International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 3– No.5 , July 2012 – www.ijais.org

Comparative Analysis of Error Correcting Codes for Noisy Channel

Sukhjeet Kaur
Lovely Professional University
G.T. Road, Phagwara, Punjab (INDIA) -144411

ABSTRACT
In the communication system there are errors during the transmission of the information. There are error correcting codes to detect as well as correct errors. These error correcting codes have vast field of application. Communication performance is improved by enabling the transmitted signal to better withstand the effects of channel impairments such as noise, interference and fading which occur during transmission. In this paper Hamming and Reed Solomon codes are discussed under same noisy conditions and there comparison is done with data transmitted without using codes and Bit Error Rate performance is shown. This comparative analysis is represented in Graphical user interface (GUI).

Keywords
BSC, AWGN, Hamming code, Reed Solomon code, BER, SNR.

1. INTRODUCTION
A communication system is used to send and receive the information from a source to a user. The channel coding provides a reliable communication system by introducing redundancy bits to the actual information. The channel encoder adds redundant bits to the source encoded information bits to generate code words. To transmit the channel encoded digital information over a band pass channel digital information modulation is performed. A communication channel is used to transmit modulated information from a transmitter to a receiver. A channel can be modeled physically by calculating the effects which modify the transmitted signal. This paper uses binary symmetric channel and AWGN channel. In a receiver section the inverse process takes place. The received information is demodulated to produce the channel encoded information that is effected by channel impairments. A channel decoder removes the redundancy using the same technique used in channel encoder. Source decoder expands the channel decoder output that produces the original transmitted information. The goal of a communication system is to transmit the information without any loss. The channel impairments create errors in the message being transmitted, which is measured in terms of BER. Channel coding is a method in which the reliability of the channel increases by reducing the information rate. This can be accomplished by adding redundancy to the information being transmitted. This process leads to a longer coded symbols vector than the actual information. The receiver can be able to detect and correct the corrupted bits in the channel using these redundant bits [1]. The two classes of error correcting codes linear block codes and convolutional codes. Block codes process the information on a block by block basis, treating each block of information bits independently from others. Block coding is a memoryless operation that codewords are independent from each other. In convolutional encoder the output depends not only on the current input information but also on the previous inputs or outputs, either on a block by block or a bit by bit basis [2]. The detection and correction capability depends on the minimum distance of the code words.

2. HAMMING CODE
Hamming code is a linear error correcting code named after its inventor, Richard Hamming. When the Hamming distance between the transmitted and received bit patterns is less than or equal to one Hamming codes can detect up to two contiguous bit errors and correct single bit errors. Thus reliable communication is possible. For each integer m>=2, there is a code with m parity bits and (2^m)-m-1 data bits. The number of bit position in which two codewords differ is called Hamming distance. Its significance is that if two codewords are a Hamming distance d apart, it will require d single-bit errors to convert one into the other. The error detecting and error correcting properties of a code depend on its Hamming distance. Hamming code is based on the principle of adding 'm' redundancy bits to 'k' data bits such that $2^m \geq k + m + 1$. These redundancy bits are to be interspersed at bit positions 2^k where k = 0, 1, 2, 3…. with the original data bits. Suppose we take seven bit data to be transmitted then the actual data which is transmitted is eleven bit. PPXPXXXPXXX, where the P refers to the Hamming bits that are to be calculated and interspersed at bit positions 1, 2, 4, & 8 and X refers to the data bits. The Hamming bit at bit position 1 is selected such that there is even parity at bit positions 1, 3, 5, 7, 9, & 11. The Hamming bit at bit position 2 is selected such that there is even parity at bit positions 2, 3, 6, 7, 10, & 11.The Hamming bit at bit position 4 is selected such that there is even parity at bit positions 4, 5, 6, & 7. The Hamming bit at bit position 8 is selected such that there is even parity at bit positions 8, 9, 10, & 11. Now the result is a codeword. Now for the detection of error in the codeword and we have to check the parity bits for parity at first position XOR of 1,2,4,8 is done for parity 2 XOR of 2,3,6,7,10 and 11 is done ,for parity at position 4 XOR of 4,5,6 and 7 is done for parity at position 8 XOR of 8 ,9 ,10and 11 is done. Then result is written as p1 p2 p4 p8, then decimal form of the respective bits is taken and the resulted bit position is in error and is changed if 0 then to1 and if 1 changed to 0.Then data bits are taken and parity bits are removed at the receiver side. The redundancy bits can be appended at the end of data bits [3].Block codes are referred to as (n, k) codes. A block of k information bits are coded to become a block of bits. The binary (n, k) Hamming codes have the following parameters:

