Vaibhav Chugh

A wall of curiousity for finding what's new and better

Wireless Communications

QUANTITATIVE ANALYSIS ON THE COMPARATIVE STUDY OF VARIOUS ERROR CORRECTION TECHNIQUES

Abstract:

With the ever increasing pace of growth of the wireless communication industry, coping with transmission errors is becoming substantially important. The channel noise, fading, attenuation and interference caused in a transmission media impediments the quality of data that is being transmitted through it hence posing challenges for reliable data communication. In this paper, I wish to compare some of the major techniques of forward error correction using Block codes, Turbo codes, LDPC codes, Reed – Solomon Codes and Convolutional codes. Some novel error correction techniques will also be studied and compared against the other techniques in literature. Quantitative analysis will be done by implementing the channel encoder and decoder in Matlab and performance will be quantified based on the response of the implemented systems for different Bit error rates and channel error models like the random error and the and burst error model. Bit correction at the decoder will be done by detecting the erroneous position followed by bit inversion and characteristics of input versus output BERs will be plotted for various different schemes. All simulations are done in Matlab with a BPSK modulation scheme over an AWGN channel.

I. Introduction:

WIRELESS Communication has evolved prominently over the years and with the modern wireless communication techniques, the world is becoming a smaller place to live in. These communication systems are easy to setup, configure and install and provide mobility at a low cost. But since nothing comes for free, it is characterized by high bit error rates which can be brought about by random processes in the nature like noise, quantization errors, attenuation, bit synchronization errors, interference with other signals and distortion. Despite these factors which act as deterrents to the data quality, it is crucial to have error correction schemes in place for certain kinds of communication which have stringent reliability and quality requirements. This project aims at analyzing the different error correction schemes that exist in the literature, some of which could just be lying in the dusty library shelves while others are very extensively used in present day communication systems. Through this work, I intend to quantitatively decide the kind of error coding technique that will suit different kinds of wireless communication applications. The performance of digital data transmission is often quantified in terms of the BER versus SNR. The error correction techniques can be broadly categorized into block codes and convolutional codes.

This paper has been organized as follows. First we discuss some novel error correction techniques and simulate their respective models using Matlab and see how these novel error correction techniques play out in error channels for different conditions. Then we estimate the coding gain of these techniques and compare it with the existing ones like Turbo codes. Section II describes error

Curriculam Vitae

Projects

Honors and Awards

Research

About the webpage

This webpage is a combination of my personal and professional goals and achievements and is used to highlight my written discourse.

Wise Words

" There are three ingredients in the good life: learning, earning and yearning."

-Christopher Morley

Copyrights

Search Box

Search

correction techniques based on CRC codes. We discuss one-bit and two-bit error correction techniques and see how these can be extended for multi-bit error correction. In Section III, another novel error correction technique is discussed, which is based on a modified Viterbi algorithm and extended trellis assisted with Cyclic Redundancy check and Bit Stuffing. In Section IV, we discuss the Reed Solomon coding technique, which is efficient for Burst error correction. Finally, the two strongest error correction techniques namely the LDPC and the Turbo code are discussed in Section V. The simulation parameters and the results are discussed in the corresponding sections and the graphs are plotted which help in estimating the performance of each of the coding techniques.

II. CRC Based Error Correction

A. INTRODUCTION

Cyclic Redundancy Check or CRC is a technique which is primarily used in computer networks as an international standard for detecting errors in a transmitted packet. CRC was first developed by CCITT which is now better known as ITU-T and is considered as the most powerful of the known redundancy based error detection techniques. CRC is typically used in most of the data link layer and TCP/IP protocols like User datagram protocol, Point to point protocol and Transmission Control protocol. In this technique, a block of data is taken and some redundant bits are added based on a long division of a message packet with a CRC polynomial. The remainder is then appended with the message and transmitted. At the receiver's end, a similar computation is done wherein the received codeword is again divided by the CRC polynomial and the remainder is obtained. If the obtained remainder is zero, then the received codeword is regarded as error-free and if the computed remainder is computed to be non-zero, the received message is regarded as erroneous. This method is typically used to detect the presence of an error in the received codeword but it does not have the capability to correct a faulty reception.

B. ERROR CORRECTION USING CRC

A novel method based on CRC has been introduced in [11] which is able to detect the error, guess the location of error based on the computed CRC remainder and corrects them. Although there are many more methods of error correction based on CRC like the ones mentioned in [10], [12] and [13] but in this paper we will do a thorough review of the technique mentioned in [11] and will contemplate on the pros and cons and will quantitatively analyze the scope of this technique.

The approach mentioned in [11] primarily deals with the correction of errors based on the remainder computed at the receiver's end. Before sending the data from the transmitter's end, a checksum is calculated and appended with the original block of data and is then transmitted over the channel. After receiving the codeword, the receiver calculates the checksum of the received codeword and checks the remainder generated. If the remainder is 0 then the received codeword is error-less. If there is a non-zero remainder, the value of the remainder is used to decipher the location of the error. At the receiver, it is difficult to know the number of errors that occurred during transmission. This remainder value will be unique for all the individual location of the bits if there is a single bit error in the received codeword. We exploit this uniqueness of the checksum to flip the bit at the location corresponding to the value of the remainder. So it means that for an n-bit code-word, we can have 2n total unique remainders. This places a constraint on the length of the codeword that can be used to uniquely identify an error location. For example a CRC generator polynomial for CCITT CRC-16, X16 + X12 + X5 + 1 which has a degree 16 can have 216 possible values of remainder generated as the checksum. Thus we can CRC encode an information sequence of length 2n – 16 -1. This is because the 16 bits will be lost in appending the CRC checksum and 1 remainder will be a zero which would correspond to a correctly received codeword.

We explain this with the example of the CRC polynomial X5+X3+1 which is a generator polynomial of order 5. Here, we can use a message sequence of a

maximum length 25-5-1.
We have developed a Matlab program to quantify the technique proposed in the paper [11].

Remainder	Error Location	Remainder	Error Location
1	1	17	11
2	2	18	31
3	19	19	18
4	3	20	8
5	6	21	23
6	20	22	29
7	12	23	27
8	4	24	22
9	30	25	26
10	7	26	10
11	28	27	17
12	21	28	14
13	9	29	15
14	13	30	25
15	24	31	16
16	5		

Table-1. A look up table for the remainders and corresponding bit error locations for a single bit error correction using CRC.

Click Here to Download the Matlab code for the Graph shown above

The table shown above shows the error locations that correspond to the respective single error bit location. So, if at the receiver, when the checksum is computed and found to be non-zero, the receiver look up table is referred to for the error location that relates to the bit in error. The bit value is then flipped and the checksum is computed again to check for a zero remainder. If the remainder thus calculated turns out to be zero, we infer that there was only one bit in error and the required corrective measures have been taken to fix the received code word. If the remainder is computed to be non-zero again, it can be inferred that the received code word had a multiple bits in error. This issue can be further resolved by forming another table that pertains to two bit error correction discussed in the section II.C.

C. TWO-BIT ERROR CORRECTION USING CRC

The CRC based error correction approach can be used to correct multiple bit errors as well, but there is an issue that has to be addressed. In a two-bit error correction scenario using this scheme, unlike the case where we had a one remainder which had a one to one correspondence to an error location, will have some remainders which get repeated for a pair of one such doublet of error locations. These are called non-unique remainders and show up even if we make sure that the information sequence is small enough to fit in all possible cases of double bit errors that can show up. A CRC generator polynomial of order n can have 2n unique remainders. Thus for a message sequence of length L including the CRC checksum, the total number of possible two bit errors that we can have is LC2. Thus, while choosing a CRC generator we make sure that the total number of possible two bit error messages fall well within the total number of

available remainders that can be generated. For example, for a 7 bit CRC generator, the total number of possible remainders is 27 or 128. Thus the maximum length of message that can be used to encode ensuring a proper detection of a dual- bit error can be calculated by the equation

$$LC2 \leq 27 \quad (1)$$

On solving, we get $L \leq 16.44$. Thus we can take an information sequence of length 16 or less. For our simulations in Matlab, we use the CRC polynomial X7+X3+1 which will append 7 checksum bits to the information sequence and we reserve 8 bits for the information sequence. It was observed that in the case of dual bit error-correction using CRC, even though we choose a remainder space larger than that of the total number of possible double bit errors, we have some non-unique remainders for a doublet of error locations. The results of the experiment are shown in the table below.

