# A Fast Method to Estimate Partial Weights Enumerators by Hash Techniques and Automorphism Group

Moulay Seddiq EL KASMI ALAOUI
TIM Lab, Faculty of Sciences Ben M'sik, Hassan II University
Casablanca, Morocco

Saïd NOUH
TIM Lab, Faculty of Sciences Ben M'sik, Hassan II University
Casablanca, Morocco

Abdelaziz MARZAK
TIM Lab, Faculty of Sciences Ben M'sik, Hassan II University
Casablanca, Morocco

***Abstract*—BCH codes have high error correcting capability which allows classing them as good cyclic error correcting codes. This important characteristic is very useful in communication and data storage systems. Actually after almost 60 years passed from their discovery, their weights enumerators and therefore their analytical performances are known only for the lengths less than or equal to 127 and only for some codes of length as 255. The Partial Weights Enumerator (PWE) algorithm permits to obtain a partial weights enumerators for linear codes, it is based on the Multiple Impulse Method combined with a Monte Carlo Method; its main inconveniece is the relatively long run time. In this paper we present an improvement of PWE by integration of Hash techniques and a part of Automorphism Group (PWEHA) to accelerate it. The chosen approach applies to two levels. The first is to expand the sample which contains codewords of the same weight from a given codeword, this is done by adding a part of the Automorphism Group. The second level is to simplify the search in the sample by the use of hash techniques. PWEHA has allowed us to considerably reduce the run time of the PWE algorithm, for example that of PWEHA is reduced at more than 3900% for the BCH (127,71,19) code. This method is validated and it is used to approximate a partial weights enumerators of some BCH codes of unknown weights enumerators.**

***Keywords*—Partial weights enumerator; PWEHA; automorphism group; hash function; hash table; BCH codes***

## I. Introduction

The growth use of computer networks, telecommunication systems and data storage in our societies shed lights on the problem of digital transmission of information where a major problem is the preservation of the entire initial information through its transmission process. Many examples illustrate this problematic; starting with a message transmission via communication systems, where the message can be changed by noise in transmission channels. A second example concerning storage this time; is the alter of data obtained from an optical disk because of stripes or reading lens jump (when there is a sudden movement).

A binary linear code is generally denoted by C (n, k, d) where n is its length, k is its dimension and d is its minimum distance and by its rate $R=\frac{k}{n}$. The BHC codes [1-2], as a class, are one of the most known powerful error-correcting cyclic codes due to their error-correcting capability and efficient coding and decoding algorithms. The most common BCH codes are characterised as follows: specifically, for any positive integer $m \geq 3$, and $t<2^{m-1}$, there exists a binary BCH code with the following parameters:

- Block length: $n=2^m -1$
- Number of message bits: $k \leq n-mt$
- Minimum distance: $d \geq 2t+1$

These BCH codes are called primitive because they are built using a primitive element of $GF(2^m)$.

Error-correcting codes are more used to detect and correct data transmission errors. Before using a correcting code it is important to know its analytical performances which require prior determination of its weights enumerator represented by the polynomial $A(x)= \sum_{i=0}^{n} A_i x^i$, where $A_i$ is the number of codewords of length n and weight i over C(n,k,d).

The enumeration of codewords is not easy, especially for codes with a relatively large dimension. Despite of all the methods developed by researchers in this field, the weights enumerators are availables only for relatively small dimensions and/or co-dimensions. For example the weights enumerators of BCH (Bose, Ray-Chaudhuri et Hocquenghem) codes are determined only for lengths less than or equal to 127 and only for some codes of length 255.

The channel coding technique is based on information redundancy added to detect or correct errors that might be generated by a less reliable communication channel. Decoding algorithms try to find the transmited codeword as illustrated in Fig. 1.

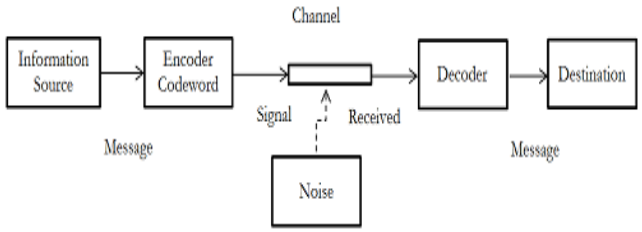


Fig. 1. Communication system model.

