Journal of mathematics and computer science 12 (2014), 271-281

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

# Error Detection Mechanism based on BCH Decoder and Root Finding of Polynomial over Finite Fields

Saeideh Nabipour[1], Javad Javidan[2], Gholamreza Zare Fatin[3]

[1,2,3]*Technical Engineering Department, University of Mohaghegh Ardabili, Ardabil, Iran*

[1]***saeidehnabipour@yahoo.com***

**Article history:**
**Received July 2014**
**Accepted August 2014**
**Available online August 2014**

***Abstract***

Error Correction Code is very important in modern communication systems. BCH (Bose, Chaudhuri, and Hocqunghem) codes are being widely used in variety communication and storage systems. In this paper the construction and decoding BCH codes which are based on finite field arithmetic is introduced and also an improved algorithm for finding roots of polynomials over finite fields is proposed. This makes possible significant speed up of the decoding process of BCH codes.

**Keywords:** Error Correction Code, BCH code, root-finding polynomial, Chien Search, BRS algorithm.

## 1. Introduction

The theory of error detecting and correcting codes is that branch of engineering and mathematics which deals with the reliable transmission and storage of data. In 1948 Claude Shannon's article "A Mathematical Theory of Communication" gave birth to the two twin disciplines: information theory and coding theory. The article specifies the meaning of efficient and reliable information and, there, the very well known term "bit" has been used for the first time. Anyway, it was only with Richard Hamming in 1950 that a constructive generating method and basic parameters of Error Correction Codes (ECC) were defined [1].

The success of the coding theory is based precisely on the motivation arisen from the manifold practical applications. Space communication would not have been possible without the use of error-correcting codes and the digital revolution has made a great use of them. Modems, CDs, DVDs, MP3 players and USB keys need an ECC which enables the reading of information in a reliable way.

Information media are not 100% reliable in practice, in the sense that noise (any form of interference) frequently causes data to be distorted. To deal with this undesirable but inevitable situation, some form of redundancy is incorporated in the original data. With this redundancy, even if errors are introduced (up to some tolerance level), the original information can be recovered, or at least the presence of errors can be detected.

Error correction plays a major role in communication and storage systems to increase the transmission reliability and achieve a better error correction performance with less signal power. Several error correction codes have been proposed in the literature, including Hamming based block codes, Reed-Solomon codes, Bose-Chaudhuri-Hocqunghem (BCH) codes, Goppa codes, Golay codes, etc. BCH codes have recently received a lot of attention because of their superiror error correction performance. BCH codes form a large of powerful random error correcting cyclic codes. This class of codes is a remarkable generalization of the Hamming codes for multiple error correction. Binary BCH codes were discovered by Hocquenghem in 1959 and independently by Bose and Chaudhuri in 1960. The cyclic structure of these codes was proved by Peterson in 1960 [1]. BCH codes operate over finite fields or Galois fields. The biggest advantage of BCH codes is the existence of efficient decoding methods due to the special algebraic structure introduced in the codes. The conventional BCH decoder contains three major blocks, i.e., syndrome calculator, error locator calculator, and Chien search. It is well known that one of most time-consuming stages of decoding BCH codes is finding roots of the error-locator polynomial using Chien search method. In this paper the construction and decoding BCH codes is introduced and also an improved algorithm for finding roots of polynomials over finite fields is proposed. This makes possible fast decoding process of BCH codes.

The rest of this paper is organized as follows. Section II and III introduces the details of the Finite Fields and the t-error-correction BCH coding scheme. Section IV and V describes BCH encoding and decoding procedure. The root finding technique based on the BRS algorithm and Chien search method is introduced in Section VI. Finally, the concluding remarks are given in Section VII.

