ISSN 1512-1232

PERFORMANCE ANALYSIS OF BPSK SYSTEM WITH HARD DECISION (63, 36) BCH CODE

Mahmoud A. Smadi

Department of Electrical Engineering, The Hashemite University, Zarqa, 13115, Jordan, Email: smadi@hu.edu.jo

***Abstract**:*

The purpose of this paper is to study the Bose, Chaudhuri, and Hocquenghem (BCH) code, with an aim to simulate the encoding and decoding processes. The gain of the proposed code in investigated through applying it to binary phase sift keying (BPSK) modulation scheme in symmetric additive white Gaussian noise (AWGN) channel. The bit error probability (BEP) of coded (63, 36) BCH system was evaluated and compared with the performance of un-coded system.

1. INTRODUCTION

Channel coding for error detection and correction helps the communication system designers mitigate the effects of a noisy transmission channel. Error control coding theory has been the subject of intense study since the 1940s and now being widely used in communication systems. As illustrated by Shannon in his paper published in 1948 [1], for each physical channel there is a parametric quantity called the channel capacity C that is a function of the channel input output characteristics. Shannon showed that there exist error control codes such that arbitrary small error probability of the data to be transmitted over the channel can be achieved as long as the data transmission rate is less than C .

A generic block diagram of digital communication system involving coded waveforms is shown in Fig.1 [2]. The binary information sequence at the encoder input has a rate of R bits/sec. Mainly there are two types of channel encoding techniques. The first is the block coding, by which a blocks of k information bits are encoded into corresponding n bits blocks. Each n burst is called a code word with a total number of 2^k possible code words. The code rate, defined as the ratio $R_c = k/n$ is a measure of the amount of the redundancy introduced by the specific block coding technique.

The second type of encoding is the linear convolution encoding. A convolution encoder converts the entire information sequence stream, regardless of its length, into a single code word [3]. The encoder output sequence is a set of linear combinations of the input sequence that can be performed using a finite-state shift register approach. The code rate in this case R_c is defined as the reciprocal of the number of the shift register output bits for each data bit.

In both cases, the bit rate at the encoder output is R/R_c. Another designed parameter associated with the coding scheme to be used is the error correcting capability of this scheme. That is, how many errors that may be introduced by the channel can this code guarantee to correct. Hence, a good code is the one that insure a certain error correcting capability at a minimum R_c or maximum output encoder rate R/R_c.

ISSN 1512-1232

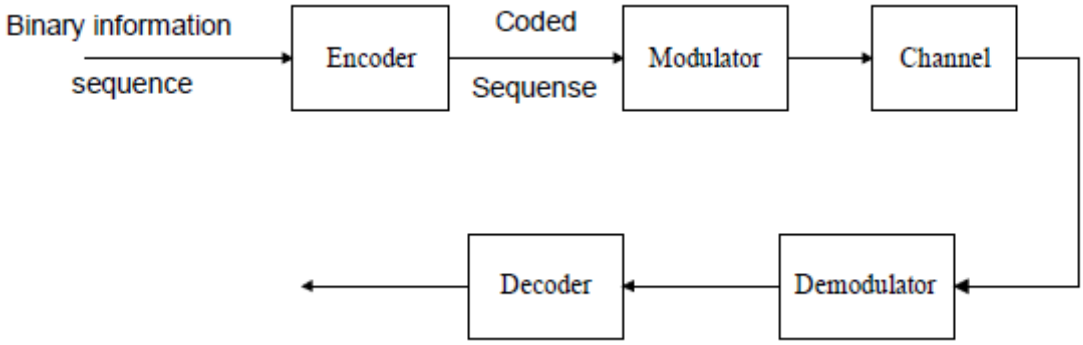

Fig.1: Generic digital communication system with channel coding [2]

The binary digits from the encoder are fed into a modulator, which maps them into one of the known digital modulation waveforms, say BPSK or BFSK. The channel over which the waveforms are transmitted will corrupt the waveforms in general by a multiplicative fading noise besides the traditional thermal AWGN. The resulting received noisy signal is demodulated to its binary regime and decoded back to the original binary information sequence.

The decoding decision scheme may be one of two possible decoding schemes hard or soft-decision scheme. In the hard decision decoding, the demodulator quantized the incoming signal into two levels, denoted as 0 and 1. The information sequence bits are then recovered by the decoder that will have a certain error correcting capability. On the other hand, if the unquantized (analog) demodulator output is fed to the decoder we call this decoding scheme soft-decision decoding.

In the hard decision case, the typical BPSK or BFSK result in a symmetric transmission error probability in which the probability that 1 is transmitted and 0 is detected $P(0/1)$ is equal to the probability that 0 is transmitted and 1 is detected $P(1/0)=p$. This channel is called a binary symmetric channel (BSC).