The importance of weight distribution is that it allows measuring the probability of non-detection of an error of the code [3]. The polynomial A gives important information about analytical performances of C in terms of errors detection and correction [4]. For a linear block code over a Binary Symmetric Channel (BSC) with an inversion probability p, the upper bound of decoding error probability [5] is given by the expression (1).

$$P_e(C) \leq \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \quad (1)$$

Where, t is the code correcting capacity.

Proakis [6] exposes that the inversion probability p can be formulated as in (2):

$$p = Q\left(\sqrt{2R\frac{E_b}{N_0}}\right) \text{ and } Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-z^2/2}\, dz \quad (2)$$

Where, R represent the code rate ($R = \frac{k}{n}$) and $\frac{E_b}{N_0}$ represents the ratio signal/noise.

On a Gaussian channel AWGN (Additive white Gaussian noise), an upper bound about decoding error probability [5] is given by (3).

$$P_e(C) \leq \sum_{w=d}^{n} A_w Q\left(\sqrt{2wR\frac{E_b}{N_0}}\right) \quad (3)$$

Where, $A_w$ represent the number of codewords of weight w, we note that always $A_0 = A_n = 1$.

Moreover, Fossorier et al. [7] demonstrated that for a systematic linear block code over a decoded AWGN channel by the Maximum Likelihood Decoder (MLD) algorithm, the binary error probability $P_e(C)$ has the following upper bound (4):

$$P_e(C) \leq P_a = \sum_{w=0}^{n} \frac{wA_w}{n} Q\left(\sqrt{2wR\frac{E_b}{N_0}}\right) \quad (4)$$

The bound $P_a$ represents the analytic performances over the AWGN channel for the code C.

The polynomial (5) is called the partial weight enumerator of radius p of the code C having the weight enumerator A, where p is a positive integer less than n-d [8].

$$A^p(x) = 1 + \sum_{i=d}^{p+d} A_i X^i \quad (5)$$

The remainder of this paper is organized as follows. In the next section, we present some related works. In Section 3, we present the proposed method PWEHA. In Section 4, we validate the method PWEHA, we compare it with PWE, we give their results for the BCH (255, 191, 17), BCH (255, 187, 19), BCH (255, 179, 21) and BCH (255, 171, 23) codes of unknown weights enumerators and we plot their corresponding analytical performances. Finally, a conclusion and a possible future direction of this research are outlined in Section 5.

## II. Related Works

In [9], the authors determine the dimension, the minimum distance and the weight enumerators for BCH codes under some conditions and for well-defined cases; in an other work [10], the authors gave a study of dimension for three type of BCH codes over a finite field of order q (GF(q)). In [11], the authors propose a study of the minimum distance of a binary cyclic code of length $n=2^m-1$ and the weight divisibility of its dual code. Based on directed graphs, the authors of [12] have developed combinatorial algorithms for computing parameters of extensions of BCH codes. In [13], [14] the authors propose the use of the complete weights enumerator in order to deduce the weights enumerators for linear codes; also they employed these codes to construct systematic authentication codes with new parameters.

In [8], the authors used genetic algorithms combined with a Monte Carlo method to find the weights enumerator for some residue quadratic codes. In a second work [15], the authors have combined the Monte Carlo method with the multiple error impulse (MIM) technique [16], [17] to find the partial weights enumerator (PWE) for some binary linear codes, in consequences they obtained an upper bound of error probability for MLD decoder for a shortened BCH codes: BCH (130, 66), BCH (103, 47) and BCH (111, 55).

In [18] we have defined a new method called PWEH, this method has obtained by integration of the hash techniques in the PWE [15] in order to reduce its run time; with PWEH we have found the partial weights enumerator of BCH (255, 199, 15) code.

Monte Carlo methods are generally used to approximate a value which is difficult or even impossible to calculate with a mathematical formula. Let X, be a random variable that admits an average $\bar{X}$ and a variance $\sigma^2$. When the list of all possible values of X is very large, the compute of $\bar{X}$ is practically impossible and Monte Carlo methods [19] allows to estimate its unknown value by a random process.

The average value of $\bar{X}$ can be calculated by (6).

$$\bar{X} = \frac{1}{q} \sum_{k=1}^{q} X_k \quad (6)$$

Where (X1, X2,…,Xq) are the samples of the same law.