## 2. Finite Fields

BCH codes are based on finite field arithmetic which involves defining closed binary operations over finite sets of elements. Finite field arithmetic is enormously used in several area of digital signal processing. It is, in fact, a special case of abstract algebra [1]. As a brief overview, we will start with the simplest example of finite field which is the binary field consisting of the elements {0,1}. Traditionally referred to as *GF(2)*, the operations in this field are defined as integer addition and multiplication reduced modulo 2. We can create larger fields by extending *GF(2)* into vector space leading to finite fields of size $2^m$. These are simple extensions of the base field *GF(2)* over m dimensions. The field $GF(2^m)$ is thus defined as a field with $2^m$ elements each of which is a binary m-tuple. Using this definition, we can group m bits of binary data and refer to it as an element of $GF(2^m)$. This in turn allows us to apply the associated mathematical operations of the field to encode and decode data. There are two elements (but equivalent) representations for the field elements. First, all nonzero elements in $GF(2^m)$ may be represented as powers of a primitive field element $\alpha$ (i.e. each nonzero element is of the form $\alpha^n$ for $n = 0, 1, \ldots, 2^m - 1$). Second, each element has an equivalent representation as a binary m-tuple. While the $\alpha^n$ representation has great mathematical convenience, digital hardware prefers the binary m-tuple representation. These representations for $GF(2^3)$ are illustrated in Table 1.

**Table 1**. Canonical Representation of Finite Field $GF(2^3)$

| Power Representation | 0 | $\alpha^0$ | $\alpha^1$ | $\alpha^2$ | $\alpha^3$ | $\alpha^4$ | $\alpha^5$ | $\alpha^6$ |
|---|---|---|---|---|---|---|---|---|
| Vector Representation | 000 | 100 | 010 | 001 | 110 | 011 | 111 | 101 |
| Polynomial Representation | 0 | 1 | $\alpha$ | $\alpha^2$ | $1+\alpha$ | $\alpha+\alpha^2$ | $1+\alpha+\alpha^2$ | $1+\alpha^2$ |

## 3. Basics of BCH Codes

Bose-Chaudhuri-Hocqunghem (BCH) codes are cyclic codes for which a large number of block sizes and error correction capabilities are available. While operating under $GF(2^m)$, it has error correcting capability of t. The main parameters of BCH codes are summarized as following parameters:

Block length: $n = 2^m - 1$

Number of information bits: $k \geq n - m * t$

Minimum distance: $d_{\min} \geq 2t + 1$

The generator polynomial of the code is specified in terms of its roots over the Galois field $GF(2^m)$ which is explained in [1]. Let $\alpha$ be a primitive element in $GF(2^m)$. The generator polynomial g(x) of the code is lowest degree polynomial over $GF(2)$, which has $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ as its roots. $[g(\alpha^i) = 0, \quad 1 \leq i \leq 2t]$. Let $\phi_i(x)$ be the minimal polynomials of $\alpha^i$ then generator polynomial of BCH code is the least common multiple (LCM) of the minimal polynomials of $\alpha^i$:

$$g(x) = LCM\{\phi_1(x), \phi_2(x), \ldots\ldots \phi_{2t}(x)\} \tag{2}$$

Assume a *k*-bit message is encoded to form an *n*-bit codeword. The *k*-bit message is the input of encoder and the BCH encoder generates *(n-k)*-bit parity. After encoding, the *(n-k)*-bit parity together with *k*-bit message form a codeword which is stored into the memory. In decoding process, the *n*-bit data is retrieved from the memory and is put into the syndrome calculator block. If there is no error in the retrieved data, the syndrome should be all zero and the decoding procedure is finished. Otherwise the syndrome should be sent to error-location block in order to generate the error-locating polynomial. Then the Chien search block is used to find out which bits are erroneous. Finally, the corrected message extracted.

## 4. Description of BCH Encoding

Since BCH codes are cyclic codes, encoding in systematic form is similar to the binary encoding procedure. The generating polynomial for BCH code is:

$$g(X) = g_0 + g_1 X + g_2 X^2 + \ldots. + g_{2t-1} X^{2t-1} + X^{2t} \tag{3}$$

The degree of generator polynomial is equal to the number of parity bit. Since the generator polynomial is equal to degree *2t*, there must be precisely *2t* successive powers of $\alpha$ that are roots of the polynomial. Message polynomial *m(X)* can be shifted into the leftmost *n-k* stages of a codeword register and then appending a parity polynomial *p(X)*. Therefore following steps are followed for encoding:

**Step1:** Multiply *m(X)* by $X^{n-k}$ thereby manipulating the message polynomial algebraically so that it is left-shifted *n-k* positions.