Error Location 1	Error Location 2	Remainder
3	11	2
10	14	
4	12	4
11	15	
5	9	9
8	15	
4	6	13
4	8	
4	9	
4	10	
4	13	
8	11	
1	15	19
5	12	
9	10	
9	13	
1	11	23
4	5	
1	8	26
9	12	
3	7	38
10	13	
2	3	39
2	4	
2	6	
2	8	

2	9	
2	10	
2	13	
4	14	41
7	8	
2	12	46
5	6	
6	8	52
6	9	
6	9	
6	10	
6	10	
6	13	
2	5	61
6	12	
1	5	65
4	11	65
8	9	
8	10	
8	10	
8	13	
8	13	
12	15	
4	15	69
11	12	
1	9	72
8	12	
3	4	78
3	6	
3	8	
3	9	
3	10	
3	13	
7	10	
1	12	82
5	15	
1	4	86
5	11	

5	8	91
9	15	
4	7	101
8	14	
3	14	106
10	11	
1	6	111
2	15	
1	2	124
6	15	

Table-2. A look up table for the remainders and corresponding bit error locations for a double bit error correction using CRC.

Click Here to Download the Matlab code for the Graph shown above

From the table above, It can be inferred that one remainder can correspond to a different combination of error location doublets. As the information sequence length increases, the number of such non-unique remainders increases exponentially and the thus the memory requirement for using this scheme for error correction increases significantly. When a non-unique remainder is generated at the receiver, the lookup table is referred and one of the possibilities is randomly chosen and the bits at those locations are flipped. The CRC checksum is again computed to check if the error has been corrected or not. If the reminder computed after making the desired changes is computed to be 0, then the received codeword is supposed to be corrected of errors, else the next possibility of the bit error location is taken and remainder is recomputed. This process is iterated until the correct sequence is obtained. It is evident that, this technique of double bit error correction requires a lot of resources in terms of memory and computation time. As the error correction demand increases, the requirement for such resources also increases drastically. We plot a graph which compares the order of the CRC polynomial required to correct the different number of errors and the maximum length of information sequence considering a fixed amount of resources are available in terms of memory and computation time.

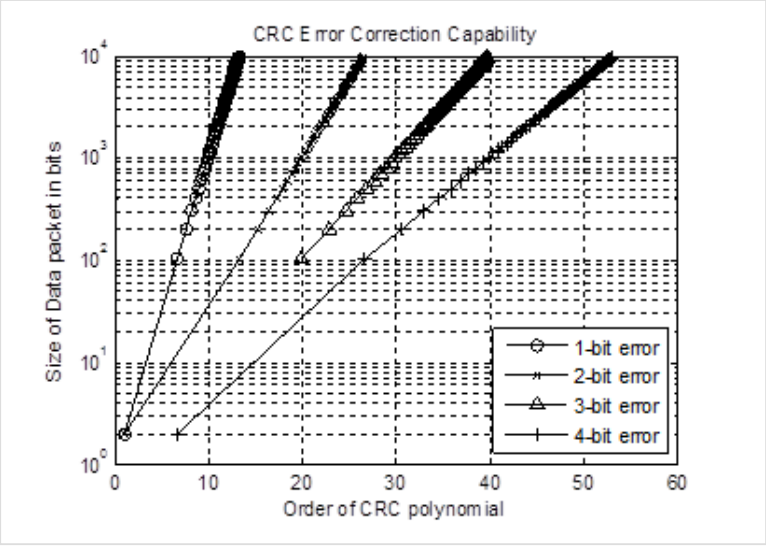


Figure 1. Length of the information sequence versus the required CRC generator polynomial order for different number of bits to be corrected.

Click Here to Download the Matlab code for the Graph shown above

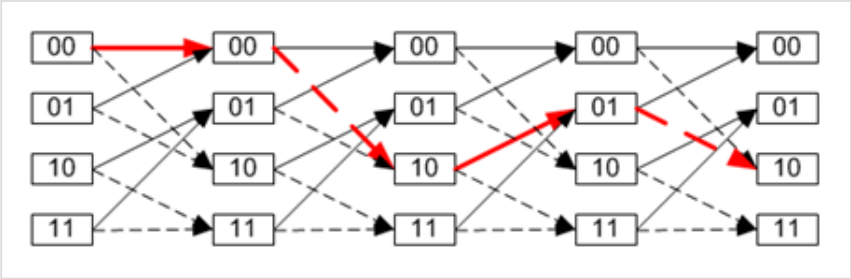
It is clear from the plot shown above that as the error correction demand increases; the number of information bits that can be transmitted goes down linearly on a logarithmic scale. This graph is plotted considering that sufficient amount of storage is available for storing the error look up tables. Consequently, if there is a set constraint of the availability of resources, this capability of this scheme would go down accordingly.

III. Trellis Coded Error Correction Assisted By Crc

The paper in [13] introduces a novel error correction technique using CRC for a trellis coded system with bit-stuffing. They propose a special encoder and decoder to achieve high reliability in communication. The receiver can receive and correct the bits simultaneously based on a modified Viterbi algorithm which uses the concept of forming an extended trellis to enable CRC assistance and bit-stuffing.

A. TRELLIS CODING AND VITERBI ALGORITHM:

Trellis Code is basically a convolution code of rates (k, k+1) that is a part of Trellis Modulation of signals for highly efficient transmission of information for communications system. In a Trellis modulation the parity check is done on a per symbol basis rather than applying it for a bit stream and then modulating the bits. Trellis code for multilevel/phases are designed to achieve maximum free Euclidian distance rather than hamming distance (which measures number of positions at which the symbols in two equal length strings are different). This results in improved channel capacity and lower error rates than an equivalent uncoded system. We used the (2,1,4) Trellis Coding scheme in this project to realize the objective. An encoded message is decoded using the Viterbi algorithm which explores the most probable paths through the trellis. It uses some metrics to assign weights to different paths and every time a new path evolves, a corresponding metric related to the path is updated which depends on the probability of the current symbol being correct based on the previous state of the trellis. This process of assigning the metrics is a dynamic process and after all the symbols are received, the path corresponding to the highest metric is chosen to be the correct path. Here in the simulations carried out in Matlab, we choose distance as the weight assigning metric. For example if the received codeword is 11 and the next state is 10, then the the receiver assigns the path a weight of 1.



* Dotted lines represent a transition if input '0'
* Solid lines represent a transition if input is '1'

Figure 2. Example of a Trellis Tree

We know that the hardware implementation of CRC involves the use of shift registers and some XOR gates in between shift registers based on the design. The register contents decide the state of the CRC trellis and with each input, the trellis moves one step forward. Thus, the CRC code can also be represented in the form of a trellis tree. Figure 3 shows the representation of a trellis tree for the CRC with generator polynomial X2 + X+ 1.