The confidence interval $[\bar{X}_q - \varepsilon_q\ ; \bar{X}_q + \varepsilon_q]$ contains the value of $\bar{X}$ with the precision μ where:

$$\varepsilon_q = \frac{\beta\sigma(X)}{\sqrt{q}} \quad (7)$$

The standard deviation of X is

$$\sigma(X) = \sqrt{\frac{1}{q-1} \sum_{i=1}^{q} (\bar{X} - X_i)^2} \quad (8)$$

β is the solution of the equation:

$$\frac{2}{\sqrt{2\pi}} \int_{\beta}^{\infty} e^{-\frac{u^2}{2}} du = 1 - \mu \quad (9)$$

The partial weight enumerator of order p of a linear code C can be obtained by finding for each weight w ($d \leq w \leq d+p$) the number $A_w = |C_w|$ of all codewords of weight w in C. Where the symbol | . | denotes the cardinal.

The main idea in [15] is to look for a list $L_w$ of codewords of the same weight w by using the error impulse method [16], [17] with the ordered statistic decoder [20] as given in the algorithm A1. The list $L_w$ is used to approximate the value $A_w$ by a Monte Carlo method as given in the algorithm A2.

**Algorithm A1: Construction of a list $L_w$ of codewords of weight w**

```
1   Inputs:
2   G: The generator matrix of the code C(n, k, d)
3   w: The corresponding weight.
4   L: The number of codewords to find
5   Outputs:
6   Lw : a random set of codewords of the weight w
7   Begin
8     S←0;
9     Lw←Empty list;
10    While (S <L) do:
11        Drawn at random a codeword c of weight w by using the MIM method on the matrix G
12        For i=1 to n do
13            If c not in Lw then
14                insert c in the list Lw
15                S←S+1
16                c←cyclic permutation of c
17            End If
18        End For
19    End While
20  End
```

The following algorithm gives an approximation of number $A_w$:

**Algorithm A2: Approximation of the number $A_w$**

```
1   Inputs:
2   w: the corresponding weight.
3   Lw : a random set of codewords of the weight w
4   M : minimum number of intern code words
5   Outputs:
6   Approximate value of the number Aw of all codewords of weight w in C
7   Begin
8       S←0;
9       i←0;
10      While (S < M) do:
11          i←i+1;
12          Drawn at random a codeword c of weight w
13          If c in Lw then
14              S←S+1;
15          End If
16      End While
17      R (Lw) ← S/i
18      Aw← |Lw| / R(Lw)   ;
19  End
```

## III. PROPOSED METHOD PWEHA

In order to decrease the run time of the algorithm A1 we propose to use a large part of the Automorphism Group instead of only cyclic permutations in the line number 16. The algorithm A1 becomes A3.

The automorphism group of BCH codes [21] contains the sub group generated by the following permutations:

- V: $y \rightarrow y+1$
- S: $y \rightarrow 2^i.y$, with i=1,2,...,m-1.

Each element of this group is called a stabilizer of the corresponding BCH code.

**Algorithm A3: Construction of a list $L_w$ of codewords of weight w using a part of the Automorphism Group**

```
1   Inputs:
2   G: The generator matrix of the code C(n,k,d)
3   w: The corresponding weight.
4   L: The number of codewords to find
5   Laut: a set of z permutations from the Automorphisms Group of C.
6   Outputs:
7   Lw : a random set of codewords of the weight w
8   Begin
9       S←0;
10      Lw←Empty list;
11      While (S <L) do:
12          Drawn at random a codeword c of weight w by using the MIM method on the matrix G
13          For i=1 to Laut do
14              If c not in Lw then
15                  insert c in the list Lw
16                  S←S+1
17                  c←σ(c), with σ is the i^th element of Laut
18              End If
19          End For
20      End While
21  End
```

In the algorithm A2 the hardest step (HS) is that presented in the line 13 to verify if the $L_w$ list contains or not the codeword c randomly pull out in the step represented in the line 12. The step HS is repeated many times and therefor it increases the PWE run time. In order to decrease this run time we have proposed to use the Hash methods for accelerating the research in the step HS [18].

The Hash method [22], [23] is based on the definition of a Hash function and a Hash table.A Hash function is a particular function that, from given information in the input (key), calculates a Hash value that allowed to gives the position of the element we are looking for in the Hash table. The Hash table is a data structure that permits an association between the key and the corresponding value.