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

Number of information bits $k = 2^m - m - 1$

Number of parity bits $n - k = m$

Error correcting capability $t = 1$

The block codes are specified by ($2^m-1, 2^m-m-1$). Most block codes are systematic in that the information bits remain unchanged with parity bits attached either to the front or to the back of the information sequence. Other method of encoding and decoding with G and H matrix. G is the generator matrix and H is the parity check matrix. If the generator matrix G of a linear blocks (n, k, $d_{\min}$) code c can be brought to a systematic form, G_{sys} by elementary row operations and/or column permutations. G_{sys} is composed of two sub-matrices. The k-by-k identity matrix, denoted by I_k and a k-by-(n-k) parity sub-matrix P, such that

$$G_{sys} = (I_k \mid P) \tag{1}$$

where

$$P = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \tag{2}$$

Since GH^T =0, is in the systematic form, H_{sys} of the parity-check matrix is $H_{sys} = (P^T \mid I_{n-k})$ now to achieve the codeword C information is multiplied with the generator matrix. Now all we have to do is multiply the information vector d with matrix G to get a codeword c.

$$c = dG \tag{3}$$

The main concept in decoding is to determine which of the 2^k valid codewords was transmitted given that one of the 2^n has been received. Decoding of block codes is pretty easy. We compute something called a syndrome of an incoming codeword.

$$s = H^T c \tag{4}$$

We do that by multiplying the incoming vector by the transposed parity matrix H. Data is multiplied with matrix G on the encoder side and by H on the decoder side. The size of the syndrome will be equal to (n-k) bit. All zero syndrome means no error has occurred. If there is some value of syndrome then syndrome must be zero. Multiply the received code vector with the H^T matrix, to see if we get all zeros since we know that this is a valid codeword. Now compute the ordinal number from this sequence. This tells us that which bit is in error. A syndrome table can be pre-created in the decoder so that a quick mapping to the bit location where the error has occurred can be made. This is done by changing a valid code vector one bit at a time and looking at the computed syndrome. An advantage of Hamming codes is that encoding and decoding are easy to implement. They would be effective as a simple and efficient code over a channel where it is known that errors are burst free and tend to be widely dispersed. Disadvantages of Hamming codes are that they are very ineffective for low SNR, where the received signal level is very low. These types of conditions will tend to cause more frequent errors. Hamming codes do very poorly against the bursts of errors [4].

3. REED SOLOMON CODES

In 1960, Irving Reed and Gustave Solomon described a construction of error correcting codes, which are called Reed Solomon codes, based on polynomials over finite fields. Almost 50 years after their invention, Reed Solomon codes remain ubiquitous today in diverse applications ranging from magnetic recording to UPS bar codes to satellite communications Reed-Solomon codes have been largely employed as channel codes due to their excellent error detection and correction properties. A Reed Solomon code is a block code and the message to be transmitted is divided up into separate blocks of data. Each block then has parity protection information added to it to form a self-contained code word. It is also used as a systematic code which means that the encoding process does not alter the message symbols and the protection symbols are added as a separate part of the block. A Reed Solomon code is linear code and it is cyclic. It belongs to the family of Bose-Chaudhuri-Hocquenghem (BCH) codes but is distinguished by having multi-bit symbols. This makes the code particularly good at dealing with bursts of errors because, although a symbol may have all its bits in error, this counts as only one symbol error in terms of the correction capacity of the code. Choosing different parameters for a code provides different levels of protection and affects the complexity of implementation. Thus a Reed-Solomon code can be described as an (n, k) code where n is the block length in symbols and k is the number of information symbols in the message.

$$n \le 2^m - 1 \tag{5}$$

where m is the number of bits in a symbol.

RS codes can be designed to have any redundancy. However, the complexity of a high speed implementation increases with redundancy. Thus, the most attractive RS codes have high code rates. The Reed Solomon codes are particularly useful for burst error correction. They are effective for channels that have memory. Also, they can be used efficiently on channels where the set of input symbols is large. The RS encoding and decoding require a considerable amount of computation and arithmetical operations over a finite number system with certain properties that is algebraic systems, which in this case is called fields. RS's initial definition focuses on the evaluation of polynomials over the elements in a finite field (Galois field GF).

3.1 Reed Solomon Encoding

The most conventional form of Reed Solomon codes in the terms of the parameters n,k,t and any positive integer m>2.

Where n-k=2t is the number of parity symbols, and t is the symbol error correcting capability of the code. The generating polynomial for RS code

$$g(X) = g_0 + g_1 X + g_2 X^2 + \ldots\ldots + g_{2t-1} X^{2t-1} + X^{2t} \tag{6}$$

The degree of the generator polynomial is equal to the number of parity symbols. RS codes are a subset of Bose Chaudhuri Hocquenghem (BCH) codes. Since the generator polynomial is of degree 2t, there must be precisely 2t successive powers of α that are roots of the polynomial. Message polynomial m(x) can be shifted into the rightmost k stages of a codeword register and then appending a parity polynomial p(X), by placing it in the leftmost n-k stages. 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 right-shifted n-k positions.