So many papers deal with the performance analysis of coded digital communication systems. The authors in [4] and [5] proposed several decoding techniques for the BCH codes and evaluate their performance. The performance of digital radio communication systems with a BCH coding scheme under a microwave oven interference environment is investigated in [6]. The authors found that performance improvement could be obtained by combining BCH codes with bit interleaving. The problem of efficient maximum-likelihood soft decision decoding of binary BCH codes is considered by the authors in [7]. On the other hand, the BER performance of severely punctured codes and the equivalent systematic codes is obtained assuming maximum likelihood decoding for (63,57) Hamming code in [8]. In contrast, the authors in [9] proposed an improved Hamming code method which is shown to be highly scalable without such overhead. Furthermore, the paper [10] analyzes the performance of concatenated coding systems operating over the BSC by examining the loss of capacity resulting from each of the processing steps. Finally, two schemes for differential encoding of block coded M-ary PSK signals are presented and compared in [11].

In this paper, the performance of BPSK coded system will be simulated. We will base our analysis on linear BCH block coding scheme with a hard decision decoding over AWGN binary symmetric channel. The rest of this paper is organized as follows. A brief description of linear block codes and algebraic field concepts will be given in the next section. The BCH codes will be treated in deep in section 3. Numerical Results are given in section 4. Finally, brief conclusions are provided in section 5.

ISSN 1512-1232

2. LINEAR BLOCK CODES

A block code C is constructed by breaking up the message data stream into blocks of length k has the form $\{m_0, m_1, ..., m_{k-1}\}$, and mapping these blocks into code words in C. The resulting code consists of a set of M code words $\{C_0, C_1, ..., C_{M-1}\}$. Each code word has a fixed length denoted by n and has a form $(c_0, c_1, ..., c_{n-1})$. The elements of the code word are selected from an alphabet field of q elements. In the binary code case, the field consists of two elements, 0 and 1. On the other hand, when the elements of the code word are selected from a field that has q alphabet elements, the code is nonbinary code. As a special case when q is a power of 2 (i.e. $q = 2^m$) where m is a positive integer, each element in the field can be represented as a set of distinct m bits.

As indicated above, codes are constructed from fields with a finite number of q elements called Galois field and denoted by GF (q). In general, finite field GF (q) can be constructed if q is a prime or a power of prime number. When q is a prime, the GF(q) consist of the elements $\{0,1,2,....q-1\}$ with addition and multiplication operations are defined as a modulo-q. If q is a power of prime (i.e. $q = p^m$ where m is any positive integer), it is possible to extend the field GF(p) to the field GF$(q = p^m)$. This is called the extension field of GF(p) and in this case multiplication and addition operations are based on modulo-p arithmetic.

To construct the elements of the extension GF$(q = 2^m)$ from the binary GF(2) with elements 0 and 1, a new symbol α is defined with multiplication operation properties as: $0.\alpha^i = \alpha^i.0 = 0, 1.\alpha^i = \alpha^i.1 = \alpha^i$ and $\alpha^i.\alpha^j = \alpha^j.\alpha^i = \alpha^{i+j}$. The elements of the GF$(q = 2^m)$ that satisfy the above properties are $\{0, 1, \alpha, \alpha^2, ..., \alpha^j, ..\}$. As the field should has 2^m elements and be closed under multiplication α should satisfies the condition $\alpha^{q-1} = 1$. Hence; the elements of the extension GF$(q = 2^m)$ are $\{0, 1, \alpha, \alpha^2, ..., \alpha^{q-2}\}$ which is a commutative group under an addition and multiplication (excluding the zero element) operations. α is called a primitive element since it can generate all other field elements and it is a root of a primitive polynomial $p(x)$. As mentioned before, each element in the field can be represented as a set of m tuple bits. To make the picture clear, Table I shows the three representation for the elements of GF(2^4) with a primitive polynomial $p(x) = 1 + x + x^4$ [4].

Besides the code rate parameter R_c defined early, an important parameter of the code word is its minimum distance denoted by $d_{\min}$. As the code weight defined as the number of nonzero elements in the code, the minimum distance of a block code is the minimum distance between all distinct pairs of code words which is the same as the minimum weight of the code. The minimum distance is a measure of the separation between code words and thus a code with minimum distance $d_{\min}$ can detect any error pattern of weight less than or equal to $d_{\min} - 1$ [3].

ISSN 1512-1232

Table I: Representations of the elements of GF(2^4) with $p(x)=1+x+x^4$ [4].