Generating a Hash value from a key can causes a collision problem; we can find that two different keys, maybe more, could have the same Hash value which means the same element of the table. To decrease such risks, we should carefully define the Hash function.

Let N be a positive integer that represents the size of the Hash table. The set $L_w$ presented in the algorithm A3 contains many codewords (only information part) of weight w. This set is divided on N sub-sets; each one contains the words of the same Hash value given by the Hash function presented in the algorithm A4.

**Algorithm A4: The used Hash function**

```
Function hash (word, N)
    Pos←0
    For i=1 to the dimension k of the code
        If word [i] =1 then
            Pos←Pos + i ;
        End If
    End For
    Return (Pos modulo N)
End Function
```

After the use of Hash techniques the algorithm A2 has become algorithm A5. In Fig. 2 we explains the Hash process used in A5 algorithm. In the construction step of the set $L_w$, for each found codeword c of weight w, the hash value h=Hash(c, N) is computed. The information part of c is therefore inserted in the sub-set number h. So, the set $L_w$ is implemented as table of three dimensions in practice.

**Remark 1:** When the encoding is systematic, only the information parts of codewords are stored in the list $L_w$.

**Remark 2:** In the construction step of $L_w$ in the line number 13 of the A1 algorithm, before adding a word c it should verify that c doesn't already exists in $L_w$. Here also the use of the hash technique permits to decrease considerably the run time of this construction.

**Algorithm A5: Approximation of the number $A_w$ with Hash techniques**

```
1   Inputs:
2   w: the corresponding weight.
3   Lw: a random set of codewords of the weight w divided on N
    sub-sets.
4   M :minimum number of intern code words
5   Outputs:
6   Approximate value of the number Aw of all codewords of
    weight w in C
7   Begin
8       S←0;
9       i←0;
10      While (S < M) do:
11          i←i+1;
12          Drawn at random a codeword c of weight w
13          h←hash(c,N)
14          If c in the sub-set Lw of number h then
15              S←S+1;
16          End If
17      End While
18      R (Lw) ← S/i ;
```

19 $A_w \leftarrow \frac{|Lw|}{R(Lw)}$ ;

20 End

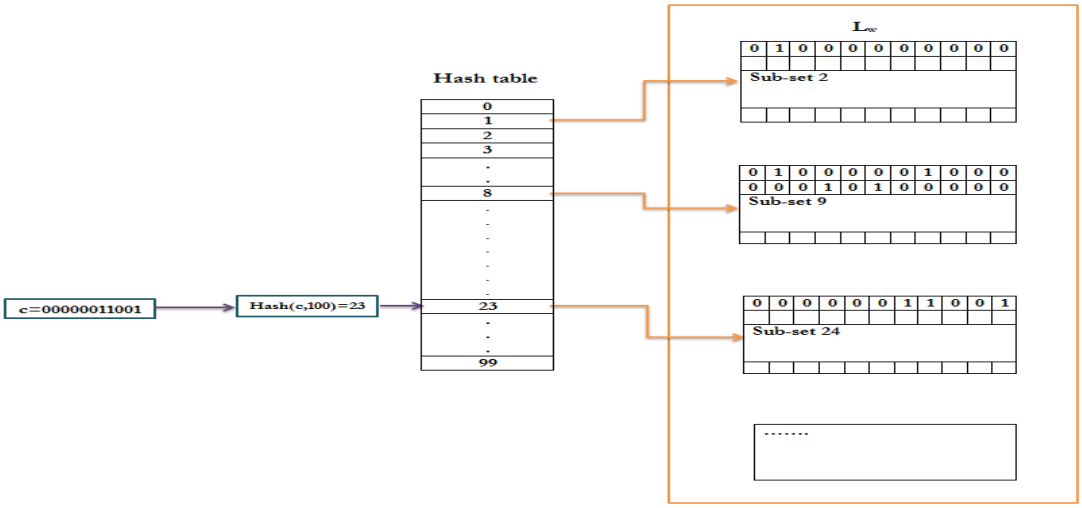


Fig. 2. Example of the Hash process with a hash table of dimension 100.