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

$$X^{n-k} m(X) = q(X) g(X) + p(X) \tag{7}$$

Where $q(X)$ and $p(X)$ are quotient and remainder polynomials, respectively. The remainder is the parity. Equation (7) can also be expressed as

$$p(X) = X^{n-k}\, m(X) \text{ modulo } g(X) \quad (8)$$

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

$$U(X) = p(X) + X^{n-k}\, m(X) \quad (9)$$

Demonstrating the steps implied by equation (8) and (9) by encoding the following three symbol message: 010 110 111with the (7, 3) RS code. Generator polynomial is

$$g(X) = \alpha^3 + \alpha^1 X + \alpha^0 X^2 + \alpha^3 X^3 + X^4 \quad (10)$$

Message symbol in polynomial form:

$$\alpha^1 + \alpha^3 X + \alpha^5 X^2 \quad (11)$$

Message polynomial is multiplied by $X^{n-k} = X^4$, result is

$$\alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6 \quad (12)$$

Divide equation (12) with equation (10)
Remainder in polynomial form is

$$p(X) = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 \quad (13)$$

Codeword is written as

$$U(X) = \alpha^0 + \alpha^2 + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6 \quad (14)$$

Output codeword can be written as

$$U(X) = \sum_{n=0}^{6} U_n X^n \quad (15)$$

$$U(X) = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6$$
$$= (100) + (001)X + (011)X^2 + (101)X^3 + (010)X^4 + (110)X^5 + (111)X^6 \quad (16)$$

The roots of a generator polynomial g(X) must also be the roots of the codeword generated by g(X), because a valid codeword is of the following form:

$$U(X) = m(X)\, g(X) \quad (17)$$

Therefore, an arbitrary codeword, when evaluated at any root of g(X), must yield zero. In other words, this means checking that

$$U(\alpha) = U(\alpha_2) = U(\alpha_3) ..... = U(\alpha_{n-k}) = 0 \quad (18)$$

This demonstrates the expected results that a codeword evaluated at any root of g(X) must yield zero.

$$U(\alpha) = U(\alpha^2) = U(\alpha^3) = U(\alpha^4) = 0 \quad (19)$$

3.2 Reed Solomon Decoding

RS code resulted in a codeword polynomial. Now we assume that during transmission this codeword is corrupted. This number of errors corresponds to the maximum error correcting capability of the code. Now we will be discussing the decoding by taking example of RS (7, 3), which can correct up to 2 symbols. For this seven-symbol codeword example, the error pattern $e(X)$ can be described in polynomial form as follows:

$$e(X) = \sum_{n=0}^{6} e_n X_n \quad (20)$$

$$e(X) = 0 + 0 + 0X^2 + \alpha^2 X^3 + \alpha^5 X^4 + 0X^5 + 0X^6 \quad (21)$$

In this one parity symbol is corrupted with a 1 bit error and one data symbol is corrupted with a 3 bit error.
The received corrupted codeword polynomial $r(X)$ is then represented by the sum of the transmitted codeword polynomial and the error-pattern polynomial as follows:

$$r(X) = U(X) + e(X) \quad (22)$$

Adding equation (16) and equation (21), we get

$$r(X) = \alpha^0 + \alpha X + \alpha^4 X^2 + \alpha^0 X^3 + \alpha^6 X^4 + \alpha^3 X^5 + \alpha^5 X^6 \quad (23)$$

In this example, there will be two error locations and two error values. The error at a particular location tells that the bit must be "flipped" from 1 to 0 or vice versa. Now the error locations are calculated, also determine the correct symbol values at those locations. Since there are four unknowns in this example, four equations are required for their solution.

3.2.1 Syndrome Computation

The syndrome is the result of a parity check performed on r to determine whether r is a valid member of the codeword set. If in fact r is a member, the syndrome S has value 0. Any nonzero value of S indicates the presence of errors. Thus, for this (7, 3) RS code, there are four symbols in every syndrome vector, their values can be computed from the received polynomial, r(X). Note how the computation is facilitated by the structure of the code, given by equation (17). From this structure it can be seen that every valid codeword polynomial U(X) is a multiple of the generator polynomial g(X). Therefore, the roots of g(X) must also be the roots of U(X). Since from equation (22), r(X) evaluated at each of the roots of g(X) should yield zero only when it is a valid codeword. Any errors will result in one or more of the computations yielding a nonzero result. The computation of a syndrome symbol can be described as follows:

$$S_i = r(X)\big|_{X=\alpha^i} = r(\alpha^i) \quad i = 1, 2, ...n-k \quad (24)$$

where r(X) contains the postulated two symbol errors. If r(X) were a valid codeword, it would cause each syndrome symbol S_i to equal 0.