Power representation	Polynomial representation	4-Tuple representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
α	α	(0 1 0 0)
α^2	α^2	(0 0 1 0)
α^3	α^3	(0 0 0 1)
α^4	$1+\alpha$	(1 1 0 0)
α^5	$\alpha+\alpha^2$	(0 1 1 0)
α^6	$\alpha^2+\alpha^3$	(0 0 1 1)
α^7	$1+\alpha+\alpha^3$	(1 1 0 1)
α^8	$1+\alpha^2$	(1 0 1 0)
α^9	$\alpha+\alpha^3$	(0 1 0 1)
α^{10}	$1+\alpha+\alpha^2$	(1 1 1 0)
α^{11}	$\alpha+\alpha^2+\alpha^3$	(0 1 1 1)
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	(1 1 1 1)
α^{13}	$1+\alpha^2+\alpha^3$	(1 0 1 1)
α^{14}	$1+\alpha^3$	(1 0 0 1)

The linearity property of a code is fairly a simple concept. Suppose that C_i and C_j are two code words in an (n,k) block code and let α_1 and α_2 are any two of the field elements over where the code is defined, then the code is called a linear code if and only if $\alpha_1 C_i + \alpha_2 C_j$ is also a code word in C.

2.1 The generator matrix and parity check matrix

let $\mathbf{X}_m = (x_{m1}, x_{m2}, ..., x_{mk})$ be the k information bite at the encoder input and $\mathbf{C}_m = (c_{m1}, c_{m2}, ..., c_{mn})$ is the encoder output vector. The encoding operation performed in linear binary block encoder can be represented in matrix form as

$$\mathbf{C}_m = \mathbf{X}_m \mathbf{G} \tag{1}$$

where $\mathbf{G}$ is called the generator matrix of the code, is

ISSN 1512-1232

$$\mathbf{G}=\begin{bmatrix}\mathbf{g}_1\\ \mathbf{g}_2\\ .\\ \mathbf{g}_k\end{bmatrix}=\begin{bmatrix}\mathbf{g}_{11} & \mathbf{g}_{12} & . & \mathbf{g}_{1n}\\ \mathbf{g}_{21} & \mathbf{g}_{22} & . & \mathbf{g}_{2n}\\ . & . & . & .\\ \mathbf{g}_{k1} & \mathbf{g}_{k2} & . & \mathbf{g}_{kn}\end{bmatrix} \tag{2}$$

and hence; any code word is a linear combination of the rows $\{\mathbf{g}_i\}$ of $\mathbf{G}$, i.e.,

$$\mathbf{C}_m = x_{m1}\mathbf{g}_1 + x_{m2}\mathbf{g}_2 + ... + x_{mk}\mathbf{g}_k \tag{3}$$

Since the linear (n,k) code with 2^k distinct code words is a subset of dimension k, the rows of $\mathbf{G}$ must be a set of linearly independent rows, and hence, $\mathbf{G}$ is not unique. Any generator matrix of (n,k) linear block code can be reduced by row operation which will keep the linearly independence property of $\mathbf{G}$ to a symmetric form given as

$$\mathbf{G}=\left[\ \mathbf{P}\ \middle|\ \mathbf{I}_k\ \right]=\left[\begin{array}{cccc|cccc} p_{11} & p_{12} & . & p_{1,n-k} & 1 & 0 & . & 0\\ p_{21} & p_{22} & . & p_{2,n-k} & 0 & 1 & . & 0\\ . & . & . & . & . & . & . & .\\ p_{k1} & p_{k2} & . & p_{k,n-k} & 0 & 0 & . & 1 \end{array}\right] \tag{4}$$

where $\mathbf{I}$ is a $k\times k$ identity matrix and $\mathbf{P}$ is a $k\times(n-k)$ matrix that determines the code redundant bits. In this case the last k bits of each code word are identical to the k information bits. Associated with any linear (n,k) block code there is a linear $(n,n-k)$ dual code with 2^{n-k} code words which is the null space of the (n,k) code. The generator matrix associated with the dual code, consists of $(n-k)$ linearly independent rows and denoted by $\mathbf{H}$. Since $\mathbf{G}$ and $\mathbf{H}$ are in the null space of each other, any code word generated by $\mathbf{G}$ is orthogonal to every row in $\mathbf{H}$. That is

$$\mathbf{C}_m\mathbf{H}^T=\mathbf{0} \quad \text{OR} \quad \mathbf{G}\mathbf{H}^T=\mathbf{0} \tag{5}$$

Now if the block code is in symmetric form, it follows from the last equation that

$$\mathbf{H}=\left[\ \mathbf{I}_{n-k}\ \middle|\ \mathbf{P}\ \right] \tag{6}$$

and since for linear block code the minimum distance is equal to the minimum weight of the code, another conclusion one can draw from (5) is that the minimum distance for a linear block code is the minimum number of columns in $\mathbf{H}$ that may add up to the zero vector.

