree no larger than *r* over \mathbb{F}_2^m .

First consider the derivative of A(z) in the direction α^0 ,

$$\Delta_{1}A(z) = A(z+1) - A(z)$$

$$= A\Big(\sum_{i=1}^{m-1} z_{i}\alpha^{i} + (z_{0}+1)\Big) - A(\sum_{i=0}^{m-1} z_{i}\alpha^{i})$$

$$= \sum_{\substack{V \subseteq [m], |V| \le r, \\ 0 \notin V}} u_{V}(z_{0}+1) \prod_{i \in V/\{0\}} z_{i} +$$

$$\sum_{\substack{V \subseteq [m], |V| \le r, \\ 0 \notin V}} u_{V} \prod_{i \in V} z_{i} - \sum_{\substack{V \subseteq [m], |V| \le r}} u_{V} \prod_{i \in V} z_{i}$$

$$= \sum_{\substack{V \subseteq [m], |V| \le r, \\ 0 \in V}} u_{V} \prod_{i \in V/\{0\}} z_{i}.$$
(12)

It is a (m-1)-variate Boolean polynomial with degree no larger than r-1 and we denote it by $A'(z_1, z_2, ..., z_{m-1})$. Denote $T_1 = \{\sum_{j=1}^{m-1} z_j \alpha^j : z_j \in \mathbb{F}_2 \text{ for } j = 1, 2, ..., m-1\}$ such that $T_1 \cup (1+T_1) = \mathbb{F}_{2^m}$. The vector

$$[\Delta_1 A(\alpha^i), \alpha^i \in T] = [A'(z_1, z_2, ..., z_{m-1}), [z_1, z_2, ..., z_{m-1}] \in (\mathbb{F}_2)^{m-1}]$$
(13)

is a codeword in RM(r-1, m-1).

Now consider the derivative of A(z) in the direction $\beta = \alpha^b$. Take $T = \{\beta \alpha^i : \alpha^i \in T_1\}$. We have $\beta + T = \{\beta \alpha^i : \alpha^i \in 1 + T_1\}$ and $T \cup (\beta + T) = \mathbb{F}_{2^m}$. According to Corrollary 1, the *b*-cyclic shift of $\Delta_\beta A(z)$ is equal to $\Delta_1 A(\beta z)$. It indicates the evaluation of $\Delta_\beta A(z)$ at $\beta \alpha^i$ is equal to the evaluation of $\Delta_1 A(\beta z)$ at α^i . As a result,

$$[\Delta_{\beta}A(\alpha^{i}), \alpha^{i} \in T] = [\Delta_{1}A(\beta\alpha^{i}), \alpha^{i} \in T_{1}\}]$$

is also a codeword in RM(r-1, m-1).

REFERENCES

- [1] E. Prange, Cyclic error-correcting codes in two symbols. Air force Cambridge research center, 1957.
- [2] F. J. MacWilliams and N. J. A. Sloane, The theory of error correcting codes. Elsevier, 1977, vol. 16.
- [3] S. Lin and D. J. Costello, Error control coding. Prentice hall New York, 2001, vol. 2, no. 4.
- [4] R. E. Blahut, Algebraic codes for data transmission. Cambridge university press, 2003.

- [5] E. R. Berlekamp, Algebraic coding theory (revised edition). World Scientific, 2015.
- [6] A. Vardy and Y. Be'ery, "Maximum-likelihood soft decision decoding of BCH codes," *IEEE Trans. on Inf. Theory*, vol. 40, no. 2, pp. 546–554, 1994.
- [7] N. Kamiya, "On algebraic soft-decision decoding algorithms for BCH codes," IEEE Trans. on Inf. Theory, vol. 47, no. 1, pp. 45–58, 2001.
- [8] M. Bossert, R. Schulz, and S. Bitzer, "On hard and soft decision decoding of BCH codes," *IEEE Transactions on Information Theory*, 2022.
- [9] X. W. T. L. Tapp, A. A. Luna and S. B. Wicker, "Extended Hamming and BCH soft decision decoders for mobile data applications," *IEEE trans. on commun.*, vol. 47, no. 3, pp. 333–337, 1999.
- [10] S. Lin, K. Abdel-Ghaffar, J. Li, and K. Liu, "A Scheme for Collective Encoding and Iterative Soft-Decision Decoding of Cyclic Codes of Prime Lengths: Applications to Reed-Solomon, BCH, and Quadratic Residue Codes," *IEEE Trans. on Inf. Theory*, vol. 66, no. 9, pp. 5358–5378, 2020.
- [11] Y. Kou, S. Lin, and M. P. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.
- [12] V. M. Sidel'nikov and A. Pershakov, "Decoding of Reed-Muller codes with a large number of errors," *Probl. Peredachi Inf.*, vol. 28, no. 3, pp. 80–94, 1992.
- [13] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed-Muller codes: recursive lists," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1260–1266, Mar. 2006.
- [14] M. Ye and E. Abbe, "Recursive projection-aggregation decoding of Reed-Muller codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4948–4965, Aug. 2020.
- [15] M. Lian, C. Häger, and H. D. Pfister, "Decoding Reed-Muller codes using redundant code constraints," in Proc. IEEE Int. Symp. Inf. Theory. IEEE, June 2020, pp. 42–47.
- [16] H. Mattson and G. Solomon, "A new treatment of Bose-Chaudhuri codes," *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 654–669, 1961.
- [17] M. P. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1379–1396, 1995.
- [18] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
- [19] R. Lucas, M. P. Fossorier, Y. Kou, and S. Lin, "Iterative decoding of one-step majority logic deductible codes based on belief propagation," *IEEE Trans. Commun.*, vol. 48, no. 6, pp. 931–937, June 2000.
- [20] J. Massey, "Shift-register synthesis and BCH decoding," IEEE Trans. Inf. Theory, vol. 15, no. 1, pp. 122-127, Jan. 1969.
- [21] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," vol. 55, no. 7, pp. 3051–3073, 6 2009.
- [22] "3GPP TS 38.212, 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Multiplexing and Channel Coding (Release 16) V16.4.0 Technical Specification (TS)," Dec. 2020.
- [23] I. Tal and A. Vardy, "List decoding of polar codes," IEEE Trans. Inf. Theory, vol. 61, no. 5, pp. 2213–2226, Mar. 2015.
- [24] R. C. Bose and D. K. Ray-Chaudhuri, "Further results on error correcting binary group codes," *Information and Control*, vol. 3, no. 3, pp. 279–290, 1960.
- [25] —, "On a class of error correcting binary group codes," Information and control, vol. 3, no. 1, pp. 68–79, 1960.