## IV. Validation of the Proposed Method PWEHA, New Results and Discussion

### A. Validation of the Proposed Method PWEHA

To validate the proposed PWEHA method, we have used it to find partial weights enumerator of the BCH(127, 78) code using 889 stabilizers. Table 1 summarizes the obtained results. The weights enumerator of this code is known and it is available at [24]. The comparison between the approximate values of $A_w$ obtained by PWEHA and the corresponding exact values given in the browser [24] shows that all approximate values found are in the confidence interval which allows us to validate the proposed method successfully. Therefore PWEHA can be used to approximate the weights enumerator of other BCH codes for which theses metrics are still unknown.

TABLE I. Validation of the PWEHA Method

| Code | w | \|Lw\| | The recovery rate R | The standard deviation σ | The exact value of Aw | The approximate value of Aw by PWEHA | I (Aw) |
|---|---|---|---|---|---|---|---|
| BCH(127, 78) | 15 | 20 000 | 0.403 | 0.449 | 48 387 | 49567 | [34 045;56 098] |
| | 16 | 30 000 | 0.089 | 0.064 | 338 709 | 335908 | [290 878; 430 234] |
| | 17 | 30 000 | 0.038 | 0.019 | 768 096 | 772987 | [678 435;879 087] |

TABLE II. Comparison between the Methods PWEHA and PWE

| Code | Weight | Run time of the PWE method(in seconds) | | | Run time of the PWEHA method(in seconds) | | |
|---|---|---|---|---|---|---|---|
| | | *Time required to complete the listLw* | *Time required to estimate the value of Aw* | *Total execution time of the PWE algorithm* | *Time required to complete the list Lw* | *Time required to estimate the value of Aw* | *Total execution time of the PWE algorithm* |
| BCH(127, 71) | 19 | 24704.01 | 19202.22 | 43906.23 | 96.23 | 1019.63 | 1115.86 |
| | 20 | 55649.78 | 119567.89 | 175217.67 | 285.57 | 18856.5 | 19142.07 |
| | 21 | 62765.89 | 618917.78 | 681683.67 | 936.91 | 142347.5 | 143284.41 |

### B. Comparison between the Methods PWEHA and PWE

Table 2 gives a comparison between the run time of PWEHA and PWE for BCH (127, 71) using a simple configuration computer: Intel (R) Core(TM) 2 Duo CPU T9600 @2.8GHz, 2 GB of RAM.

With M=10, β=2.57, q=100, z=889, N=100 and |Lw|=100 000. From the results presented in Table 2, we note that the time required to fill the list $L_w$ is much reduced (more than 256 times for the weight 19) with the use of a part of the Automorphism Group, this is justified by:

In the case where cyclic permutations (algorithm A1) are used and from a codeword of length n extracted ($n$ = 127 in this case), just $n$ other codewords can be deduced. Contrariwise, using a part of the Automorphism Group (algorithm A3) and from a codeword of length $n$, we can deduce up to $n * m$ (889 in this case) other codewords, which justifies the large difference between the execution time of the two algorithms.

Similarly, a simple comparison between the run time of the algorithm for estimating the value of $A_w$ with and without hash techniques shows that there is a large difference in favor of the algorithm A5 where there is the hash techniques (reduction at more than 18 times for the weight 19). This rapidity is quite normal, since that without using hash and for every found codeword it is necessary to browse the list $L_w$ to check if it contains it or not. The repetition of this task at several times makes the algorithm A2 very heavy and influences its run time. On the other hand, with the use of the hash techniques, for each found codeword, it will not be necessary to traverse the entire list $L_w$ each time, but only the part of $L_w$ which corresponds to the value returned by the hash function.

The comparison of the total run time of the two algorithms shows that the use of the PWEHA method allows to considerably reduce the run time, for example for the weight 19, this is reduced by more than 3900% comparing to the PWE.

### C. New Results of theMethod PWEHA

The integration of Hash techniques and the use of a large part of the Automorphism Group that we added in the PWE method allowed us to reduce considerably the run time. In this section we present the results of PWEHA for the BCH(255, 191, 17), BCH(255, 187, 19), BCH(255, 179, 21) and BCH(255,171,23) codes where the weights enumerators are still unknown. Table 3 summarizes the results corresponding to the parameters M=10, β=2.57, q=100, z=2040 and N=1000.