2.2 Cyclic codes

BCH code is a subset of a general linear block codes called cyclic codes. Cyclic codes are a class of linear codes which satisfy the following cyclic shift property: if C is a code word of a cyclic code, then any cyclically shifts of C is also a code word. In discussing cyclic code and later a BCH code, its more convenient to deal with polynomials representation rather than matrices representation of the code. So, to develop the algebraic properties of a cyclic code, we represent the components of a code word $C=(c_0,c_1,...,c_{n-1})$ as the coefficients of a polynomial called a code polynomial as follows

$$c(x)=c_0+c_1x+c_2x^2+...+c_{n-1}x^{n-1} \tag{7}$$

it can be shown that the code polynomial resulting from cyclically shifting the code word C i-th times denoted by $c^{(i)}(x)$ is the remainder resulting from dividing the polynomial $x^i c(x)$ by x^n+1 [4].

ISSN 1512-1232

We can generate a binary (n,k) cyclic code by using a generator polynomial $g(x)$ of degree $n-k$ has the form

$$g(x) = g_0 + g_1 x + ... + g_{n-k} x^{n-k} \tag{8}$$

where g_i's is either 0 or 1 in the binary code case. A number of important properties of the generator polynomial can be summarized [4]:

1. The coefficients g_0 and g_{n-k} have to be 1.
2. Any code word polynomial $c(x)$ is multiple of $g(x)$ (i.e. $c(x) = m(x)g(x)$) where $m(x) = m_0 + m_1 x + m_2 x^2 + ... + m_{k-1} x^{k-1}$ is the message polynomial.
3. $g(x)$ is a factor of $x^n + 1$.

The last property says that any factor of $x^n + 1$ with degree $n-k$, generates an (n,k) cyclic code. For large n, $x^n + 1$ may have many factors of degree $n-k$. Some of theses polynomials generate good codes and others generate bad codes [4]. To be consistent with the matrix representation of a general block code as discussed in the pervious subsection, the generator matrix for (n,k) cyclic code can be derived from the generator polynomial given in (8) as

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & . & . & g_{n-k} & 0 & 0 & 0 & 0 \\ 0 & g_0 & g_1 & g_2 & . & . & g_{n-k} & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & . & . & g_{n-k} & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & . & g_{n-k} \end{bmatrix} \tag{9}$$

with $g_0 = g_{n-k} = 1$. A symmetric cyclic code can be obtained similarly by row operations.

4. BCH CODES

BCH codes are a large class of cyclic codes that include both binary and nonbinary codes. Binary (n,k) with any positive integer $m \geq 3$ BCH codes can be constructed with the following parameters

$$\begin{aligned} n &= 2^m - 1 \\ n - k &\leq mt \\ d_{\min} &\geq 2t + 1 = \delta \end{aligned} \tag{10}$$

where t is the error correcting capability and δ is called the code design distance. That is a BCH code with specified parameters given in (10), guarantees to correct t or less number of errors in the received n block bits. The generator polynomial $g(x)$ of the t-error correcting BCH code is the lowest degree polynomial over GF (2), which has the consecutive $\alpha, \alpha^2, ..., \alpha^{2t}$ as its roots (i.e. $g(\alpha^i) = 0$, $i = 1, 2, .., 2t$). Let $\phi_i(x)$ be the minimal polynomial (the minimum degree polynomial that has α^i and its conjugates as a roots) corresponding to α^i, the generator polynomial must be the least common multiple (LCM) of $\phi_i(x)$, $i = 1, 2, ..., 2t$. That is [4]

$$g(x) = \text{LCM}\{\phi_1(x), \phi_2(x), \phi_3(x), ..., \phi_{2t}(x)\} \tag{11}$$

All the field elements of the form $\alpha^j = (\alpha^i)^{2^l}$, $l \geq 1$ and i is odd, are called conjugate of α^i and all of them over the defined field have the same minimal polynomial (i.e. $\phi_j(x) = \phi_i(x)$). Hence,

ISSN 1512-1232

every even power of α in (11) has the same minimal polynomial as the preceding odd power of α. This reduces the number of terms in (11) to t terms, so

$$g(x) = \mathrm{LCM}\{\phi_1(x), \phi_3(x), ..., \phi_{2t-1}(x)\} \tag{12}$$

the BCH codes defined above are called primitive, narrow-sense BCH codes.

Since any code word polynomial $c(x) = c_0 + c_0 x + ... + c_{n-1}x^{n-1}$ is a multiple of the generator polynomial $g(x)$ (i.e. $c(x) = m(x)g(x)$), $c(x)$ has $\alpha, \alpha^2, ..., \alpha^{2t}$ as a roots, then

$$c(\alpha^i) = c_0 + c_1\alpha^i + ... + c_{n-1}\alpha^{i(n-1)} = 0 \ , \ i = 1, 2, .., 2t \tag{13}$$

or in matrix form

$$(c_0, c_1, ..., c_{n-1}) \begin{bmatrix} 1 \\ \alpha^i \\ \alpha^{2i} \\ . \\ \alpha^{i(n-1)} \end{bmatrix} = \mathbf{0} \tag{14}$$

by combining (14) with (5), the parity check matrix of the BCH codes in α's form can be written as

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & . & \alpha^{n-1} \\ 1 & (\alpha^2) & (\alpha^2)^2 & . & (\alpha^2)^{n-1} \\ 1 & (\alpha^3) & (\alpha^3)^2 & . & (\alpha^3)^{n-1} \\ . & . & . & . & . \\ 1 & (\alpha^{2t}) & (\alpha^{2t})^2 & & (\alpha^{2t})^{n-1} \end{bmatrix} \tag{15}$$

which can be written in 0,1 form by represent α^i's in its m tuples form.

