RESEARCH ARTICLE OPEN

Design and Performance Analysis of Convolutional Encoder and Viterbi Decoder for Various Generator Polynomials

Deepa Kumari*, Madan Lal Saini**
M. Tech. Scholar, Dept of CS/IT, Jagannath University Jaipur, Rajasthan, India
Asst. Prof., Dept. of CS/IT, Jagannath University Jaipur, Rajasthan, India

ABSTRACT
In digital communication forward error correction methods have a great practical importance when channel is noisy. Convolutional error correction code can correct both type of errors random and burst. Convolution encoding has been used in digital communication systems including deep space communication and wireless communication. The error correction capability of convolutional code depends on code rate and constraint length. The low code rate and high constraint length has more error correction capabilities but that also introduce large overhead. This paper introduces convolutional encoders for various constraint lengths. By increasing the constraint length the error correction capability can be increased. The performance and error correction also depends on the selection of generator polynomial. This paper also introduces a good generator polynomial which has high performance and error correction capabilities.

Keywords : *Convolutional Encoder, Code Rate, Constraint Length (CL), Generator Polynomials (GP), Viterbi Decoder.*

I. INTRODUCTION

The error correction codes are used in digital communication and encoding and decoding task are performed by channel encoder/ decoder. Convolutional codes are used in digital communication system like GSM and are implemented by channel encoder. A Convolution Encoder accepts an input stream of message and generates encoded output streams to be transmitted. In this process for one input bit the encoder generates more than one output bits and these redundant symbols in output bit pattern makes the transmitted data more immune to the noise in the channel. The redundant bits help to decide and correct the errors in received pattern. Convolutional codes basically used in space communication or in very noisy channel. It can correct burst error as well as random errors.

In this paper we introduce a strategy to present convolutional code such that they can correct maximum number of errors. This strategy is the selection of generator polynomial and code rate, the selection of generator polynomial is performed in MATLAB. Second section describe convolutional encoder parameters, viterbi decoder and generator polynomials. Third section introduces selection of generator polynomial and code rate, and fourth section gives performance analysis.

II. CONVOLUTIONAL ENCODER

A convolutional encoder can be described by these following parameters (N, K, M) as summarized:
N: Number of output symbols.
K: Number of input symbols.
M: Length of the shift registers stage in the encoder or number of shift register.

Constraint Length (L) = (M+1) or M: This number represents the number of input bits required to generate a unique output pattern in the encoder. A constraint length of L=3 means that each output symbol depends on the current input symbol and the two previous input symbols. The constraint lengths of the encoder form a vector whose length is the number of inputs in the encoder diagram. The elements of this vector indicate the number of bits stored in each shift register, *including* the current input bits. In the figure-1 given below, the constraint length is three. It is a scalar because the encoder has one input stream, and its value is number of shift registers for that input.**[1]**

Number of States = $2^{(L-1)}$: Defines the maximum number of states that is possible to be mapped by the combinations of the L number of input bits for the convolution encoder.

Convolution Code Rate R: Number of input bits to create a symbol at the output (k)/ Number of output bits in a symbol at the output (n). For example, 1/2 code rate means each bit entering the encoder results in 2 bits leaving the encoder.[2]

The encoder has N modulo-2 adders, and N generator polynomials one for each adder. This process multiplies the number of input bits at the output. For example, a 4-bit input is converted into an N*4-bit output, 8-bit input into an N*8- bit output and so on. **[3]**

A. Viterbi Decoder

Viterbi decoding was developed by Andrew J. Viterbi, is an Italian-American electrical engineer. In his seminar paper titled "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", published in IEEE Transactions on Information Theory, in April, 1967. **[4]**

The basic units of viterbi decoder are branch metrics, Add compare select and Survivor management unit. Viterbi decoder consist of three blocks: the branch metric unit (BMU), which computes metrics, the add–compare–select unit (ACSU), which selects the survivor paths for each trellis state, also finds the minimum path metric of the survivor paths and the survivor management unit (SMU), that is responsible for selecting the output based on the minimum path metric. Viterbi algorithm is called optimum algorithm since it minimizes the probability of error. The Viterbi algorithm can be explained briefly with the following three steps.