**Step2:** Divide $X^{n-k}m(X)$ by generator polynomial $g(X)$, which is written in following form:

$$X^{n-k}m(X) = q(X)g(X) + p(X) \tag{4}$$

Where $q(X)$ and $p(X)$ are quotient and remainder polynomials, respectively. The remainder polynomial represents the parity bits. Equation (2) can also be expressed as:

$$p(X) = X^{n-k}m(X) \bmod g(X) \tag{5}$$

**Step3:** The resulting codeword polynomial, $C(X)$ can be written as:

$$C(X) = p(X) + X^{n-k}m(X) \tag{6}$$

## 5. Description of BCH Decoding

The decoding of BCH code has three main steps that are expressed as follows:

I. Computing the syndromes from the received codeword.

II. Key equation solver, which determines the error locator polynomial $\sigma(X)$ through the BM (Berlekamp-Massey) algorithm.

III. Determining the error locating numbers by finding the roots of error locating polynomial (identifying the position of erroneous bit).

### 5.1. Syndrome Calculation

In the BCH decoding process, the received *n*-bit codeword $(r_0, r_1, \ldots, r_{n-1})$ is interpreted as a polynomial, $R(x) = r_0 + r_1x^1 + r_2x^2 + \ldots + r_{n-1}x^{n-1}$. By computing 2t syndromes in (7) the existence of any error in the received codeword can be checked:

$$S_i = R(\alpha^i) = \sum_{j=0}^{n-1} r_j\alpha^{ij} = r_0 + r_1\alpha^1 + \ldots + r_{n-1}\alpha^{i(n-1)}. \tag{7}$$
$$1 \le i \le 2t$$

If the received codeword contains errors, the syndromes are not all zero. The syndromes are a set of the field elements in GF($2^m$). Therefore, each syndrome component is calculated by dividing r(x) by the minimal polynomial $\phi_i(x)$ of $\alpha^i$:

$$r(X) = q_i(X)\phi_i(X) + b_i(X) \tag{8}$$

$\phi_i(x)$ is the minimal polynomial and $b_i(X)$ is reminder, So by evaluating $b_i(X)$ with $X = \alpha^i$, the syndrome components can be found. Since,

$$\phi_i(\alpha^i) = 0$$

We have the following equation,

$$S_i = r(\alpha^i) = b_i(\alpha^i)$$

It can be seen that the 2t syndrome components $S_1, S_2, \ldots, S_{2t}$ can be calculated by substituting the field element $\alpha, \alpha^2, \ldots, \alpha^{2t}$ into the received polynomial *r(x)* in decoding a *t*-error-correcting BCH code.

### 5.2. Error Locator Polynomial Calculation

Suppose there are v errors in the codeword at location $X^{j1}, X^{j2}, \ldots, X^{jv}$ locations. Then, the error polynomial *e(X)* can be written as $e(X) = e_{j1}X^{j1} + e_{j2}X^{j2} + \ldots + e_{jv}X^{jv}$. The indices *1, 2, …, v* refer to first, second and *v*th errors, whereas the index *j* refers to error location. To correct the corrupted codeword, each error value $e_{jl}$ and its location $X^{jl}$, where $l = 1, 2, \ldots, v$ must be determined. Based on *2t* syndromes, we calculate the error locator polynomial as $\sigma(x) = 1 + \sigma_1 X + \sigma_2 X^2 + \ldots + \sigma_t X^t$ using the Berlekamp-Massey (BM) algorithm [6]. Conventionally, the iterative BM algorithm is employed to obtain the coefficients of $\sigma(X)$ in the key equation solver. The procedure of BM algorithm is illustrated in Table 2. The basic principle of this algorithm is that it compares the polynomial computed in the current cycle with that from the previous cycle and determines the polynomial in the next cycle [7]. It contains of two major tasks including: discrepancy calculation, $\delta$, and error location polynomial update. We assume that the numbers of errors $v \le t$ have occurred and the error locator polynomial $\sigma(X)$ is:

$$\begin{aligned} \sigma(X) &= \sigma_0 + \sigma_1 X + \sigma_2 X^2 + \ldots + \sigma_v X^v \\ \sigma(X) &= (1 + \beta_1 X)(1 + \beta_2 X)\ldots(1 + \beta_v X) \end{aligned} \tag{9}$$