Example

Consider a primitive, narrow-sense BCH code with $n = 2^4 - 1 = 15$ and $t = 2$. It follows from (12) that this code is generated by

$$g(x) = \mathrm{LCM}\{\phi_1(x), \phi_3(x)\} \tag{16}$$

where

$$\begin{aligned} \phi_1(x) &= (x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8) = 1 + x + x^4 \\ \phi_3(x) &= (x+\alpha^3)(x+\alpha^6)(x+\alpha^9)(x+\alpha^{12}) = 1 + x + x^2 + x^3 + x^4 \end{aligned} \tag{17}$$

since there is no common factor between $\phi_1(x)$ and $\phi_3(x)$

$$g(x) = \phi_1(x)\phi_3(x) = 1 + x^4 + x^6 + x^7 + x^8 \tag{18}$$

this the resulting code is a primitive $(15, 7)$ BCH code with $d_{\min} \geq 2t + 1 = 5$. The parity check matrix for this code is

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & . & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & . & \alpha^{42} \end{bmatrix} \tag{19}$$

The general definition of binary BCH codes is as follow. Let β be an element of GF(2^m) and b any nonnegative integer. Then a binary BCH code with design distance δ has a generator polynomial $g(x)$ with the following consecutive powers of β as roots

$$\beta^b, \beta^{b+1}, ..., \beta^{b+\delta-2}$$

let $\psi_i(x)$ and n_i be the minimal polynomial and the order of β^{b+i}, respectively. Then

ISSN 1512-1232

$$g(x) = \mathrm{LCM}\{\psi_1(x), \psi_2(x), ..., \psi_{b+2}(x)\} \tag{20}$$

with a code length

$$n = \mathrm{LCM}\{n_1, n_2, ..., n_{b+2}\} \tag{21}$$

The BCH code defined above is called a nonprimitive, wide-sense binary BCH code with a design distance δ. As a special cases when $b=1$ the code becomes narrow-sense and if β is a primitive element, the code is primitive code.

3.1 Decoding of binary BCH codes

Decoding process of the BCH codes is the most challenging task. Mainly, we have two decoding algorithms for BCH codes, namely: Peterson-Gorentien-Zierler algorithm and Berlekamp-Massey algorithm. Assume that the received code word $(r_0, r_1, ..., r_{n-1})$ is differs from the sent code word $(c_0, c_1, ..., c_{n-1})$ in $x^{i_1}, x^{i_2}, ..., x^{i_\nu}$ positions, then the error code word will have a nonzero elements at these positions and the error polynomial can be written as

$$e(x) = x^{i_1} + x^{i_2} + ... + x^{i_\nu} \tag{22}$$

Both of these algorithms need the computation of the syndromes of the received code polynomial $r(x)$. Define the syndrome S_j to be

$$S_j = r(\alpha^j) = e(\alpha^j), \quad j = 1, 2, ..., 2t \tag{23}$$

or

$$\begin{aligned} S_1 &= X_1 + X_2 + ... + X_\nu \\ S_2 &= X_1^{\,2} + X_2^{\,2} + ... + X_\nu^{\,2} \\ &\cdots \\ S_{2t} &= X_1^{\,2t} + X_2^{\,2t} + ... + X_\nu^{\,2t} \end{aligned} \tag{24}$$

where $X_l = \alpha^{i_l}$ is the error locations. Defining what is called error locator polynomial as

$$\Lambda(x) = \Lambda_\nu x^\nu + \Lambda_{\nu-1} x^{\nu-1} + ... + \Lambda_1 x + 1 = (1 - xX_1)(1 - xX_2)...(1 - xX_\nu) \tag{25}$$

that has zeros at $x = X_l^{-1}$. It can be shown that (24) and (25) can be coupled together in matrix form and written as

$$\underbrace{\begin{bmatrix} S_1 & S_2 & . & S_t \\ S_2 & S_3 & . & S_{t+1} \\ . & . & . & . \\ S_t & S_{t+1} & . & S_{2t-1} \end{bmatrix}}_{A} \begin{bmatrix} \Lambda_t \\ \Lambda_{t-1} \\ . \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} S_{t+1} \\ S_{t+2} \\ . \\ S_{2t} \end{bmatrix} \tag{26}$$

Peterson's algorithm is based on solving (26) for Λ_i's. If A is found to be singular that means we have less than t errors in the received code word. In this case we have to reconstruct a new syndrome matrix by deleting the two right most columns and the two bottom rows from A and solve a gain for Λ_i's excluding Λ_t and so on. After Λ_i's are found the error correct polynomial defined in (25) is constructed. Finally, the roots of $\Lambda(x)$ are to be found using Chien's search algorithm and the error locations set to be the reciprocal of these roots.