1. Weigh the trellis; that is, calculate the branch metrics.

2. Recursively computes the shortest paths to time n, in terms of the shortest paths to time n-1.
In this step, decisions are used to recursively update the survivor path of the signal. This is known as add-compare-select (ACS) recursion.

3. Recursively finds the shortest path leading to each trellis state using the decisions from Step 2. The shortest path is called the survivor path for that state and the process is referred to as survivor path decode. Finally, if all survivor paths are traced back in time, they merge into a unique path, which is the most likely signal path. [5][6]

B. Bit Error Rate/Ratio

In digital communication, during transmission the bits usually gets altered, i.e., the transmitted '1' is interpreted as '0' & transmitted '0' is interpreted as '1', this constitutes as a bit error. This happens due to various factors, like noise, fading etc. The bit error rate (BER) is the number of bit errors divided by the total number of transmitted bits over a channel. BER although unit-less also expressed in terms of percentage.
E.g., let the transmitted bit sequence:
1 1 1 0 0 0 1 1 1 0
& the received bit sequence:
1 1 $\underline{0}$ 0 $\underline{1}$ 0 1 1 $\underline{0}$ 0 ,

Then the number of bit errors in this case is 3. The BER is 3 incorrect bits divided by 10 transmitted bits, resulting in a BER of 0.3 or 30%.

C. Additive White Gaussian Noise (AWGN)

AWGN is a basic channel model used to study the effect of random processes that occurs in nature.

- It's assumed to be 'Additive', so the modeling will be easy.
- 'White' refers to idea that it has uniform power across the frequency band.
- 'Gaussian' because it has a normal distribution in the time domain, with average value zero.

D. Generator Polynomial:

A generator polynomial specifies the encoder connections. In another words, the generator polynomial can be deduced as the mathematical description of the convolution encoder. Each polynomial forming the generator polynomial should be at most K degree and specifies the connections between the shift registers and the modulo-2 adders.

Two generator polynomials are $g_1(x)=1+x^2$, $g_2(x)=1+x+x^2$ which give $g_1(x)=(101)$ and $g_2(x)=(111)$. **[3]**

If the encoder diagram has *k* inputs and *n* outputs, then the code generator matrix is a *k*-by-*n* matrix. The element in the ith row and jth column indicates how the ith input contributes to the jth output.

For systematic bits of a systematic feedback encoder, match the entry in the code generator matrix with the corresponding element of the feedback connection vector. In other situations, it can be determined ((i, j) entry in the matrix) as follows:

1. Build a binary number representation by placing a 1 in each spot where a connection line from the shift register feeds into the adder, and a 0 elsewhere. The leftmost spot in the binary number represents the current input, while the rightmost spot represents the oldest input that still remains in the shift register.

2. Convert this binary representation into an octal representation by considering consecutive triplets of bits, starting from the rightmost bit. The rightmost bit in each triplet is the least significant. If the number of bits is not a multiple of three, then place zero bits at the left end as necessary. (For example, interpret 1101010 as 001 101 010 and convert it to 152.)

III. SELECTION OF GENERATOR POLYNOMIAL AND CODE RATE

A. *Structure of Convolutional Code*

The convolutional code structure is easy to draw from its parameters. First draw m boxes representing the m memory registers. Then draw n modulo-2 adders to represent the n output bits. Now connect the memory registers to the adders using the generator polynomial as shown in the Fig. 1.

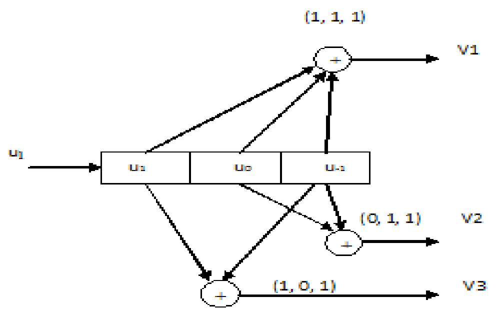

Figure 1 - This (3, 1, 3) convolutional code has 3 memory registers, 1 input bit and 3 output bits.