The coefficient of error locator polynomial and the error location numbers are related by the following set of equations:

$$\begin{aligned} \sigma_0 &= 1 \\ \sigma_1 &= \beta_1 + \beta_2 + \ldots + \beta_v \\ \sigma_2 &= \beta_1\beta_2 + \beta_2\beta_3 + \ldots + \beta_{v-1}\beta_v \\ &\vdots \\ \sigma_v &= \beta_1\beta_2 + \ldots + \beta_v \end{aligned}$$

Where the coefficient of error locator polynomial $\sigma_{i, 1 \le i \le v}$ are related to the syndrome components $S_i, 1 \le i \le v+1$. The BMA procedure begins with the *2t* syndrome. With help of syndrome components $(S_1, S_2, \ldots, S_{2t})$ it is possible to determine the coefficients $\sigma_1, \sigma_2, \ldots, \sigma_t$ of the error location polynomial.

**Table 2**. The Berlekamp-Massey Algorithm

$L = 1, \sigma^{(0)}(x) = 1 + S_1 x, \tau^{(0)}(x) = 1$
$for(\mu = 1; \mu < 2t; \mu + +)$
$\delta = \sum_{j=0}^{L} \sigma_j^{(\mu-1)} S_{\mu+1-j}$
$\sigma^{(\mu)}(x) = \sigma^{(\mu-1)}(x) + \frac{\delta}{\gamma} x \tau^{(\mu-1)}(x)$
$if(\delta \neq 0 \quad and \quad 2L < \mu + 1)$
$L = \mu + 1 - L$
$\gamma = \delta$
$\tau^{(\mu)} = \sigma^{(\mu-1)}(x)$
$else$
$\tau^{(\mu)} = \tau^{(\mu-1)}(x)$

### 5.3. Chien Search

Once error location polynomial $\sigma(X)$ was obtained in the decoding process, a Chien search method can be used to exhaustively examine whether $\sigma(\alpha^i) = 0$ for $0 \le i \le n-1$ or not, where

$$\sigma(\alpha^i) = \sum_{j=0}^{t} \sigma_j \alpha^{i^{(j)}} = \sum_{j=1}^{t} \sigma_j \alpha^{ij} + 1. \tag{10}$$

All $2^m$ possible elements of Galios field are substituted into the error polynomial, one after another, and the polynomial is evaluated. If the result is equal to then zero, there will be a root for the polynomial. Chien search circuit produces an error vector e in such a way that, if $\alpha^i$ is a root, then the $(n-i)th$ component $e_{n-i} = 1$; otherwise $e_{n-i} = 0$ for all $0 \le i \le n-1$. Finally, error will be corrected in the received codeword.

## 6. Polynomial Root Finding

It is well known that one of the most time-consuming stages of decoding process BCH codes is finding roots of the error-locator polynomial. The most widely known root finding algorithm is Chien search method, which is a simple substitution of all elements of the field into the polynomial, so it has very high time complexity for the case of large fields and polynomials of high degree. In this section, the Berlekamp-Rumsey-Solomon (BRS) algorithm [2], [3], together with the Chien-search method, is developed in order to find the roots of error locator polynomial for BCH decoders. This fast algorithm makes the root-finding problem quite practical and efficient for BCH decoders.

### 6.1. BRS Algorithm with the Chien Search Method

Berlekamp et al. [3], proposed a novel algorithm for finding the roots of a special class of polynomials, called the *p*-polynomial, [3], [4]. Before description of the algorithm, first consider some definitions and a theorem that are needed to develop this algorithm.

**Definition 1:** The polynomial *L(y)* over $GF(2^m)$ is called a *p*-polynomial for *p=2* if

$$L(y) = \sum_i c_i y^{2i} \tag{11}$$

where $c_i$ are restricted to $GF(2^m)$ and exponents are restricted to be the powers of two. These polynomials are also called linearized polynomials.

**Definition 2:** A polynomial *A(y)* over $GF(2^m)$ is called an affine polynomial if

$$A(y) = L(y) + \beta \tag{12}$$

Where *L(y)* is a *p*-polynomial and $\beta \in GF(2^m)$.