ISSN 1512-1232

3.1.1 Berlekamp's decoding algorithm

Berlekamp's algorithm is much more difficult to understand than Peterson's approach, but results in more efficient implementation. Berlekamp's algorithm for binary BCH codes decoding is a recursive algorithm that is summarized in the following steps [3]

1. Define the syndrome polynomial $S(x) = S_1 x + S_2 x^2 + ... + S_{2t+1} x^{2t+1} + ...$
2. Set the initial conditions: $k = 0, \Lambda^{(0)}(x) = 1, \text{and} T^{(0)}(x) = 1.$
3. Let $\Delta^{(2k)}$ be the coefficient of x^{2k+1} in the product $\Lambda^{(2k)}(x)\left[1 + S(x)\right]$.
4. Compute $\Lambda^{(2k+2)}(x) = \Lambda^{(2k)}(x) + \Delta^{(2k)}\left[x.T^{(2k)}(x)\right]$.
5. Compute $T^{(2k+2)}(x) = \begin{cases} x^2 T^{(2k)}(x) & \text{if } \Delta^{(2k)} = 0 \text{ or } \deg\left[\Lambda^{(2k)}(x)\right] > k \\ \dfrac{x\Lambda^{(2k)}(x)}{\Delta^{(2k)}} & \text{if } \Delta^{(2k)} \neq 0 \text{ and } \deg\left[\Lambda^{(2k)}(x)\right] \leq k \end{cases}$
6. Set $k = k + 1.$ if $k < t$ then go to step 3.
7. Determine the roots of $\Lambda(x) = \Lambda^{(2k)}(x)$. If the roots are distinct, then correct the corresponding locations in the received code word and STOP.
8. Declare a decoding failure and STOP.

5. RESULTS

The aim of this project is to simulate the performance of BPSK in an AWGN environment with hard decision detection using $(63,36)$ binary BCH code with error-correcting capability $t = 5$. The MATLAB code that carried out this simulation is given in the appendix.

The key thing here; as I believe; is to build up the alpha table for the code which contains the m-tuple representation of the elements in GF ($2^6 = 64$) $\left\{0, 1, \alpha, \alpha^2, ..., \alpha^{62}\right\}$. The power of α for each entry in the table is evaluated using *index.m* function. Note that the indices are $\left\{-1, 0, 1, 2, ..., 62\right\}$ with -1 refers to the 0 element in the field. This is done so that adding two elements in the field is performed by adding the two m-tuple of the elements using the alpha table. The index for the resulting tuple will be the resulting element. While the multiplication operation is performed by adding the two indices of the two elements.

The average BEP for the simulated system is shown in Fig.2. For comparison purpose, the BEP for uncoded BPSK is also shown using the analytical formula $P_b = \frac{1}{2} erfc\left(\sqrt{\gamma_b}\right)$, where $\gamma_b = \frac{E_b}{N_0}$ is the average SNR/bit.