$$S_1 = r(\alpha) = \alpha^3 \quad (25)$$

$$S_2 = r(\alpha^2) = \alpha^5 \quad (26)$$

$$S_3 = r(\alpha^3) = \alpha^6 \quad (27)$$

$$S_4 = r(\alpha^4) = 0 \quad (28)$$

The results confirm that the received codeword contains an error (which we inserted), since S≠0.
Now a secondary check on the syndrome values is performed. For the (7, 3) RS code example under consideration, the error pattern is known, since it was chosen earlier. An important property of codes when describing the standard array is that each element of a cosset (row) in the standard array has the same syndrome .This property is also true for the R-S code by evaluating the error polynomial e(X) at the roots of g(X) to demonstrate that it must yield the same syndrome values as when r(X) is evaluated at the roots of g(X). In other words, it must yield the same values obtained in syndrome.

$$S_i = r(X)\big|_{X=\alpha^i} = r(\alpha^i) \qquad i = 1, 2, ...n-k \quad (29)$$

$$S_i = r(\alpha^i) = U(\alpha^i) + e(\alpha^i) = 0 + e(\alpha^i) \quad (30)$$

$$S_i = [U(X) + e(X)]\big|_{X=\alpha^i} = U(\alpha^i) + e(\alpha^i) \quad (31)$$

From equation (25)

$$e(X) = \alpha^2 X^3 + \alpha^5 X^4 \quad (32)$$

Therefore,

$$S_1 = e(\alpha^1) = \alpha^3$$
$$S_2 = e(\alpha^2) = \alpha^5$$
$$S_3 = e(\alpha^3) = \alpha^6$$
$$S_4 = e(\alpha^4) = \alpha^4 = 0$$

These results confirm that the syndrome values are the same, whether obtained by evaluating e(X) at the roots of g(X), or r(X) at the roots of g(X).

3.2.2 Error Location

Suppose there are ν errors in the codeword at location $X^{j1}, X^{j2}, ...X^{jv}$. Then, the error polynomial e(X) can be written as follows:

$$e(X) = e_{j1}X^{j1} + e_{j2}X^{j2} + ... + e_{jv}X^{jv} \quad (33)$$

The indices 1, 2 … ν refer to the first, second… vth errors, and the index j refers to the error location. To correct the corrupted codeword, each error value e_{jl} and its location X^{jl} , where $l = 1, 2.....v$ must be determined. We define an error locator number as $\beta_l = \alpha^{jl}$ .Next, we obtain the n - k = 2t syndrome symbols by substituting α^i into the received polynomial for i = 1, 2 … 2t:

$$S_1 = r(\alpha) = e_{j1}\beta_1 + e_{j2}\beta_2 + ...e_{jv}\beta_v$$
$$S_2 = r(\alpha^2) = e_{j1}\beta_1^2 + e_{j2}\beta_2^2 + ... + e_{jv}\beta_v^2$$
$$\vdots$$
$$S_{2t} = r(\alpha^{2t}) = e_{j1}\beta_1^{2t} + e_{j2}\beta_2^{2t} + ...e_{jv}\beta_v^{2t} \quad (34)$$

There are 2t unknowns (t error values and t locations), and 2t simultaneous equations. However, these 2t simultaneous equations cannot be solved in the usual way because they are nonlinear (as some of the unknowns have exponents). Any technique that solves this system of equations is known as a Reed-Solomon decoding algorithm.

Once a nonzero syndrome vector (one or more of its symbols are nonzero) has been computed, that signifies that an error has been received. Next, it is necessary to learn the location of the error or errors. An error-locator polynomial, σ(X), can be defined as follows:

$$\sigma(X) = (1 + \beta_1 X)(1 + \beta_2 X).....(1 + \sigma_v X^v)$$
$$= 1 + \sigma_1 X + \sigma_2 X^2 + ... + \sigma_v X^v \quad (35)$$

The roots of σ(X) are $1/\beta_1, 1/\beta_2 .....1/\beta v$. The reciprocal of the roots of σ(X) are the error-location numbers of the error pattern e(X). Then, using autoregressive modeling techniques, we form a matrix from the syndromes, where the first t syndromes are used to predict the next syndrome. That is,