B. TRANSMITTER AND RECEIVER DESIGN

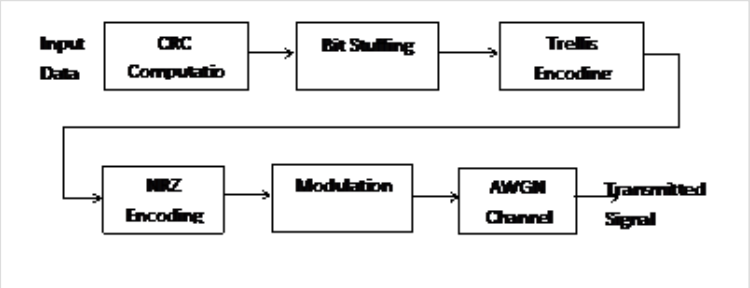


Figure 3. Transmitter Block Diagram

CRC - 16 Matlab Function (Example Run Command "CRC16 ([1 0 1 1 1])")

CRC - 2 Matlab Function (Example Run Command "CRC2 ([1 0 1 1 1])")

Bit Stuffing Matlab Function (Example Run Command "bitstuff ([1 0 1 1 1])")

Trellis Encoder Matlab Code (Example Run Command "trelliscoder ([1 0 1 1])")

NRZ Encoder Matlab Code

Example Run Command "NRZ_Encoder(Data,Bitrate,amplitude,'Polar');

Modulation and AWGN Channel Matlab Code

Sample output for BPSK Modulation ower AWGN channel

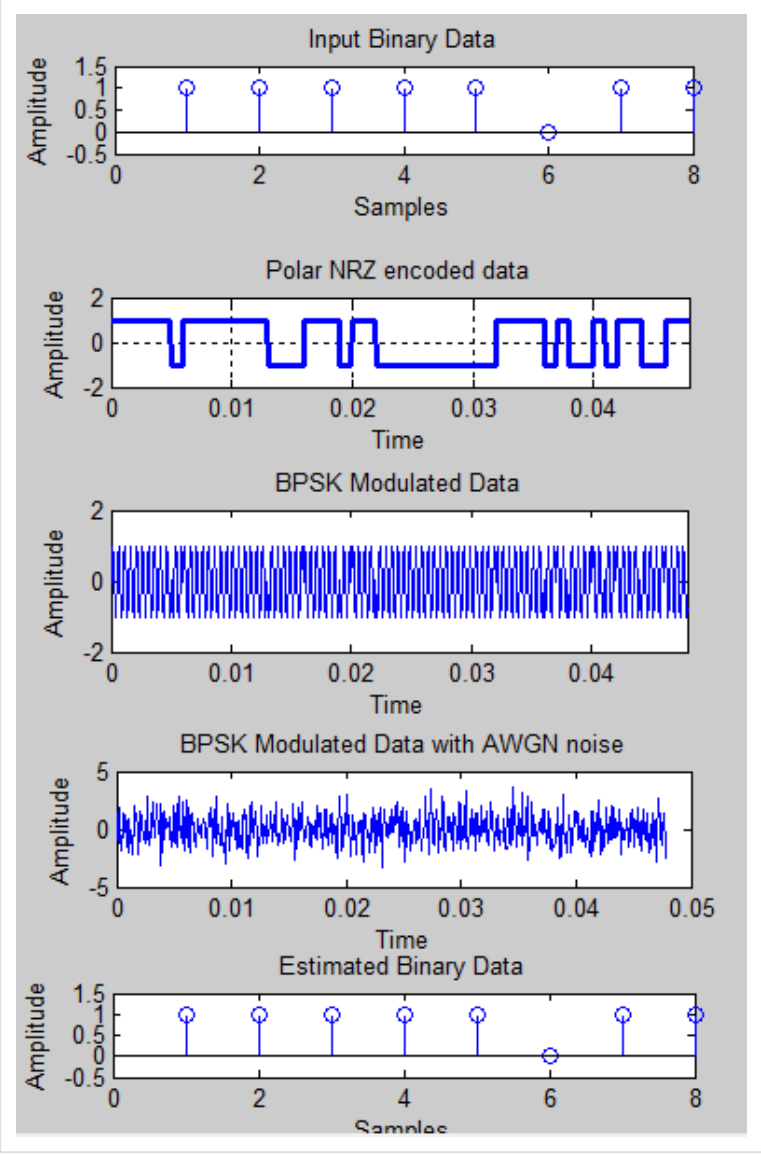


Fig. Demonstration of BPSK Modulation and AWGN channel with NRZ Encoding

Click Here to Download the Matlab code for the Graph shown above

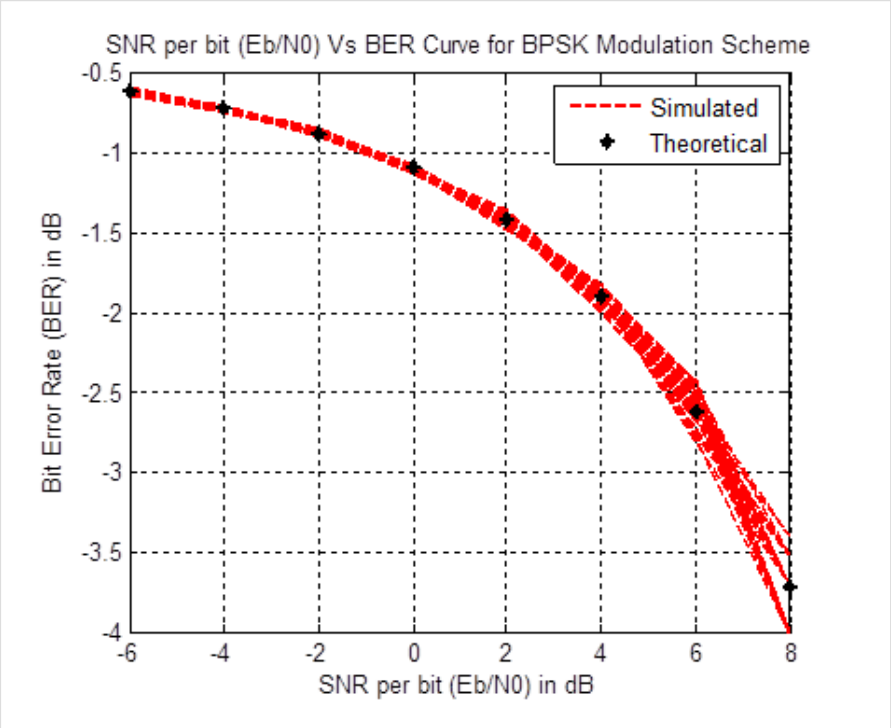


Fig. Demonstration of BPSK Modulation BER vs Eb/No

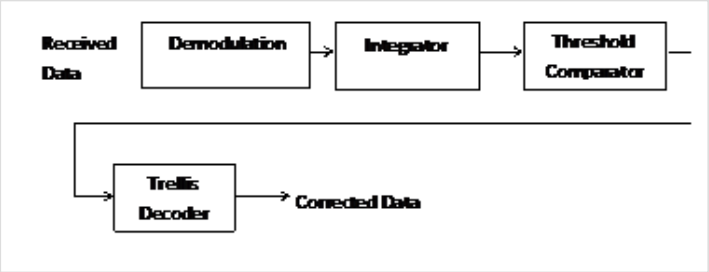
Click Here to Download the Matlab code for the Graph shown above

All the individual blocks shown above are implemented in Matlab with the following test parameters for carrying out simulations.

Test Conditions:	
CRC Generator Polynomial	X16 + X12 + X5 + 1
Bit-Stuffing	0 after every 5 consecutive 1's
Trellis Encoding	(2,1,4) Convolutional Encoder
Bit coding	NRZ Polar
Modulation Technique	BPSK
Bit rate	1000bps
Amplitude	1
Channel	AWGN channel with varying noise variance
Packet Size	100 bits

The Receiver is designed in a way so that it can implement the Extended trellis presented in the original paper, composed of original trellis as well as a CRC trellis. As stated earlier the CRC scheme can also be represented using the Trellis and interestingly can be decoded using the Viterbi algorithm for demodulation of a Trellis. We utilize this aspect to generate the code-word from the received sequence. The proposed algorithm is based on a constrained maximum likelihood estimation minimizing the square Euclidean distance defined by the following constraint of the CRC satisfaction. In order to satisfy this constraint, the proposed receiver is based on a trellis composed of extended states formed by a CRC state and a TC state. The trellis is designed so that all paths ending with a final state give a message whose joint CRC is zero. Moreover, the stuffed bits are taken into account by considering special transitions in the extended trellis. The intermediate CRC calculations can be seen as states in a trellis. These CRC states are interconnected by transitions linking a CRC state to a new CRC state obtained after updating the previous CRC

with one bit and the method shown in the Trellis Diagram above. The trellis proposed in the algorithm consists of extended states composed of a CRC state and a TC state. Every time the Trellis for the encounters a sequence of five 1's while dynamically decoding, it stalls the CRC trellis which runs in parallel so as to account for the stuffed bit hence taking care that the stuffed bit doesn't affect the CRC computation. This is like a double shielded error coding scheme where both CRC and Convolution code [6] help simultaneously in Error Correction.