ISSN 1512-1232

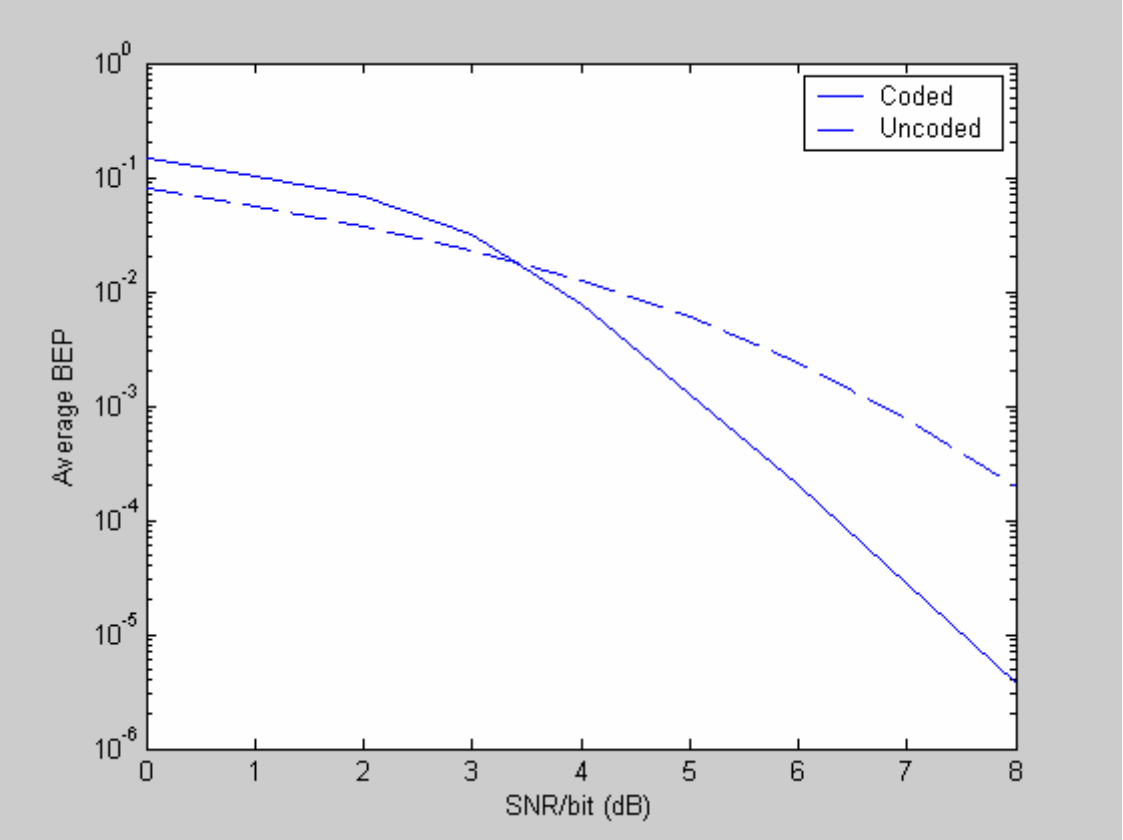

Fig.2: Average BEP for BPSK systems.

As we see, for low SNR, the uncoded system performance is better than the coded one. That is because at low SNR let us say 2 dB, the BEP for the encoded case is about 0.04. For coded system the SNR will be reduced by $k/n = 0.57$ so that it will be about –0.43 dB and hence the BEP is around 0.1, which means that out of 63 code word bits 6 bits will be in error after the hard decision detection. So, the decoding algorithm will fail in detecting these errors, which leads to worse performance.

6. CONCLUSIONS

The BEP for coded BPSK system in symmetric AWGN channel based on a hard decision decoding was simulated. The coding scheme that used is binary BCH code with error-correcting capability. The system's performance improvement using channel coding at reasonable SNR is considerable in most cases. Since this improvement is due to the redundancy that is inserted by the coding technique, the price to be paid for this improvement is the higher transmission data rate and hence higher transmission bandwidth is required. Generally, analytical evaluation of the coded system performance is very tough, so simulation should be carried to do that.

ISSN 1512-1232

APPENDIX: MATLAB Codes
A. Main code

```
clear all,clc,format long
n=63;m=6;k=36;t=5;
%G: Generator matrix in symmetric form
%alpha: m-tuples representation of the field elements
%mes:  Message data stream
%c: Code word corresponding to mes
%cpbsk: Baseband BPSK version of c
%SNR: Signal-to-noise ratio for the uncoded system
%uvar: Gaussian noise variance for the uncoded system
%cvar: Gaussian noise variance for the coded system
%cn: Noisy version of cpbsk
%r: Transmitted code word
%e: Error code vector
%d: Demodulated received code word
%dmes: Demodulated message data

%Generator matrix
G=zeros(k,n);
for i=1:k
  G(i,i:n-k+i)=bchpoly(n,k);
end
B=G(:,n-k+1:n);
G=mod(mod(inv(B),2)*G,2);
%Alpha-Index look-up tables
alpha=zeros(n+1,m);
alpha(2:m+1,:)=eye(m);
for i=m+2:n+1
  alpha(i,:)=mod(alpha(i-m,:)+alpha(i-m+1,:),2);
end
%Data encoding,decoding and Pe calculations
ind=1;L=3e4;
for SNR=0:8
  Pe(ind)=0;
  for l=1:L
    r=zeros(1,n);d=zeros(1,n);cbpsk=zeros(1,n);e=zeros(1,n);
    mes=round(rand(1,k));
    c=mod(mes*G,2);
    for j=1:n
      if c(j)==0
        cbpsk(j)=-1;
      else
        cbpsk(j)=1;
      end
    end
    uvar=10^(-SNR/10)/2;
    cvar=uvar*n/k;
    Gn=sqrt(cvar)*randn(1,n);
    cn=cbpsk+Gn;
    for j=1:n
      if cn(j)<0
        r(j)=0;
```