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots S_{t-1} & S_t \\ S_2 & S_3 & S_4 & \cdots S_t & S_{t+1} \\ S_{t-1} & S_t & S_{t+1} & \cdots S_{2t-3} & S_{2t-2} \\ S_t & S_{t+1} & S_{t+2} & \cdots S_{2t-2} & S_{2t-1} \end{bmatrix} \begin{bmatrix} \sigma_t \\ \sigma_{t-1} \\ \vdots \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{t+1} \\ -S_{t+2} \\ -S_{2t-1} \\ -S_{2t} \end{bmatrix} \quad (36)$$

We apply the autoregressive model of Equation (36) by using the largest dimensioned matrix that has a nonzero determinant. For the (7, 3) double symbol error correcting RS code, the matrix size is 2 × 2, and the model is written as follows:

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_3 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix}$$
$$\begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^6 \\ 0 \end{bmatrix} \quad (37)$$

To solve for the coefficients σ1 and σ2 and of the error-locator polynomial, σ(X), we first take the inverse of the matrix. The inverse of a matrix [A] is found as follows:

$$Inv[A] = \frac{cofactor}{\det[A]}$$
$$Inv\begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} = \begin{bmatrix} \alpha^1 & \alpha^0 \\ \alpha^0 & \alpha^5 \end{bmatrix} \quad (38)$$

Now safety check for this is done. If the inversion was performed correctly, the multiplication of the original matrix by the inverted matrix should yield an identity matrix.

Continuing from Equation (37), the error locations are found by solving for the coefficients of the error-locator polynomial, σ(X).

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^1 & \alpha^0 \\ \alpha^0 & \alpha^5 \end{bmatrix} \begin{bmatrix} \alpha^6 \\ 0 \end{bmatrix} = \begin{bmatrix} \alpha^0 \\ \alpha^6 \end{bmatrix} \quad (39)$$

From equation (35) and equation (39)

$$\sigma(X) = \alpha^0 + \alpha^6 X + \alpha^0 X^2 \quad (40)$$

Error locations are found by solving for the coefficients of the error-locator polynomial, σ(X).The roots of σ(X) are the reciprocals of the error locations. Once these roots are located, the error locations will be known. In general, the roots of σ(X) may be one or more of the elements of the field. We determine these roots by exhaustive testing of the σ(X) polynomial with each of the field elements, as shown below. Any element X that yields σ(X) = 0 is a root, and allows us to locate an error.

The error locations are at the inverse of the roots of the polynomial. Therefore $\sigma(\alpha^3) = 0$ indicates that one root exits at $1/\beta_l = \alpha^3$. Thus, $\beta_l = 1/\alpha^3 = \alpha^4$. Similarly, $\sigma(\alpha^4) = 0$ indicates that another root exits at $1/\beta_{l'} = \alpha^4$. Thus $\beta_{l'} = 1/\alpha^4 = \alpha^3$, where *l* and *l′* refer to the first, second … vth error. Therefore, in this example, there are two-symbol errors, so that the error polynomial is of the following form:

$$e(X) = e_{j1}X^{j1} + e_{j2}X^{j2} \quad (41)$$

The two errors were found at locations α_3 and α_4. Note that the indexing of the error-location numbers is completely arbitrary. Thus, for this example, we can designate the $\beta_l = \alpha^{jl}$ values as $\beta_1 = \alpha_3$ and $\beta_2 = \alpha_4$.

3.2.3 Error Values

An error had been denoted e_{jl}, where the index j refers to the error location and the index l identifies the l^{th} error. Since each error value is coupled to a particular location, the notation can be simplified by denoting e_{j1}, simply as e_1 Preparing to determine the error value e_1 and e_2 associated with locations β_1 and β_2. Any of the four syndrome equations can be used. Let us use syndrome 1 and 2.

$$S_1 = r(\alpha) = e_1\beta_1 + e_2\beta_2$$
$$S_2 = r(\alpha^2) = e_1\beta_1^2 + e_2\beta_2^2 \quad (42)$$

We can write these equations in matrix form as follows:

$$\begin{bmatrix} \beta_1 & \beta_2 \\ \beta_1^2 & \beta_2^2 \end{bmatrix}\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$
$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \alpha^2 \\ \alpha^5 \end{bmatrix} \quad (43)$$

Similarly, by above method e_1 and e_2 are found, $e_1 = \alpha^2$ and $e_2 = \alpha^5$.