The obtained partial weights enumerators of the BCH(255,191,17), BCH(255,187,19), BCH(255,179,21) and BCH(255,171,23) codes are used to plot their analytical performances given in Fig. 3 corresponding to (4) and (5).

TABLE III. RESULTS FOR BCH(255,191,17), BCH(255,187,19), BCH(255,179,21) AND BCH(255,171,23) CODES

| Code | w | \|Lw\| | The recovery rate R | The standard deviation σ | The approximate value of Aw by PWEHA | I (\|Aw\|) |
|---|---|---|---|---|---|---|
| BCH (255,191) | 17 | 1 000 000 | 0.579 | 0.125 | 1 724 773 | [1 633 701 ; 1 826 598] |
| | 18 | 7 000 000 | 0.460 | 0.116 | 15 188 984 | [14 261 044 ; 16 246 087] |
| | 19 | 7 255 001 | 0.445 | 0.103 | 16 298 008 | [15 381 826 ; 17 330 243] |
| BCH (255,187) | 19 | 3 318 639 | 0.426 | 0,112 | 7 779 558 | [7 284 318 ; 8 347 050] |
| | 20 | 4 469 231 | 0.129 | 0,041 | 34 548 438 | [31 926 905 ; 37 638 994] |
| | 21 | 2 746 038 | 0.007 | 0,002 | 382 761 276 | [347 838 417 ; 425 479 248] |
| BCH (255,179) | 21 | 1 254886 | 0.565 | 0.100 | 2 220 330 | [2 123 233 ; 2 326 734] |
| | 22 | 2 865 315 | 0.386 | 0.103 | 7 411 309 | [6 934 650 ; 7 958 332] |
| | 23 | 6 000 001 | 0.149 | 0.183 | 40 095 792 | [30 503 715 ; 58 487 553] |
| BCH (255,171) | 23 | 1 069 174 | 0.533 | 0.429 | 1 660 080 | [1 417 324 ; 2 003 181] |
| | 24 | 3 218 071 | 0.583 | 0.125 | 5 517 063 | [ 5 228 811; 5 838 950] |
| | 25 | 6 000 001 | 0.351 | 0.202 | 17 085 515 | [14 883 174 ; 20 052 838] |

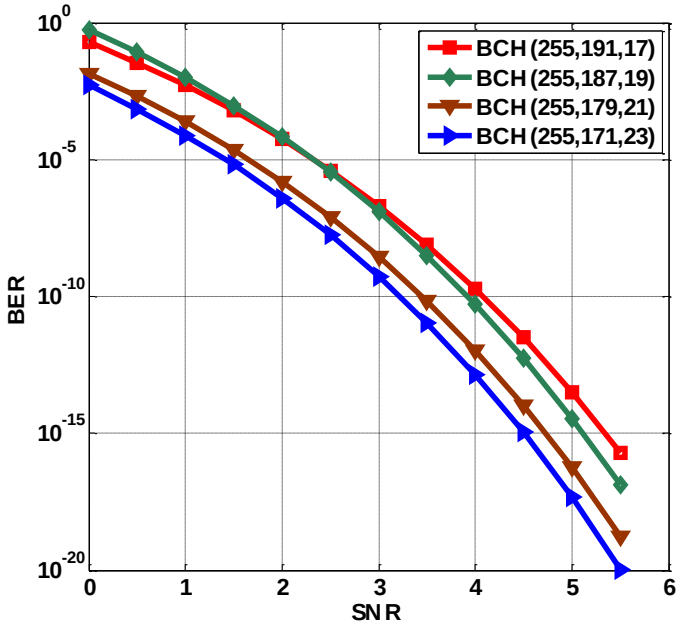


Fig. 3. The analytical performances obtained by PWEHA for the BCH(255,191,17), BCH(255,187,19), BCH(255,179,21) and BCH(255,171,23) codes.

## V. Conclusion and Perspectives

In this work, we have studied the impact of using Hash techniques and adding a large part of the Automorphism Group in the PWE algorithm. The found results are very important in terms of run time and solution quality. This important improvement will help us to find the weights enumerators of many other linear codes of unknown weights distribution. In the perspectives, we will expand the used part of the Automorphism Group in BCH codes and other linear codes like Quadratic Residue and LDPC codes.