ISSN 1512-1232

```
  else
    r(j)=1;
  end
end
%Syndromes calculation
S=zeros(2*t,m);s=zeros(1,2*t);
for i=1:2*t
  for j=1:n
    if r(j)==0
      S(i,:)=S(i,:);
    else
      S(i,:)=mod(S(i,:)+alpha(rem(i*j-i,n)+2,:),2);
    end
  end
  s(i)=index(S(i,:))-1;
end
%Data decoding using Berlekamp's algorithm
lmd=zeros(1,t);T=zeros(1,t);
lmd(1)=s(1);
if s(1)==-1
   T(1)=-1;T(2)=0;
else
   T(1)=mod(n-s(1),n);
end
for v=1:t-1
   deltar=alpha(rem(s(2*v+1),n)+2,:);
   for j=1:v
      if lmd(j)==-1 | s(2*v+1-j)==-1
         deltar=deltar;
      else
         deltar=mod(deltar+alpha(rem(lmd(j)+s(2*v+1-j),n)+2,:),2);
      end
   end
   delta=index(deltar)-1;
   V=lmd;
   if delta==-1 |T(v)==-1
      lmd(v+1)=-1;
   else
      lmd(v+1)=rem(delta+T(v),n);
   end
   for i=2:v
      if delta==-1 |T(i-1)==-1
         lmdr=alpha(rem(lmd(i),n)+2,:);
       else
         lmdr=mod(alpha(rem(lmd(i),n)+2,:)+alpha(rem(T(i-1)+delta,n)+2,:),2);
       end
       lmd(i)=index(lmdr)-1;
    end
    if delta~=-1
       T(1)=mod(n-delta,n);
       for i=2:v+1
          T(i)=mod(V(i-1)-delta,n);
       end
    else
```

ISSN 1512-1232

```
            T(v+2)=T(v);
            for j=1:v+1
               T(j)=0;
            end
         end
      end
      %Error locations using Chien's algorithm
      for i=0:n-1
         xx=zeros(1,m);
         for j=1:t
            if lmd(j)==-1
               xx=xx;
            else
               xx=mod(xx+alpha(rem(lmd(j)+j*i,n)+2,:),2);
            end
         end
         if index(xx)==1
            e(mod(n-i,n)+1)=1;
         end
      end
      d=mod(r+e,2);
      dmes=d(:,n-k+1:n);
      for i=1:k
         if dmes(i)~=mes(i)
            Pe(ind)=Pe(ind)+1;
         end
      end
   end
   Pe(ind)=Pe(ind)/(k*L);
   ind=ind+1;
end
SNR=0:8;
semilogy(SNR,0.5*erfc(sqrt(10.^(SNR/10))),'--');
hold
semilogy(SNR,Pe)
```

B. index.m

```
function y=index(x)
n=63;m=6;
alpha=zeros(n+1,m);
alpha(2:m+1,:)=eye(m);
for i=m+2:n+1
   alpha(i,:)=mod(alpha(i-m,:)+alpha(i-m+1,:),2);
end
for i=1:n+1
   if x==alpha(i,:)
      y=i-1;
   end
end
```

ISSN 1512-1232

REFERENCES

[1] C. E. Shannon: "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27 , October 1948, pp. 379–423.
[2] J. Proakis, *Digital communications* , NY: McGraw Hill, 2001.
[3] S. Lin and D.J. Costello, Jr. *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1983.
[4] Joiner, L.L. and Komo, J.J., **"Decoding binary BCH codes,"** *IEEE Southeastcon Proceedings*, Mar 1995, pp. 67 – 73.
[5] Salah A. Aly, Andreas Klappenecker and Pradeep Kiran Sarvepalli," **On Quantum and Classical BCH Codes,"** *IEEE Transactions on Information Theory*, vol. 53, March 2007, pp.1183 –1188.
[6] Kanemoto, H.; Miyamoto, S.; Morinaga, N.**, "Performance of digital radio communication system with BCH coding under microwave oven interference** ," IEE Electronics Letters, vol. 34, Jul 1998, pp.1465 –1466.
[7] Vardy, A. and Beapos;ery, Y.**, "Maximum-likelihood soft decision decoding of BCH codes,"** IEEE Transactions on Information Theory, vol. 40, Mar 1994, pp. 546 –554.
[8] Michelson, A.M. and Freeman, D.F.**, "Bit-error rate performance of the (63,57) Hamming code and a severely punctured convolution code with maximum likelihood decoding** ," IEEE Military Communications Conference, 2-5 Oct 1994.
[9] Kumar, U.K. and Umashankar, B.S. **"Improved Hamming Code for Error Detection and Correction** ," IEEE International Symposium on Wireless Pervasive Computing, 5-7 Feb 2007.
[10] MacMullan, S.J. and Collins, O.M., "The capacity of binary channels that use linear codes and decoders," IEEE Transactions on Information Theory, vol. 44, pp. 197 – 214, Jan 1998.
[11] Sayegh, S. and Hemmati, F. "Differentially encoded M-PSK block codes," IEEE Transactions on Communications, vol. 43, pp. 2426 – 2428, Sep 1995.

Article received: 2009-11-24