3.2.4 Correcting the Received Polynomial with Estimates of the Error Polynomial

From equation (43) and equation (41) and error values, the estimated error polynomial is formed, to yield the following:

$$\hat{e}(X) = e_1 X^{j1} + e_2 X^{j2}$$
$$= \alpha^2 X^3 + \alpha^5 X^4 \quad (44)$$

The demonstrated algorithm repairs the received polynomial, yielding an estimate of the transmitted codeword, and ultimately delivers a decoded message. It is given as

$$\hat{U}(X) = r(X) + \hat{e}(X) \quad (45)$$

$$\hat{e}(X) = (000) + (000)X + (000)X^2 + (001)X^3 + (111)X^4 + (000)X^5 + (000)X^6$$
$$r(X) = (100) + (001)X + (011)X^2 + (100)X^3 + (101)X^4 + (110)X^5 + (111)X^6$$
$$\hat{U}(X) = (100) + (001)X + (011)X^2 + (101)X^3 + (010)X^4 + (110)X^5 + (111)X^6$$
$$= \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6 \quad (46)$$

Since the message symbols constitute the rightmost k = 3 symbols, the decoded message is 010 110 111which is exactly the test message that was chosen earlier for this example [5] [6].

4. RESULT

Layout design contains several buttons. For data selection BROWSE button. Two Pop-up-menu buttons one for selecting error probability for BSC and other for selecting SNR value for AWGN channel. HAM ENC, LB ENC and RS ENC are buttons for encoding. HAM BSC D,LB BSC D and RS BSC D are buttons for decoding when data passed from BSC. HAM AGN D, LB AGN D and RS AGN D are buttons for decoding when data passed from AWGN channel.BER BSC button will plot BER vs. error probability for BSC.BER AWGN button will plot BER vs. SNR for AWGN channel.Fig1. shows the layout design.

Data passed without using codes shown in Fig2.Steps followed when data passed without using error correcting codes:

Step1: Data is taken from the BROWSE button. Then DATA button selected.

Step2: To pass data from BSC error probability is selected from popupmenu1 and then selected BSC button.BER of the resulted output on the axis and data is seen at BER DT BSC button.

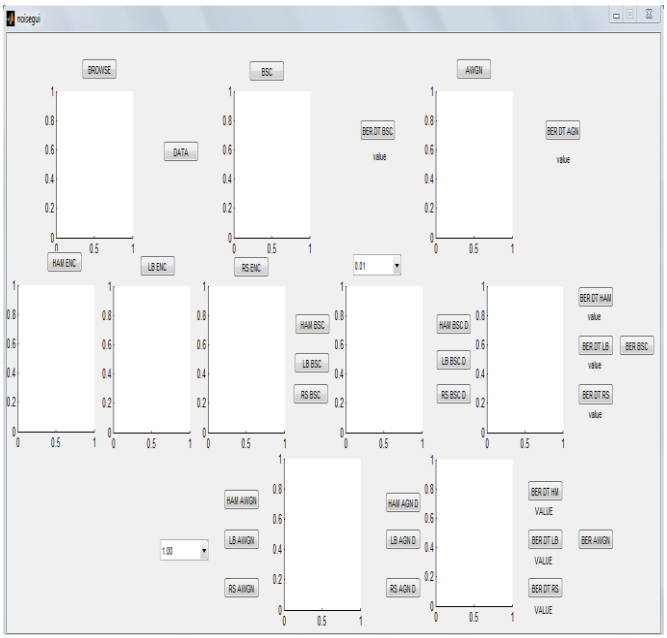

Fig 1: Layout design

Step3: Similarly for AWGN channel SNR value selected from popupmenu2 then AWGN button selected output is shown on third axis.

Step4: BER of data and output of AWGN channel is seen by clicking BER DT AGN Button.

Fig.3 shows data passed through BSC using codes. Fig.4 shows data passed through AWGN using codes. Layout design steps followed when data passed with using error correcting codes:

Step1: Data is selected from BROWSE button, then on DATA button. Then HAM ENC for Hamming (11, 7), LB ENC for Hamming (7, 4) and RS ENC for RS (7, 3) button selected for the encoding required.

.

Step2: Then to choose BSC to add noise to the encoded data, value from popupmenu1 is selected and then HAM BSC for

Hamming (11, 7), LB BSC for Hamming (7, 4) and RS BSC for RS (7, 3) encoding button is selected. For adding AWGN noise to the data SNR value selected from popupmenu2,HAM AW… selected to add noise to the Hamming(11,7), LB AWGN for Hamming(7,4)and RS AWGN for RS (7,3).

Step3: For decoding of encoded data passed from BSC for Hamming (11, 7) HAM AG… is selected, for Hamming (7, 4) LB AGN D is selected and for RS (7, 3) RS AGN D is selected. For decoding of encoded data passed from AWGN for Hamming (11, 7) HAM BS… is selected, for Hamming (7, 4) LB BSC D is selected and for RS (7, 3) RS BSC D is selected.

Step4: To see the BER for data decoded and data sent push buttons are there.

Step5: BER BSC selected to plot BER versus error probability for BSC.