The Receiver is designed in a way so that it can implement the extended trellis presented in the original paper [11], composed of original trellis as well as a CRC trellis. As stated earlier the CRC scheme can also be represented using the Trellis and interestingly can be decoded using the Viterbi algorithm for demodulation of a Trellis. We utilize this aspect to generate the code-word from the received sequence. The proposed algorithm is based on a constrained maximum likelihood estimation minimizing the square Euclidean distance defined by the following constraint of the CRC satisfaction. In order to satisfy this constraint, the proposed receiver is based on a trellis composed of extended states formed by a CRC state and a TC state. The trellis is designed so that all paths ending with a final state give a message whose joint CRC is zero. Moreover, the stuffed bits are taken into account by considering special transitions in the extended trellis. The intermediate CRC calculations can be seen as states in a trellis. These CRC states are interconnected by transitions linking a CRC state to a new CRC state obtained after updating the previous CRC with one bit and the method shown in the Trellis Diagram above. The trellis proposed in the algorithm consists of extended states composed of a CRC state and a TC state. Every time the Trellis for the encounters a sequence of five 1's while dynamically decoding, it stalls the CRC trellis which runs in parallel so as to account for the stuffed bit hence taking care that the stuffed bit doesn't affect the CRC computation. This is like a double shielded error coding scheme where both CRC and Convolution code [6] help simultaneously in Error Correction.

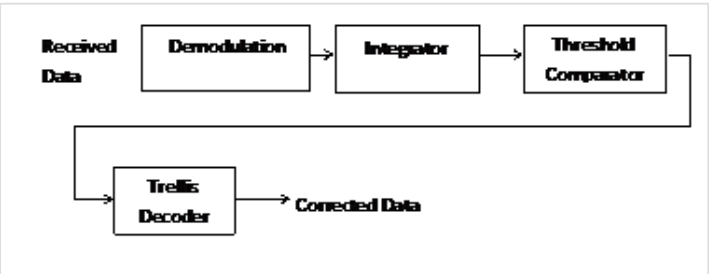


Figure 4.Receiver Block Diagram

Thus, the trellis that has been proposed in [11] will have two states one corresponding to the trellis coding and the other corresponding to the CRC. If a trellis coded state α follows another trellis state β and a CRC state p follows another CRC state q then we can say that an extended state (α, p) follows another extended state (β, q). For using the Viterbi Algorithm with Bit Stuffing, the algorithm was modified as suggested in [11].

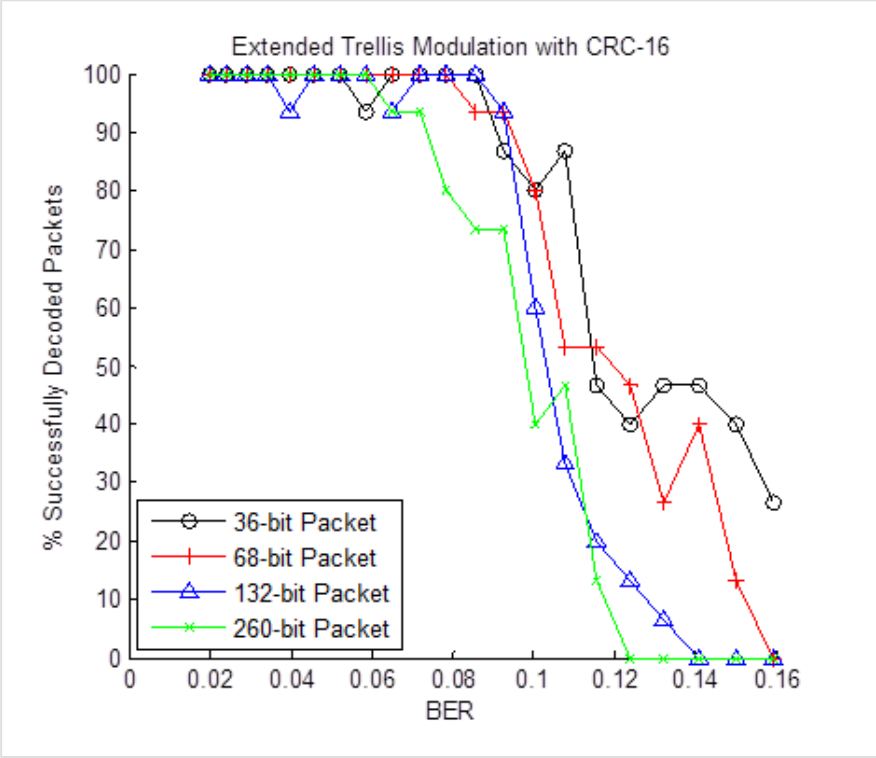
CRC State	Trellis State	Extended State
$A \xrightarrow{0} B$	$\alpha \xrightarrow{0} \beta$	$(A;\alpha) \xrightarrow{0} (B;\beta)$

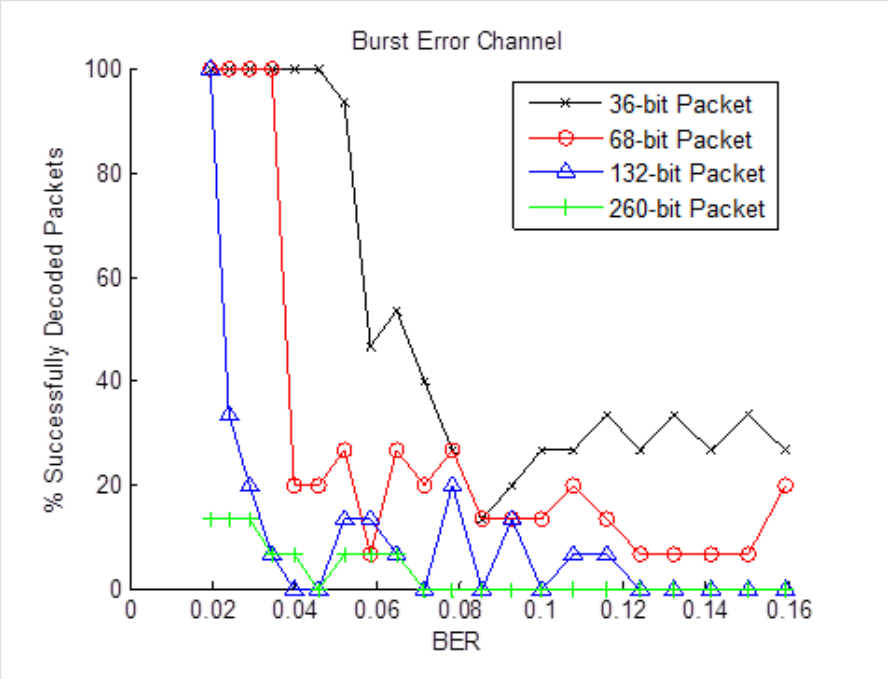
$$A \xrightarrow{1} C \qquad \alpha \xrightarrow{1} \gamma \qquad (A;\alpha) \xrightarrow{1} (C;\gamma)$$

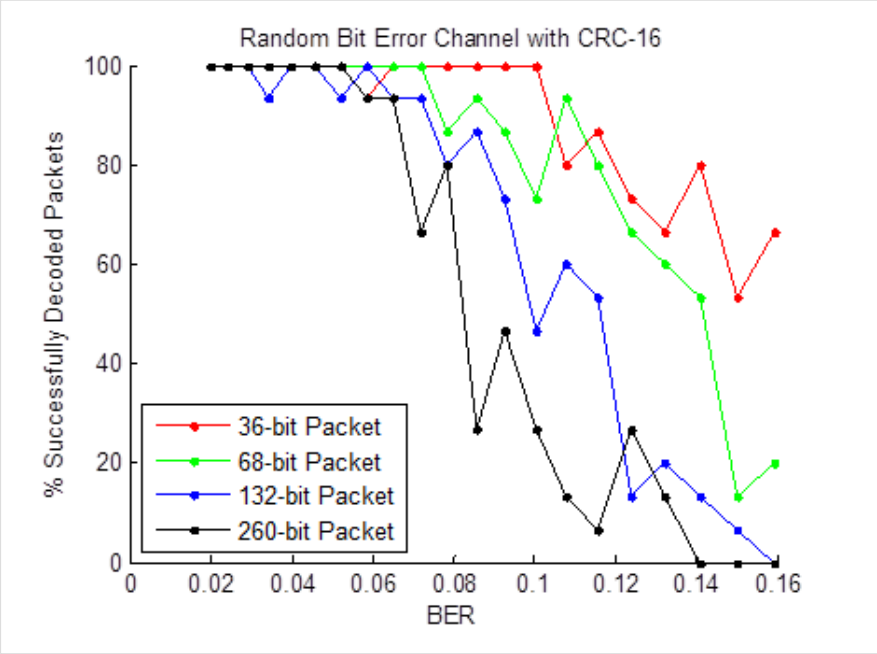
$$A \xrightarrow{SB} B \qquad \alpha \xrightarrow{SB} \beta \qquad (A;\alpha) \xrightarrow{SB} (A;\beta)$$