From Fig 1, the generator vectors for v1 adder are [111], for v2 adder [011], and for v3 adder [101], which gives corresponding generator polynomials $g_1(x) = 1+x+x^2$, $g_2(x) = x+x^2$, $g_3(x) = 1+x^2$. This is a rate 1/3 code. Each input bit is coded into 3 output bits. The constraint length of the code is 3. The 3 output bits are produced by the 3 modulo-2 adders by adding up certain bits in the memory registers. The selection of which bits are to be added to produce the output bit is called the generator polynomial g(x) for that output bit. For example, the first output bit has a generator polynomial of (1, 1, 1). The output bit 2 has a generator polynomial of (0, 1, 1) and the third output bit has a polynomial of (1, 0, 1). The polynomials give the code its unique error protection quality. One (3, 1, 4) code can have completely different properties from another one depending on the polynomials chosen.

When code rate is minimized, the error correction capability is increased. A very low code rate is used in deep space communication where noise level is high. In this research paper a code rate 1/3 is used.

B. *How Polynomials are Selected*

There are many choices for polynomials for any m order code. They do not all result in output sequences that have good error protection properties. Petersen and Weldonís book contains a complete list of these polynomials.**[7]** Good polynomials are found from this list usually by computer simulation/MATLAB.**[8]** There is no known constructive way for selection of generator polynomials, however a convolutional code can be analysed to find its distance properties. A good convolutional code has large free hamming distance (difference of bits in two code words at same positions). The small Hamming distance produces the minimally separated code sequence. Thus for a study of distance properties it is possible to focus on computer simulation programs.

A list of good polynomials for rate 1/3 codes is given below. For selection of good polynomial MATLAB simulation is used, a random sequence of binary bits of 200 lengths is generated. Now convolutional encoding is performed for code rate 1/3 and constraint length 3. After encoding 10 random errors are added in code words. For different polynomial combination the Bit Error Ratio (BER) is calculated. On the basis of BER the generator polynomial are selected which gives low or minimum BER.

Table 1 gives the BER value for MaxErrors =10, random/ burst errors injected in communication. Table 2 gives BER values for MaxErrors = 20, and table 3 for constraint length 3 and MaxErrors =20. Among all these set of polynomials, best polynomials are selected which give minimum BER, these polynomials are given in table 4.

Table 1-Good Generator Polynomials found for code rate = 1/3, MaxErrors =10

Constraint Length	**G1**	**G2**	**G3**	**Octal Value**	**BER**
3	001	101	111	1,5,7	0.05
	011	100	111	3,4,7	0
	010	101	111	2,5,7	0
	111	010	111	7,2,7	0
	110	101	111	6,5,7	0
	101	100	111	5,4,7	0.05
	111	010	111	7,4,7	0

Table 2-Good Generator Polynomials found for code rate = 1/3, MaxErrors =20

Constraint Length	**G1**	**G2**	**G3**	**Octal Value**	**BER**
3	010	011	111	2,3,7	0.20
	101	010	111	5,2,7	0.30
	111	011	111	7,3,7	0.35
	111	001	110	7,1,6	0.15
	111	101	100	7,5,4	0.15
	111	110	101	7,6,5	0.10
	101	111	010	5,7,2	0.20

Table 3-Good Generator Polynomials found for code rate = 1/3, MaxErrors =20, CL=4

Constraint Length	**G1**	**G2**	**G3**	**Octal Value**	**BER**
4	0011	0101	1111	3,5,17	0.20
	0110	0101	1111	6,5,17	0.15
	1000	0101	1111	10,5,17	0.15
	1001	0101	1111	11,5,17	0.15
	0110	1101	1111	6,15,17	0.20
	1111	0010	0100	17,2,4	0.15
	1111	0110	0001	17,6,1	0.20
	1111	1101	0011	17,15,3	0.15
	1111	0101	0010	17,5,2	0.20