Step6: BER AW selected to plot BER versus SNR for AWGN.

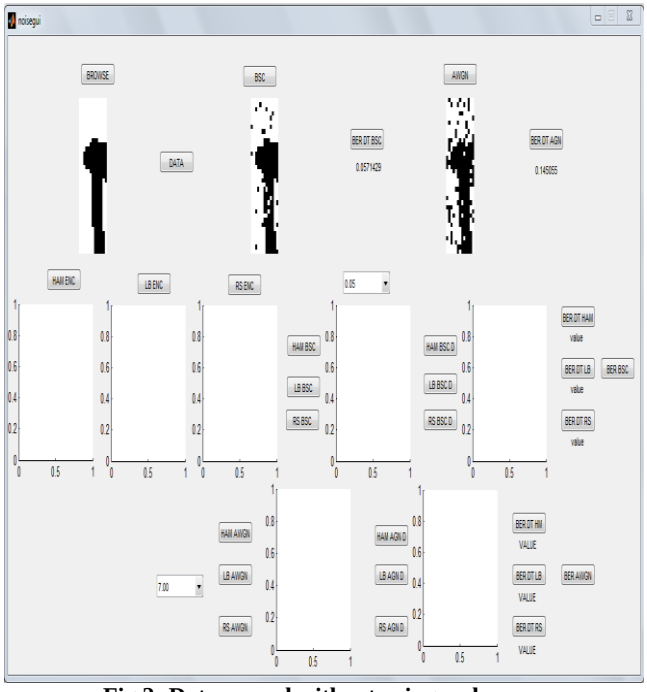

Fig 2: Data passed without using codes

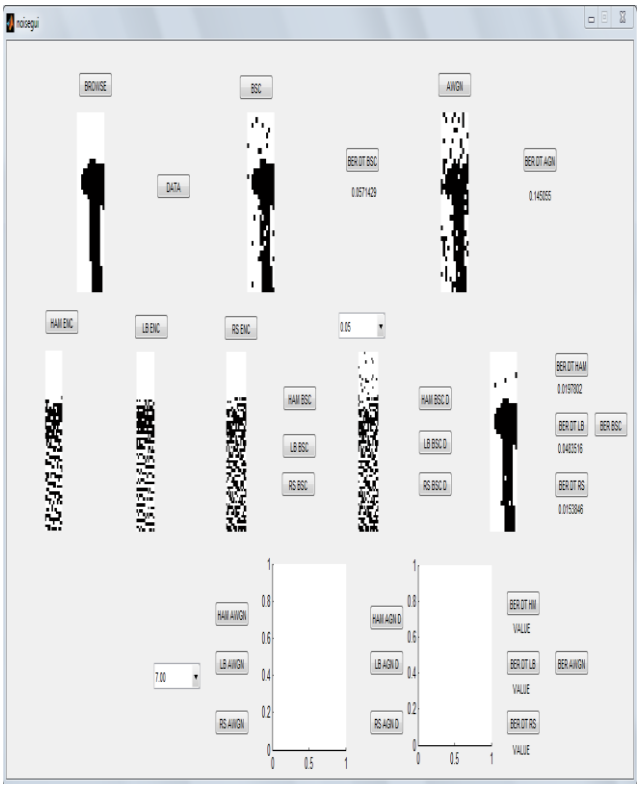

Fig 3: Data passed through BSC using codes

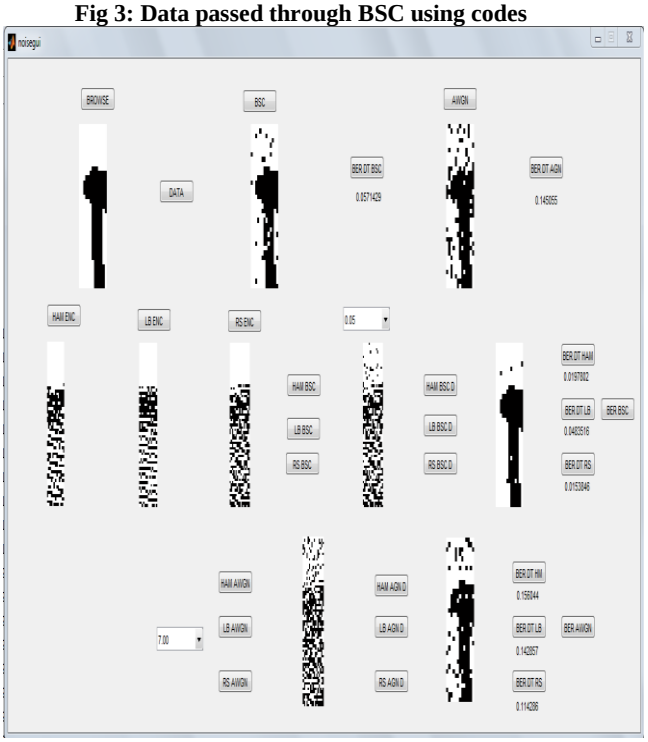

Fig 4: Data passed through AWGN using codes

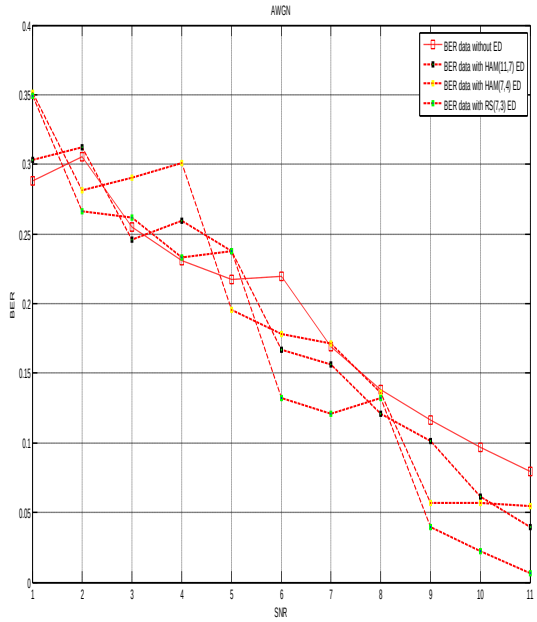

Fig 5: BER versus SNR in AWGN

5. CONCLUSION

The communication system is to communicate from one place to another in any way at any time whether using internet, mobile, television broadcasting, military services and in many more ways, if receiver receives corrupted data then it is of no use to the receiver. Data security is integral part of it. The solution is error detection and correction. Results shows that RS codes are better than Hamming code. BER is less in RS (7, 3) as compared to Hamming (11, 4) and Hamming (7, 4).

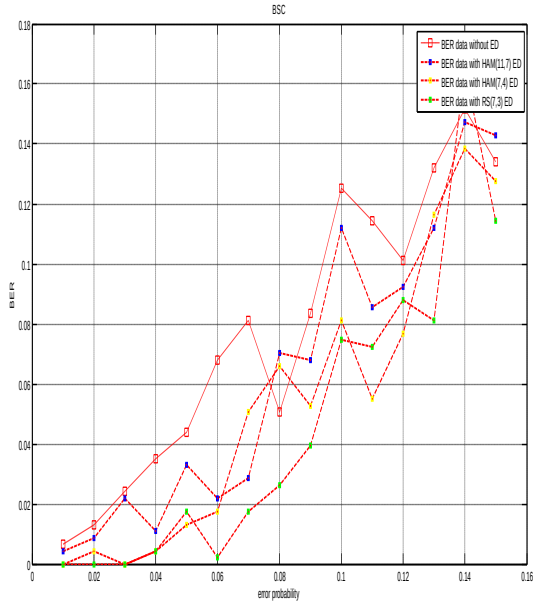

Fig 6: BER versus error probability in BSC

In the present study we investigated the performance of Reed Solomon codes as a flexible single code. In future the comparison will be shown by taking more error correcting codes and more noises can be added to it. The BER performance improves as the code rate decreases. The BER performance also improves for