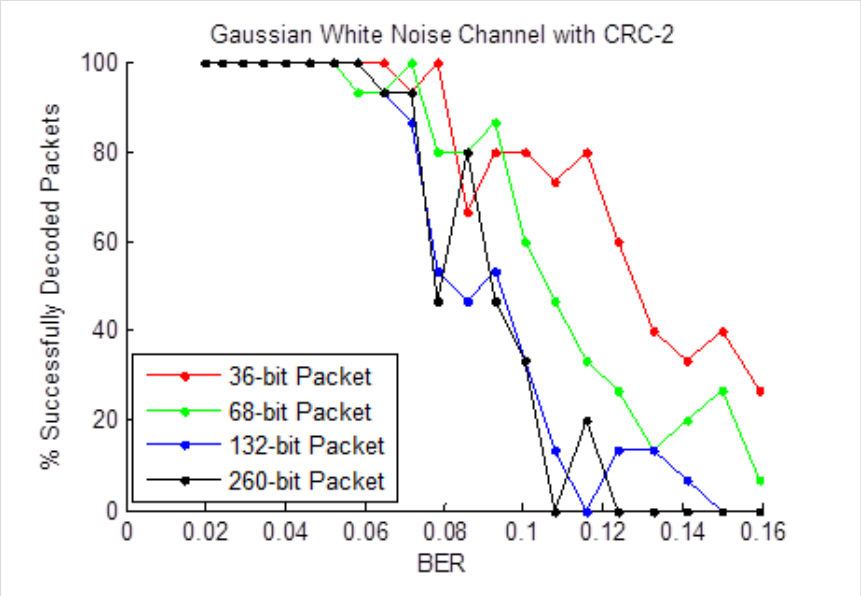
The Viterbi algorithm for the decoder is modified to account for the bit stuffing that is being done at the transmitter. There are special transitions in the trellis that when a stuffed bit is encountered during the decoding process. It is evident from the transition table shown above that the CRC trellis stalls for once cycle just when a stuffed bit is encountered while traversing the extended trellis tree.

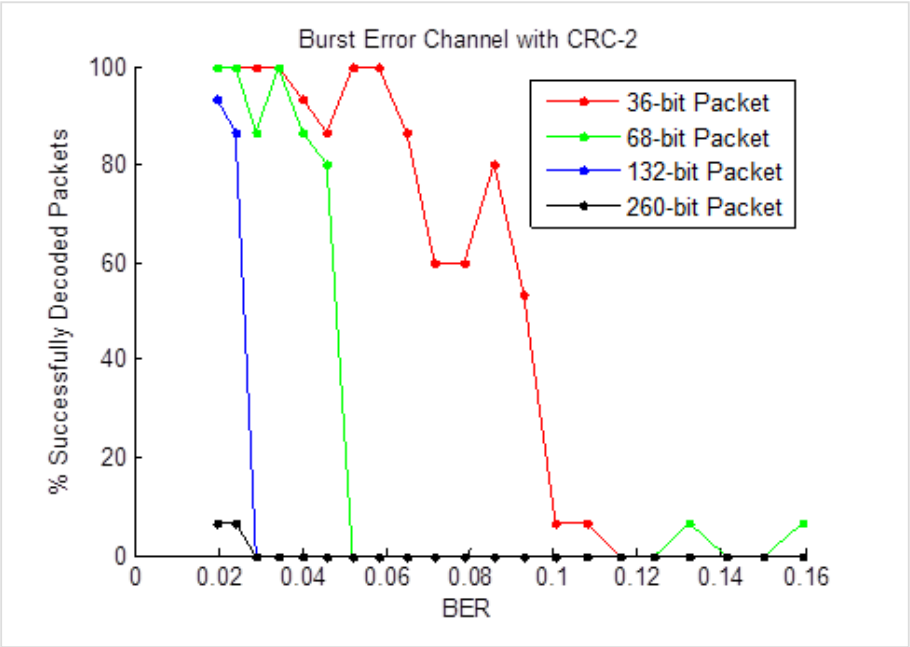
The main job of the algorithm that was implemented in the receiver is to decode the incoming packets that have been encoded using the Extended Trellis, using the Viterbi algorithm [8]. In order to complete this task, the receiver must explore every possible trellis path, while applying a pre-determined metric to the paths that are most likely to be correct. The most significant portion of the algorithm concerns itself with the task of creating these path transitions. Each path through the trellis is maintained in a trellis matrix, where the rows of this matrix are the various current possible paths of the trellis. It should be noted that since the extended trellis is two dimensional, two separate trellis matrices are created, one for the original TC trellis, and one generated for the CRC trellis. To overcome the presence of bit stuffing, a state variable associated with each path of the CRC trellis is updated during every path transition, and when this state variable for a given path reaches a threshold (signifying the presence of a stuffed bit), the CRC matrix maintains its current state (stalls), rather than moving to the next, as the main TC trellis will do. Additionally, due to the nature of the way a trellis is constructed, after each transition, many of the existing paths will converge. Because of this, after each state transition, the metric for each current path is evaluated, and multiple paths that converge are reduced by eliminating the paths with the lower metric. After both trellises are constructed, along with the corresponding metric values for each path, the CRC trellis is specifically examined. Since it is known that correctly received packets will have a zero remainder when divided by the CRC generator polynomial, resulting in a zero ending CRC trellis state, paths in the CRC trellis (and corresponding TC trellis) are removed from the list of possible correct packets. After this last step is done, the receiver chooses the packet from the remaining trellis paths corresponding to the path with the highest metric as the correctly received packet. In the event that channel noise is large enough to prevent any of the RC paths from converging to a zero state, the influence of the CRC trellis is removed from the final decision. It should be noted that although the tests that generated these simulation results were ran somewhat exhaustively, in order to produce smoother plots with no spikes the simulations would need to be run for a very long time. For each data point shown, around 50 tests were run to determine the corresponding percentage. Alternatively, a function could be used to produce a best fit plot for the results presented in Fig.5, however what is shown is the actual simulation results.


Extended Trellis Modulation with CRC-16
% Successfully Decoded Packets
BER
36-bit Packet
68-bit Packet
132-bit Packet
260-bit Packet
0
10
20
30
40
50
60
70
80
90
100
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16


Burst Error Channel
% Successfully Decoded Packets
BER
36-bit Packet
68-bit Packet
132-bit Packet
260-bit Packet
0
20
40
60
80
100
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16


Random Bit Error Channel with CRC-16
% Successfully Decoded Packets
BER
36-bit Packet
68-bit Packet
132-bit Packet
260-bit Packet
0
20
40
60
80
100
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16


Gaussian White Noise Channel with CRC-2
% Successfully Decoded Packets
BER
36-bit Packet
68-bit Packet
132-bit Packet
260-bit Packet
0
20
40
60
80
100
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16