**Theorem 1:** Let $y \in GF(2^m)$ and let $\alpha^0, \alpha^1, \alpha^2, ..., \alpha^{m-1}$ be a standard basis. If *y* is represented in the standard basis, i.e., if

$$y = \sum_{k=0}^{m-1} y_k \alpha^k$$

Where $y_k \in GF(2)$, then

$$L(y) = \sum_{k=0}^{m-1} y_k L(\alpha^k)$$

**Example1:** Solve the quadratic *p*-polynomial over $GF(2^3)$, namely

$$A(y) = y^2 + \alpha^3 y + \alpha^4 = 0$$

or

$$y^2 + \alpha^3 y = \alpha^4 = \alpha^2 + \alpha + 1 \tag{13}$$

Where $\alpha$ is a root of the primitive irreducible polynomial $p(x) = x^3 + x^2 + 1$. The left-hand side of (13), $L(y) = y^2 + \alpha^3 y$, is a p-polynomial over $GF(2^3)$ and (13) can be expressed as:

$$L(y) = \alpha^2 + \alpha + 1 \tag{14}$$

If $y = y_2\alpha^2 + y_1\alpha + y_0 \in GF(2^3)$, then, in accordance with Theorem 1, (14) becomes:

$$y_2 L(\alpha^2) + y_1 L(\alpha) + y_0 L(\alpha^0) = \alpha^2 + \alpha + 1 \tag{15}$$

Where $y_0, y_1, y_2 \in GF(2)$.

In (15), the values of field element $L(\alpha^0), L(\alpha^1)$ and $L(\alpha^2)$ can be calculated as:

$$\begin{aligned} L(1) &= 1 + \alpha^3 = \alpha^2 \\ L(\alpha) &= \alpha^2 + \alpha^3.\alpha = \alpha + 1 \\ L(\alpha^2) &= \alpha^4 + \alpha^3.\alpha^2 = \alpha^2 \end{aligned} \tag{16}$$

A substitution of (16) into (15) yields finally:

$$(y_2 + y_0)\alpha^2 + y_1\alpha + y_0 = \alpha^2 + \alpha + 1. \tag{17}$$

In matrix form, (17) can be expressed as:

$$[y_2 \quad y_1 \quad y_0]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = [1 \quad 1 \quad 1] \tag{18}$$

Observe that the roots of (13) can by found by solving the three simultaneous equations given in (18) with three unknowns $y_0, y_1$ and $y_2$. Evidently solutions of (17) are $y = 011$ and $y = 110$, which are two roots of (17). From this example, one observes that one needs only to compute the three values $L(\alpha^0), L(\alpha^1)$, and $L(\alpha^2)$ instead of all of the values $L(\alpha^0), L(\alpha^1), \ldots, L(\alpha^6)$ needed in the Chien-search method.

The authors in [3] suggested a fast algorithm to find the roots of a polynomial up to degree 11. In their algorithm, the Berlekamp-Rumsey-Solomon (BRS) algorithm together with the Chien-search method, is developed in order to find the roots of error locator polynomial up to degree n = 11. In their recent paper [3] Truong, Jeng and Reed proposed a transformation which allows grouping of some summands of the polynomial of degree not higher than 11 into multiples of affine polynomials. Since affine polynomials can be easily evaluated using very small pre-computed tables, it is possible to speed up computations. in the following, polynomial root finding of degree 6 and 7 is examined.

**Example 2:** Solve the polynomial of degree 6, namely

$$\sigma_6 x^6 + \sigma_5 x^5 + \sigma_4 x^4 + \sigma_3 x^3 + \sigma_2 x^2 + \sigma_1 x^1 + \sigma_0 = 0 \tag{19}$$

or

$$A_6(x) = B_6(x) \tag{20}$$

Where $A_6(x) = \sigma_4 x^4 + \sigma_2 x^2 + \sigma_1 x + 1$ and $B_6(x) = \sigma_6 x^6 + \sigma_5 x^5 + \sigma_3 x^3$.

In (20) since $A_6(x)$ is an affine polynomial over $GF(2^m)$, then the values of $A_6(\alpha), A_6(\alpha^2), \ldots, A_6(\alpha^{2^m-1})$ can be obtained by the use of algorithm illustrated in Example 1. On the other hand, the values of $B_6(\alpha), B_6(\alpha^2), \ldots, B_6(\alpha^{2^m-1})$ can be obtained by the use of Chien-search method. If $A(\alpha^i) = B(\alpha^i)$ for $1 \le i \le 2^m - 1$, then $x = \alpha^i \in GF(2^m)$ is solution of (20). Therefore, by

exhaustively searching all nonzero field elements in $GF(2^m)$ for the roots of (20), there exists a set of field elements such that they satisfy (20). These solutions are the roots of (19).