Table 4-Best Generator Polynomial for code rate = 1/3

CL	Max-Err-ors	G1	G2	G3	Octal Value	BER
		110	101	111	6,5,7	0.0
3	10	011	100	111	3,4,7	0.0
		111	110	101	7,6,5	0.1
3	20	101	111	010	5,7,2	0.20
		1111	1101	0011	17,15,3	0.15
4	20	1110	1011	1101	16,13,15	0.15

IV. PERFORMANCE ANALYSIS

MATLAB Implementation

In this work, we generate a random stream of binary bits of length 2000. SNR values are set from k=0 to 10, and noise values are set as 1/10^(SNR(k)/30). BPSK modulation is applied and signals are passed through noisy channel.[9] For every value of SNR we have generated 1000000 bits in order to have enough number of bits to generate a nearly real life curve. Also for each SNR value we have waited for 5 bit errors to occur before proceeding to the other SNR value, as to save the execution time. Later on all bit errors that occurred are added & the number of bits generated are also added & divided by the former to obtain BER [10].

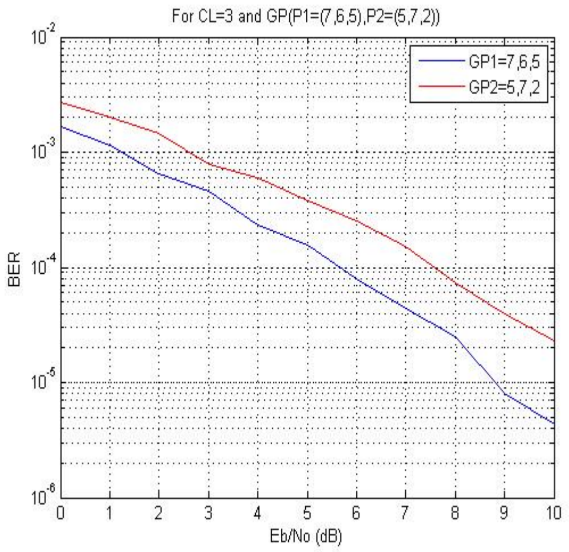

Figure 2. BER performance for GP1=7,6,5 and GP2=5,7,2

Error performance analysis is performed by plotting the bit error ratio (BER) versus signal to noise ratio (SNR) for AWGN channel. The SNR is used to represent signal to noise ratio where signal power is energy of one bit. The parameter E_b/N_0 (the energy per bit to noise power spectral density ratio) is taken for performance measurement. Simulations were run for different generator polynomials and constraint length.

The best polynomials are selected from tables 1 to 3, and are presented in table 4. For these polynomials (table 4) the performance is plotted in terms of BER and E_b/N_0. In figure 2 we can see that generator polynomial (7,6,5) has good performance over (5,7,2). Same we can see in figure 3 and 4, it means generator polynomial affects the performance of code.