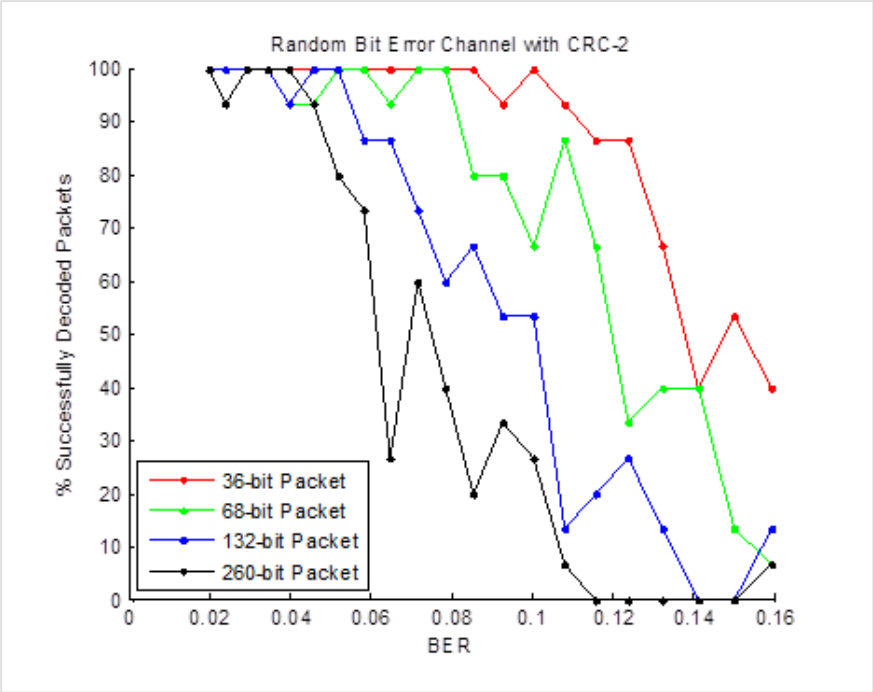


Figure 5.1 - 5.6. Percentage of Successfully decoded packets using different packet sizes for different Bit Error rates. and Different Error Channels

is evident from Fig 5. that the extended trellis coding scheme based on the modified Viterbi Algorithm starts failing for BERs close to 0.1. For smaller packet sizes the probability of successfully receiving a packet is more which intuitively also makes sense because one faulty bit in a packet can result in discarding the packet. Since the probability of having a faulty bit in a longer packet is more, the PER decreases drastically with the increase in packet size.

The data messages have fixed length of 100 bits over which CRC is computed and bit stuffed and trellis encoded followed by Polar NRZ encoding and BPSK modulation. For our simulations, we transmitted 100 packets of length 100 bits each over an AWGN channel varying the noise variance and studies the number of successfully received packets for different values of noise variance in an AWGN system. The AWGN channel in Matlab was modeled by using the randn function which generates noise with zero mean and a unit variance. In an AWGN system the relation between noise variance and power spectral density is given by

$$\sigma^2 = \frac{No}{2}$$

and if the symbol energy is normalized to 1, Eb/No is given as

$$\frac{E_b}{N_o} = \frac{E_s}{R_m R_c N_o}$$

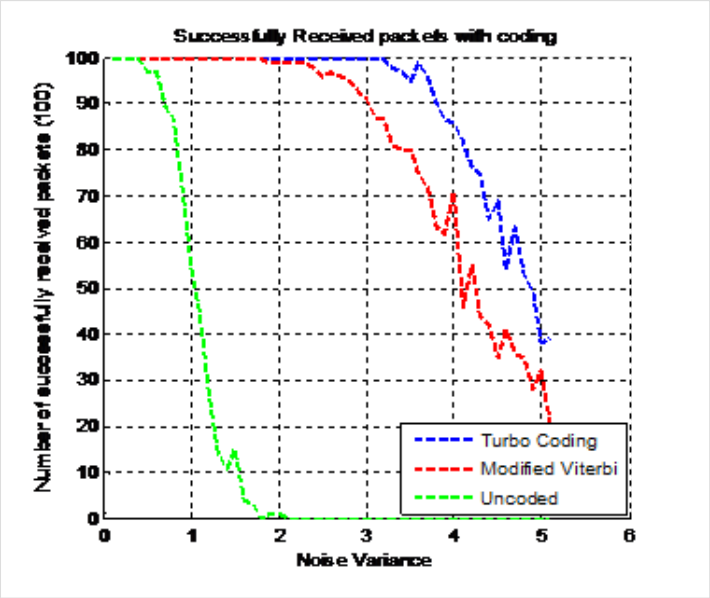


Figure 6. Novel Receiver based on modified Viterbi algorithm compared with a conventional BPSK Receiver in successfully receiver packets.

It can be inferred from the graph shown in Figure 6, that the packet error rate for the proposed receiver remains 0 for a noise variance of 2 and starts degrading henceforth. For higher values of noise variance, the amount of tolerance is higher

We also plotted Eb/No versus Packet Error Rate (PER) for different values of noise variance in the AWGN channel for standard BPSK receiver, Turbo coded data and the proposed receiver design.

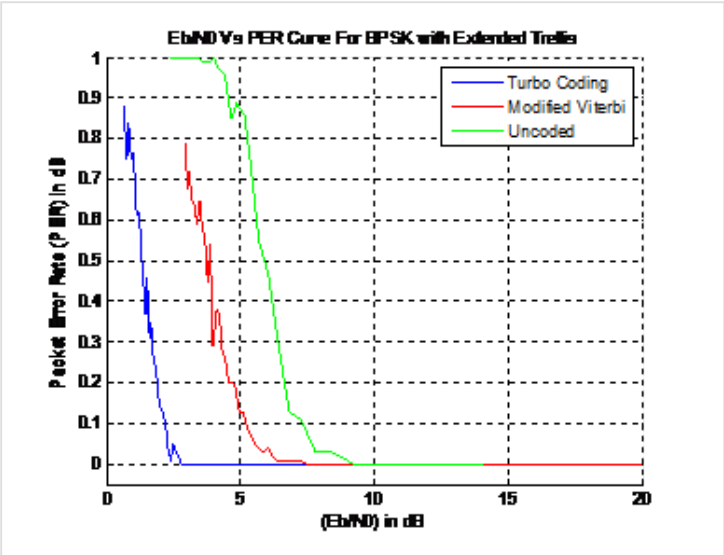


Figure 7. Novel Receiver based on modified Viterbi algorithm compared with a conventional BPSK Receiver in Packet Error rate versus Eb/No in dB

Again, these simulations were carried out for 100 packets and packet error rate was determined based on the number of correctly received packets in every 100 packets transmitted. The coding gain here varies from 2 dB to 4 dB based on the Eb/No value.

Although this scheme looks good for AWGN channels, it does not perform as good for a burst error channel model. We simulated the performance of the proposed receiver for a bursty channel using different sized data packets and found out that the percentage of successfully received packets falls drastically. This is because, while decoding using the Viterbi algorithm, when a burst of

error occurs, the trellis path digresses from the original path that it should have followed otherwise. Once the shift from the path is drastic, the weights associated with the paths tend to become smaller and the ultimate chosen path will have a very small weight as compared to the true path.

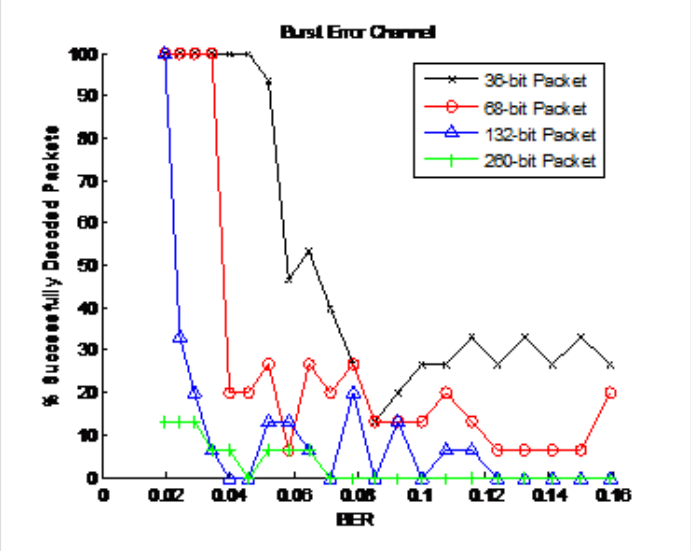


Figure 8. Performance of the proposed receiver for a bursty error channel based on BER vs percentage of successfully decoded packets.