large block lengths and RS codes shows a poor BER performance for lower SNR. As the SNR value increases the curve becomes steeper [7]. Further BCH codes can be added for the comparison. Length of message can vary in Hamming and RS codes for better results of error correction.

6. REFERENCES

[1] Muzhir AL-ANI and Qeethara AL-SHAYEA "Unidirectional Error Correcting Codes for Memory Systems: A Comparative Study", January 2010.

[2] Robert H.Morelos-Zaragoza "The art of error correcting codes", Second Edition, John Block Codes", January 2000.

[3] U. K. Kumar and B. S. Umashankar "Improved Hamming Code for Error Detection and Correction", Member IEEE.

[4] Investigation of Hamming, Reed-Solomon, and Turbo Forward Error Correcting Codes by Gregory Mitchell Sensors and Electron Devices Directorate, ARL, July 2009.

[5] Bernard Sklar "Digital Communications: Fundamentals And Applications", Second edition.

[6] Hamood Shehab and Widad Ismail "Hardware Implementation for Error Correction Using Software Defined Radio Platform", EuroJournals Publishing, Inc.2009.

[7] Sanjeev Kumar and Ragini Gupta "Bit Error Rate Analysis of Reed-Solomon Code for Efficient Communication System ", International Journal of Computer Applications (0975 – 8887) Volume 30– No.12, September 2011