## References

[1] Hocquenghem. Codes correcteursd'erreurs.Chiffres, 2 :147–156, sept 1959.

[2] R.C Bose, and D. K. Ray-Chaudhuri."On a class of error correcting binary group codes. Informationand Control", 3 :68–79, mars 1960.

[3] BrocheroMartínez F.E.,Giraldo Vergara C.R.: "Weight enumerator of some irreducible cyclic codes", 2014.

[4] Clark G.C., and Cain J.B., "Error-Correction Coding for Digital Communications", first edition Springer, New York, 30 June, 1981.

[5] Robert H. Morelos-Zaragoza. "The art of error correcting coding, John Wiley & Sons Second Edition", 2006.

[6] J.G. Proakis. "Digital communications 5th edition". 2001.

[7] M. P. C. Fossorier, S. Lin, and D. Rhee. "Bit-error probability for maximum-likelihood decoding of linear block codes and related soft decision decoding methods". IEEE Transaction on Information Theory, 44: 3083-3090, November 1998.

[8] S. Nouh, and M. Belkasmi. "A genetic algorithm for finding the weight enumerator of Binary linear block codes. International Journal of Applied Research on Information Technology and Computing". 2, December 2011.

[9] C. Ding, C. Fan, and Z. Zhou, "The dimension and minimum distance of two classes of primitive BCH codes", pp 237-263, Vol. 45, 2017.

[10] H. Liu, C. Ding, and C. Li. "Dimensions of three types of BCH codes over GF(q)". pp 1910–1927, Vol. 340, 2017.

[11] X. Zeng, J. Shan, and L. Hu. "A Triple-Error-Correcting Cyclic Code from the Gold and Kasami-Welch APN Power Functions", arXiv:1003.5993, 2012.

[12] A.V. Kelarev, "Algorithms for computing parameters of graph-based extensions of BCH codes", Journal of Discrete Algorithms Vol. 5, pp 553–563, 2007.

[13] Wang, X., Gao, J., and Fu, FW."Complete weight enumerators of two classes of linear codes", Cryptogr. Commun. 9: 545. doi:10.1007/s12095-016-0198-1, 2017.

[14] Yang, S. & Yao, ZA., "Complete weight enumerators of a family of three-weight linear codes", Des. Codes Cryptogr. 82: 663. doi:10.1007/s10623-016-0191-x, 2017.

[15] S. Nouh, B. Aylaj, and M. Belkasmi. "A method to determine partial weight Enumerator for linear block codes". Computer Engineering and Intelligent Systems 3, October 2012.

[16] M. ASKALI, S. NOUH, and M. Belkasmi."An Efficient method to find the Minimum Distance of Linear Codes", International Conference on Multimedia Computing and Systems proceeding, May 10-12, Tangier, Morocco, 2012 .

[17] M. ASKALI, A. AZOUAOUI, S. NOUH, and M. BELKASMI. "On the computing ofthe minimum distance of linear block codes by heuristic methods". International Journal of Communications,Network and System Sciences, N° 11, Vol 5, 2012.

[18] S. El Kasmi Alaoui, S. Nouh, and A. Marzak. "Determination of partial weight enumerators of BCH codes by Hash methods", IEEE Wireless Technologies, Embedded and Intelligent Systems (WITS), International Conference on, 2017.

[19] D. P. Kroese, T. Taimre, and Z.I. Botev. "Handbook of monte carlo methods". 2011.

[20] Fossorier M.P.C. and lin S "Soft decision decoding of linear block codes based on ordered statistics", IEEE Trans. information theory Vol. 41, pp. 1379-1396. Sep, 1995.

[21] T. P. Berger and P. Charpin, "The automorphism groups of BCH codes and of some affine-invariant codes over extension fields", Design, Codes and Cryptography. Vol. 18, Issue 1-3, pp 29-53, 1999.

[22] Cormen, C., Leiserson, C. E., Rivest, R. L., and Stein, C. 2001. "Introduct. Algorithms". 2nd Ed. MIT Press.

[23] A.Andoniand P.Indyk. "Near-optimal hashing algorithms for approximate nearest neighbour in high dimensions". In FOCS, pages 459–468. IEEE, 2006.

[24] http://www.ec.okayama-u-ac.jp/~infsys/kusaka/wd/index.html, created by M. Terada, J. Asatani and T. Koumoto.