The Viterbi algorithm is vulnerable to burst errors and it is very evident from the plot shown in Figure. 8 that the proposed receiver does not fit well in bursty channel environment and the performance degrades drastically and it can be a real problem since most of the channels introduce burst errors. Unlike the simulations for AWGN channel in which the scheme started failing at BER = 0.1, this coding scheme starts failing for BER as low as 0.03.

Thus if a communication medium is prone to error affecting contiguous blocks of bits or if burst of noise is introduced into a CD/DVD or in the hard drive where the data is written into contiguous memory cells. Different algorithms like the Maximum likelihood detection are used to increase the storage density of a disk. Error Control techniques can also be used to improve the performance of such algorithms. One such error control technique which is used for burst error correction is the Reed Solomon code discussed in the next section.

IV. Reed Solomon Codes

A. BURST ERROR CORRECTION CAPABILITY

Reed Solomon codes are a set of linear block codes and are a subset of BCH codes also called the non-binary BCH code. The Reed Solomon code is specified as RS (n, k). Such a code would take in would take k data symbols and append n-k parity bits to it to make it an n symbol codeword. The capability of error correction of the Reed Solomon code is $t \leq (n-k)/2$. Thus, a (255, 235) RS code can correct 10 errors. In our implementation of the RS code to rectify burst errors, we use the RS code with an interleaver which aids the spreading of burst errors. A de-interleaver is used at the receiver to reorder the bits to their original position. If the length of the burst b is less than the error correction capability of the RS code, then there is no interleaving required at the encoder but if the burst length is longer than the length of the maximum error correction capability of the code, interleaving can be employed. Here we try to spread the burst error sequence over different code blocks so that each block has no more than t errors.

Codeword 1	a0 a1 a2 a3 … a234	a235 … a254
Codeword 2	b0 b1 b2 b3 … b234	b235 … b254
Codeword 3	c0 c1 c2 c3 … c234	c235 … c254
Codeword 4	d0 d1 d2 d3 … d234	d235 … d254

Codeword 5	e0 e1 e2 e3 … e234	e235 … e254

Table-3. (255, 235) RS codeword and Parity bits

Here the idea is to write the data on a row by row basis and to read it out on a column by column basis.

B. SIMULATION RESULTS

For the sake of simulations we use a (255, 235) RS coding system. This code has the capability of correcting up to 10 symbol errors. Suppose that the channel that we are going to transmit the data through can bring about a maximum of 25 symbol errors. Thus the required interleaver depth is (25/10) = 2.5 or 3. The block interleaver to be used in the design would have 3 rows and 255 columns. The following are the results of the simulation that we carried out.

Input Data: = 'I AM A STUDENT AT THE UNIVERSITY OF FLORIDA. THIS IS A PART OF MY WIRELESS COMMUNICATION FINAL PROJECT. I AM TRYING TO STUDY THE BURST ERROR CORRECTION CAPABILITY OF THE REED SOLOMON CODE. LET US SEE IF INTERLEAVING HELPS IN CORECTING BURST ERRORS OR NOT!'
Interleaved Data = 'III AAAMMM AAA SSS********** AAATTT TTTHHHEEE UUUNNNIIIVVVEEERRRSSSIIITTTYYY OOOFFF FFFLLLOOORRRIIIDDDAAA... TTTHHHIIISSS IIISSS AAA PPPAAARRRTTT OOOFFF MMMYYY WWWIIIRRREEELLLEEESSSSSS CCCOOOMMMMMMUUUNNNIIICCCAAATTTIIIOOONNN FFFIIINNNAAALLL PPPRRROOOJJJEEECCCTTT... III AAAMMM TTTRRRYYYIIINNNGGG TTTOOO SSSTTTUUUDDDYYY TTTHHHEEE BBBUUURRRSSSTTT EEERRRRRROOORRR CCCOOORRRRRREEECCCTTTIIIOOONNN CCCAAAPPPAAABBBIIILLLIIITTTYYY OOOFFF TTTHHHEEE RRREEEEEDDD SSSOOOLLLOOOMMMOOONNN CCCOOODDDEEE... LLLEEETTT UUUSSS SSSEEEEEE IIIFFF IIINNNTTTEEERRRLLLEEEAAAVVVIIINNNGGG HHHEEELLLPPPSSS IIINNN CCCOOORRREEECCCTTTIIINNNGGG BBBUUURRRSSSTTT EEERRRRRROOORRRSSS OOORRR NNNOOOTTT!!!
Received Data = "I AM A S*******AT THE UNIVERSITY OF FLORIDA. THIS IS A PART OF MY WIRELESS COMMUNICATION FINAL PROJECT. I AM TRYING TO STUDY THE BURST ERROR CORRECTION CAPABILITY OF THE REED SOLOMON CODE. LET US SEE IF INTERLEAVING HELPS IN CORECTING BURST ERRORS OR NOT!"

Table-4. (255, 235) RS codeword with Interleaving

It is evident from the simulations that the channel flips 20 consecutive bits in a burst, which is beyond the error correction capability of Reed Solomon codes, but with the help of an interleaver and a de-interleaver at the RS encoder and decoder respectively, this can be reduced to 7 symbols which can be tackled by the (255, 235) Reed Solomon code.

Thus, we see that the RS code with the interleaver design is good for correcting burst errors upto a limit which is way more than the capacity of the Reed Solomon code. Results from some of the work in [16] shows that we can contatenate two codes, one which is good for a random error channel and the other one which is good for correcting burst errors like the RS code illustrated above.

V. Turbo and LDPC coodes

Turbo and LDPC codes are the codes which have a coding gain which reaches closest to the Shannon Limit. In this section we briefly compare Turbo codes with LDPC codes in terms of performance and coding gain. These codes follow the iterative decoding algorithm.

A. TURBO CODES

Turbo codes are one of the major developments in the field of error control coding because the performance of turbo codes is very close to the limits of wireless communication also known as the Shannon limit. The two key components of these coding schemes are concatenated encoding and iterative decoding. The encoder comprises of two parallel convolutional encoders one of which has an interleaver on its path. The input to the channel is the data and the parity bits. Each of the Turbo decoders is responsible for soft input/output decoding as shown in Figure 9.

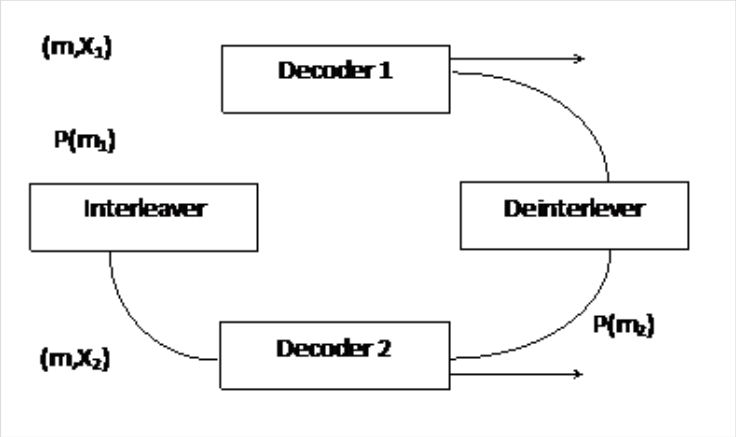


Figure. 9 Turbo decoder

In Fig. 9, a soft decision P(m1) is generated depending upon the received codeword (m, X1). This probability is calculated using MAP or SOVA algorithms. Once such information is generated, it is then passed to the second decoder, which will generate another measure P(m2) based on the received codeword(m, X2). This information is then fed back to the first decoder which in turn recomputes its measure hence forming a simple yet an effective feedback control system. Thus there are several iterations that may be required to converge onto a specific measure, however there would be some cases when the convergence just does not happen.

B. LDPC CODES

LDPC codes were invented in 1961 and these codes gained importance only after Turbo codes were introduced. Turbo codes, unlike LDPC codes have a high decoding and a low encoding complexity. LDPC codes are a class of linear block codes that approach the theoretical limit of error control coding. LDPC codes can be characterized by the sparseness of ones in the H matrix which allows for a large minimum distance of the code hence providing better performance. Simulation results show a steep drop in BER as SNR improves. LDPC codes can be broadly categorized into regular and irregular codes. LDPC codes can be generated using Density Evolution and Finite Geometry.