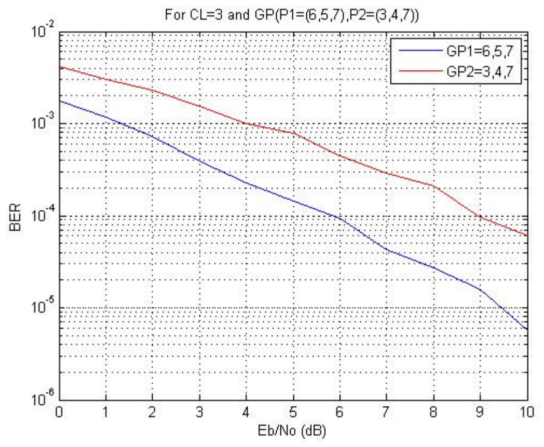

Figure3. BER performance for GP1=6,5,7 and GP2=3,4,7

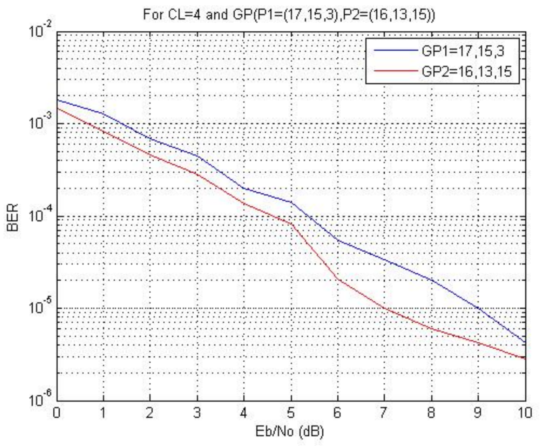

Figure4.BER performance for GP1=17,15,3 and GP2=16,13,15

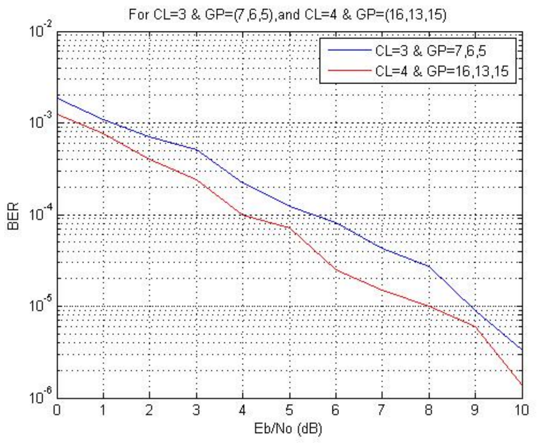

Figure 5. BER performance for CL=3, and CL=4

V. Results And Conclusions

While designing a convolutional encoder, the parameters code rate, constraint length, and generator polynomial are considered. It can be seen from tables 1 to 4 that different generator polynomials have different BER values. A good generator polynomial minimizes the BER. In figures from 2 to 4, we can see that different polynomials have different BER. In this work we selected the polynomial (7,6,5) for constraint length 3 and (16,13,15) for constraint length 4 as the best polynomials. We designed a Viterbi decoder for these generator polynomials and performance is plotted for constraint length 3 and 4. In figure 5, it can be seen that when constraint length is increased, the BER is decreased.

REFERENCES

[1] Ripple Dhingra1 and Danvir Mandal2,"Convolutional Code Optimization for Various Constraint Lengths using PSO", International Journal of Electronics and Communication Engineering. ISSN 0974-2166 Volume 5, Number 2 (2012), pp. 151-157.

[2] Ioannis A. Chatzigeorgiou, Miguel R. D. Rodrigues, Ian J. Wassell, and Rolando A. Carrasco, "Comparison of Convolutional and Turbo Coding for Broadband FWA Systems", IEEE TRANSACTIONS ON BROADCASTING, 0018-9316, 2007 IEEE.

[3] Mahe Jabeen, Salma Khan, "Design of Convolution Encoder and Reconfigurable Viterbi Decoder", International Journal of Engineering and Science ISSN: 2278-4721, Vol. 1, Issue 3 (Sept 2012), PP 15-21

[4] Andrew J. Viterbi, Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, IEEE Transactions on Information Theory, Volume IT-13, in April, 1967 pages 260-269.

[5] K. S. Arunlal and Dr. S. A. Hariprasad, "AN EFFICIENT VITERBI DECODER", International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.1, February 2012.

[6] Wong, Y., Jian, W., HuiChong, O., Kyun, C., Noordi, N. "Implementation of Convolutional Encoder and Viterbi Decoder using VHDL", Proceedings of IEEE International conference on Research and Development Malaysia, November 2009.

[7] W.W. Peterson and E. J. Weldon, Jr., Error Correcting Codes, 2nd Edition Cambridge, MA: The MIT Press, 1972.

[8] Davoud Arasteh ,"Teaching Convolutional Coding using MATLAB in Communication Systems Course", Proceedings of the 2006 ASEE Gulf-Southwest Annual Conference Southern University and A & M College.

[9] J. H. Yuen, "Modulation and Coding for Satellite and Space Communications", IEEE Procedings, Vol. 78, No. 7, pp. 1250-1266, July 1990.

[10] J. G. Proakis, M. Salehi, G. Bauch, "Contemporary Communication Systems Using MATLAB and Simulink ", Second Edition, Thomson, 2004.