**Example 3:** Solve the polynomial of degree 7, namely

$$\sigma_7 x^7 + \sigma_6 x^6 + \sigma_5 x^5 + \sigma_4 x^4 + \sigma_3 x^3 + \sigma_2 x^2 + \sigma_1 x^1 + \sigma_0 = 0 \tag{21}$$

A substitution of $x = y + a$ into (21) yields

$$\begin{aligned} &\sigma_7 y^7 + (\sigma_7 a + \sigma_6) y^6 + (\sigma_7 a^2 + \sigma_5) y^5 + (\sigma_7 a^3 + \sigma_6 a^2 + \sigma_5 a + \sigma_4) y^4 \\ &+ (\sigma_7 a^4 + \sigma_3) y^3 + (\sigma_7 a^5 + \sigma_6 a^4 + \sigma_3 a + \sigma_2) y^2 + (\sigma_7 a^6 + \sigma_5 a^4 + \sigma_3 a^2 + \sigma_1) y \\ &+ (\sigma_7 a^7 + \sigma_6 a^6 + \sigma_5 a^5 + \sigma_4 a^4 + \sigma_3 a^3 + \sigma_2 a^2 + \sigma_1 a + \sigma_0) = 0 \end{aligned} \tag{22}$$

If one eliminates the sixth term of (22) by the choice $a = \frac{\sigma_6}{\sigma_7}$, then (22) becomes

$$c_7 y^7 + c_5 y^5 + c_4 y^4 + c_3 y^3 + c_2 y^2 + c_1 y + c_0 = 0 \tag{23}$$

where $c_7 = \sigma_7$ , $c_5 = \sigma_7 a^2 + \sigma_5, c_4 = \sigma_7 a^3 + \sigma_5 a + \sigma_4, c_3 = \sigma_7 \alpha^4 + \sigma_3, c_2 = \sigma_7 a^5 + \sigma_6 a^4 + \sigma_3 a + \sigma_2, c_1 = \sigma_7 a^6 + \sigma_5 a^4 + \sigma_3 a^2 + \sigma_1$ and $c_0 = \sigma_7 a^7 + \sigma_6 a^6 + \sigma_5 a^5 + \sigma_4 x^4 + \sigma_3 x^3 + \sigma_2 x^2 + \sigma_1 x^1 + \sigma_0$. Equation (23) can be rewritten in the form

$$\begin{aligned} c_4 y^4 + c_2 y^2 + c_1 y \ + c_0 &= c_7 y^7 + c_5 y^5 + c_3 y^3 \\ &= (c_7 y^4 + c_5 y^2 + c_3) y^3 \end{aligned}$$

or

$$A_7(y) = B_7(y).y^3 \tag{24}$$

Where $A_7(y) = c_4 y^4 + c_2 y^2 + c_1 y \ + c_0$ and $B_7(y) = c_7 y^4 + c_5 y^2 + c_3$.

In (24), one observes that both $A_7(y)$ and $B_7(y)$ are two affine polynomials that can be computed by the BRS algorithm. In other words, the values $A_7(\alpha^i)$ and $B_7(\alpha^i)$ for $1 \le i \le \alpha^{2^m - 1}$ can be obtained by a use of the BRS algorithm. If one exhaustively searches all nonzero field elements of $GF(2^m)$ for the roots of (23), there exists a set of field elements that satisfy (23). These elements are the solution of (23). Since the solutions of (23) are known, the roots of (21) are therefore found by the use of the transformation $x = y + a$. Finding the roots of polynomial of degree $n$=8, 9, 10, 11 is described in [3].

To show the performance of the new algorithm it has been implemented in C++ programming language. Fig.1 indicates the result of simulation. It is shown that this method can be up to 1.5 times faster than Chien search method.