C. SIMULATION MODEL

For the purpose of simulations, we have simulated the following model in Matlab and quantified the performance.

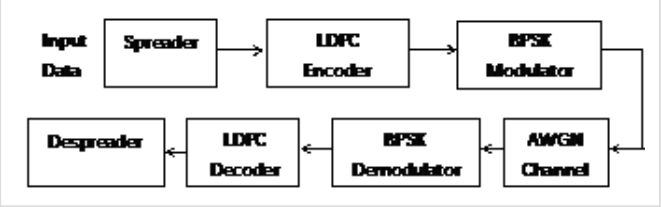


Figure. 10 LDPC Encoder and Decoder communication model

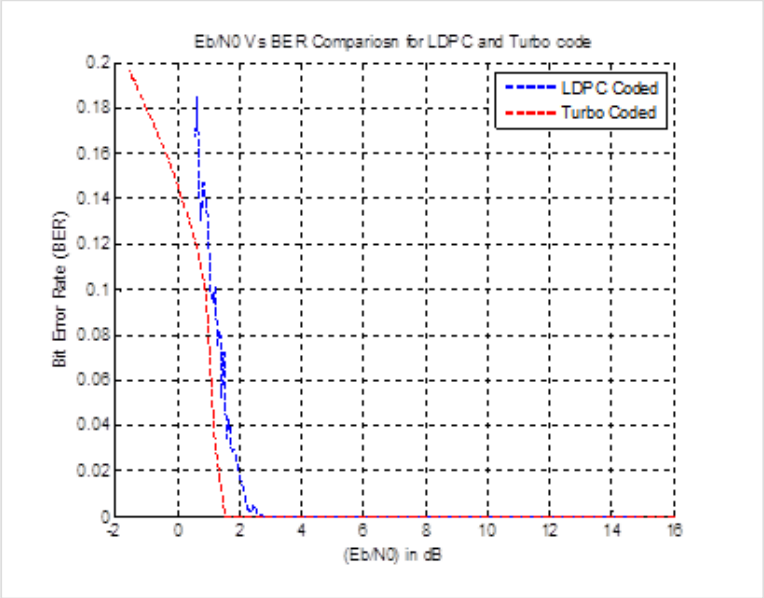


Figure. 11 Comparing the Performance of Turbo Codes and LDPC codes on a linear BER scale.

*Please note that the BER axis here is in the linear scale unlike the conventional logarithmic plots that we draw for comparing different schemes.

To achieve better and more appropriate results, we simulate for different values of noise variance which means that we change the ratio of the signal to noise energy and carry out a lot of iterations by transmitting a number of data packets. For these simulations we transmit 100 data packets for each value of noise variance in the AWGN channel and sweep over a range of noise variance. It can be observed from the graph that the coding gain difference between LDPC and Turbo codes is about 0.5dB. It can be seen from Fig 11. that turbo codes outperform LDPC codes in this specific scenario when the chosen size of data is 100 bits. It was learnt from a discussion with Dr. Wu that, for data lengths less than 800 bits, Turbo codes perform better than LDPC codes and for data lengths larger than 8000 bits, the turbo codes would be outperformed by LDPC codes. However, it is difficult to predict the performance of these codes when the frame size is between 800bits to 8000 bits.

Comparison	Gain/dB
Between uncoded and Modified receiver based on Trellis Coding and modified Viterbi based decoding. (BPSK + AWGN)	2dB
Between uncoded and Turbo coded stream. (BPSK + AWGN)	4dB
Between Modified receiver based on Trellis Coding and modified Viterbi Algorithm based decoding and Turbo coded stream. (BPSK + AWGN)	2dB
Between Turbo Coded and LDPC coded data stream (BPSK + AWGN)	0.5dB

Table-5. Coding Gain comparison chart

Turbo Coded Communication System Matlab Code

LDPC Coded Communication System Matlab Code

VI. Conclusion

In this paper we evaluate the performance of some error correction techniques in terms if Bit Error Rate and Packet Error Rate for different ratios of signal to noise power considering data streams which are source encoded, BPSK modulated and transmitted over and AWGN channel. It can be concluded that Turbo and LDPC codes provide a considerably higher coding gain as compared to the other coding techniques. Also, it can be deduced from the simulation results the the presence of interleaver can help improve the bit error rate for a bursty channel. Varying the length of the Interleaver can give further improvement, obviously with the tradeoff for redundancy and hardware because in Wireless Communication, performance improvement rarely comes for free. We wish to emphasize on the fact that the results shown in the paper do not give an exact indication of the performance of different codes, but are close estimations since these results are for specific cases. It would be imprudent to conclude which coding technique is the best or is the worst because some techniques would be good for a specific scenario while others would underperform in a similar case. So depending on the frame length, number of iterations, error channel, type of modulation and redundancy addition, a coding scheme would tradeoff performance with other factors.

VII. Acknowledgement

The author would like to acknowledge the help of the members of Electrical and Computer Engineering Department and Dr. Dapeng Wu for the useful learning during the inclass lectures and one on one discussions on the issues faced.

REFERENCES:

1. Jatinder Singh and Jaget Singh, “A Comparative study of Error Detection And Correction Coding Techniques” in *2012 Second International Conference on Advanced Computing & Communication Technologies*.
2. Wikipedia, “Forward Error Correction Techniques”, Available: *http://en.wikipedia.org/wiki/Forward_error_correction*
3. Zheng Yuan and Xinchen Zhao “Introduction of Forward Error Correction and its Application” *International School, Beijing University of Posts and Telecommunications*.
4. Kamran Azadet and Mark Yu “Forward Error Correction (FEC) techniques for optical communications” *IEEE 802.3 High-Speed Study Group Plenary meeting, Montreal July 1999*.
5. Ufuk DEMIR and Ozlem AKTA “Raptor versus Reed Solomon Forward Error Correction Codes” in *Proceedings of the Seventh IEEE International Symposium on Computer Networks (ISCN'06)*.
6. *Essentials of Error Control Coding*, John Wiley and Sons Ltd., Jorge Castiñeira Moreira and Patrick Guy Farrell, 2006.
7. Aoife Moloney, “Channel Coding Lectures Notes”. School of Electronics and Communications Dublin Institute of Technology.
8. T. S. Rappaport, “Wireless Communications: Principles and Practice”, Second Edition, 2002.
9. J. G. Proakis, “Digital Communications”, Fourth Edition, 2001.
10. Yanbin Zhang, Qi Yuan “A Multiple Bits Error Correction Method Based on Cyclic Redundancy Check Codes”, *ICSP2008 Proceedings*.
11. Babaie, S, Zadeh, A.K., Es-hagi, S.H. ; Navimipour, N.J. “Double Bits Error Correction Using CRC Method” *Semantics, Knowledge and Grid, 2009. SKG 2009. Fifth International Conference*
12. B. McDaniel, “An algorithm for error correcting cyclic redundance ecks,” C/C++ Users Journal, p. 6, 2003.
13. Raoul Pr´evost, Martial Coulon1, David Bonacci, Julia LeMaitre-Pierre Millerioux and Yves Tourneret “CRC-Assisted Error Correction In a Trellis Coded System”, *2011 IEEE Statistical Signal Processing Workshop (SSP)*.
14. .Wicker, "Error Control Systems for Digital Communication and Storage", Prentice-Hall 1995
15. Wilson, "Digital Modulation and Coding", Prentice-Hall 1996.
16. Sanjeev Kumar and Ragini Gupta “Performance Comparison of Different Forward Error Correction Coding Techniques for Wireless Communication Systems”.
17. Y. Zhang and Q. Yuan, “A multiple bits error correction method based on cyclic redundancy check codes,” ICSP Signal Processing, no. 9, pp.

1808-1810, 2008

Click Here to Download the Project report

Click Here to Download the Project Proposal

Author Contact:

Blockquote

> "The great successful men of the world have used their imagination?they think ahead and create their mental picture in all its details, filling in here, adding a little there, altering this a bit and that a bit, but steadily building."
> --Robert Collier