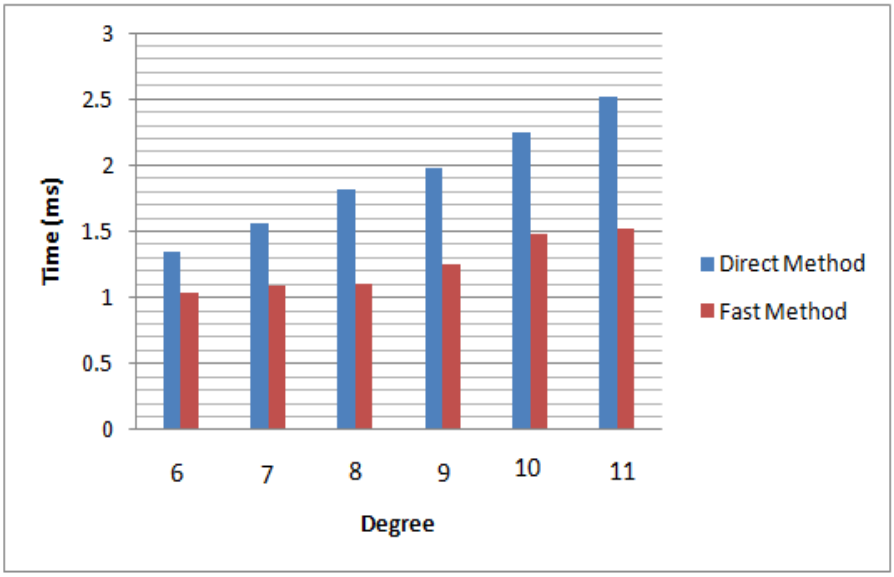


**Figure 1**. Execution time for polynomial root finding

## 7. Conclusions

Error correction plays a major role in communication and storage systems to increase the transmission reliability and achieve a better error correction performance with less signal power. One of the most common error correction codes in communication and storage systems is Bose-Chaudhuri-Hocqunghem (BCH) code. The conventional BCH decoder contains three major blocks, i.e., syndrome calculator, error locator calculator, and Chien search. It is well known that one of most time-consuming stages of decoding BCH codes is finding roots of the error-locator polynomial using Chien search method. In this paper the construction and decoding BCH codes is introduced and also an improved algorithm for finding roots of polynomials over finite fields is proposed with significantly better performance than well-known Chien search.

## References

[1] Lin, S., and Costello, D.J.: *"*Error control coding: fundamentals and applications*"* (Prentice-Hall Inc., 2004).

[2] R.T. Chien, B.D. Cunningham, and I.B. Oldham, "Hybrid methods for finding roots of a polynomial with application to BCH decoding," IEEE Transactions on Information Theory, vol. 15, no. 2, pp. 329-335, 1969.

[3] T.-K. Truong, J.-H. Jeng, and I.S. Reed, "Fast algorithm for computing the roots of error locator polynomials up to degree 11 in Reed-Solomon decoders," IEEE Transactions on Communications, vol. 49, no. 5, pp. 779-783, 2001.

[4] C. Paar, "Optimized arithmetic for Reed–Solomon encoders", *in Proc. IEEE Int. Symp. Inf. Theory, Ulm, Germany*, Jun.–Jul. 1997, pp. 250–250.

[5] O. Ore, "On a special class of polynomials," Trans. Am. Math. Soc., vol. 35, pp. 559–584, 1933.

[6] S.Lin and D.J. Constello, Error Control coding. Englewood Cliffs, NJ:Prentice-Hall, 1983.

[7] A.Hocquenghem, "Codes correcteurs d'erreurs," Chiffres, 2: 147- 56, 1959.

[8] R. E. Blahut, Algebraic Codes for Data Transmission. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[9] Y.-M. Lin et al., "A 26.9K 314.5 Mb/s Soft (32400, 32208) BCH Decoder Chip for DVB-S2 System," IEEE J. Solid- State Circuits, vol. 45, no. 11, pp. 2330-2340,Nov. 2010.
[10] C.-C. Chu, Y.-M. Lin, C.-H. Yang, H.-C. Chang, "A fully parallel BCH codec with double error correcting capability for NOR flash applications," in Proc. IEEE Int. Conf. Acoust. Speech, Signal Process., Mar. 2012, pp. 1605–1608.
[11] Dr. M. H. L. Chen, D. Fredmon, R. W. Donaldson, " Performance enhancment using Forward error correction on Power line Communication channels " m IEEE Trans. on Power Del. vol 9. no. 2 April 1994.
[12] L. Biard and D. Noguet (2008), "Reed Solomon Codes for Low Power Communication", Journal of Communications, vol. 3, no. 2, pp. 13-21.