ECE300 Design of a Communication Link – Part II

William Chen
Cooper Union, New York

Jongoh (Andy) Jeong
Cooper Union, New York

Yilun (Eric) Jiang
Cooper Union, New York

ECE300 Fall 2018 / 19 December 2018

***Abstract*—This project aims to explore different methods of error control coding schemes, i.e. block codes, bit-level and symbol-level codes, convolutional codes. These coding schemes applied to the provided channel with moderate intersymbol interference are experimented and compared in terms of the BER performance.**

Keywords—Error control coding, intersymbol interference, BER

I. INTRODUCTION

In Part I of the project, the BER performance through an additive Gaussian white noise channel with moderate ISI was explored for various modulation orders, i.e. M = 2, 4, 16, with and without equalizers. Although equalizers such as linear, decision-feedback and maximum likelihood sequence estimator equalize, or correct, the modulated symbols fairly well to the extent that BER reaches the order of 10e-5 at high SNR (dB), i.e. 12 dB. However, error correction codes result in better performance in BER. The uses of a single-level and two-level error control coding schemes on BPSK are explored in this part of the project.

II. BACKGROUND

Instead of simply meeting the requirement of BER of 10e-6 at SNR of 12 dB, different error correction codes are explored to compare against one another. Additionally, two-level codes are experimented to see any improved results.

A. Block Code

Block code schemes make use of first taking the input data bit stream into blocks of k-bit streams, and mapping each k-bit block into n-bit block. In the encoding process, it is assumed that n > k, (n - k) check bits are added to each k-bit block. The code rate of such code is given by the ratio k / n, and the ratio (n – k) / k is called the redundancy of the code. For a given set of (n, k) code, the minimum Hamming distance is given by $d_{min} = n - k$, and it can correct up to t bits of errors for it satisfies $d_{min} = 2t + 1$. Block codes in general uses a k-by-n generator matrix, G = [P I] or [I P] form, along with a (n-k)-by-n parity check matrix, H = [I –P'] or [-P' I] form, which give rise to the syndrome of the error pattern. In this simulations, the generic linear, Hamming, and cyclic block codes provided by MATLAB are employed.

For these block codes, parameters of interest are generator matrix and parity-check matrix. Each uses gen2par, gfprimdf, cyclpoly, respectively to produce generator matrix/polynomial, to be used in encode and decode processes. The generic linear block code requires a generator matrix; typically generator matrix is input by the user, but in this particular simulation, it was designed to derive from cyclic parity check matrix so as to match the size and share similar structure. A cyclic code is another linear block code with the property that cyclic shifts are also codewords. Hamming code, on the other hand, exploits an extra parity bit to allow detection of a single error.

B. Bit-level Code

Bose-Chaudhuri-Hocquenghem code, or BCH code, is another error correction code at bit-level. This code scheme is a generalization of Hamming code for multiple-bit error correction and include a large class of cyclic coes. There are binary and non-binary types (Reed-Solomon is a special type of nonbinary BCH). Often it is expressed as BCH(n,k,t) where the parameters are as follows.

For $m \geq 3$ and $t < 2^m - 1$,

$$\begin{aligned} n &= 2^m - 1, && n = \text{block length},\ m = \text{number of bits per symbol} \\ n - k &\leq mt, && \text{number of parity-check bits} \\ d_{min} &\geq 2t + 1, && \text{minimum distance} \end{aligned}$$

Equation 1. BCH code parameters

The Communications Toolbox has options for parameters in BCHEncoder(n,k,genpoly)—it is observed that the simple declaration with n and k give the best of all three methods through experiments. The parameters above provide several freedom of block lengths, code rates, and error correction capabilities. Some of the parameters compared in the simulations are (15, 11) and (255, 247) as in Table 1.

n, codeword	k, message	t, correctible bits
7	4	1
15	11	1
	7	2
	3	3
63	57	1
	51	2
	45	3
127	120	1
	113	2
	106	3
255	247	1
	239	2
	199	7

Table 1. Sample parameters for block, data, and error correcting capability lengths

C. Symbol-level Code

Reed-Solomon codes are a special type of non-binary BCH code. The parameters for RS-codes are defined by

For nonbinary BCH codes with q-ary alphabets,

$n = q^s - 1$, block length

(RS is nonbinary BCH with s = 1)

For t-error correction,

$n - k = 2t$, number of parity check bits needed

$d_{min} = 2t + 1$

Equation 2. RS Code parameters

RS codes tend to yield higher error correcting capability compared to other block codes, given fixed number of check bits. RS codes are effective in correcting burst errors and produce satisfactory performance for a large set of input symbols, i.e. Compact Disc (CD). For the simulations, the same n and k parameters as in BCH code are employed to compare BER performance at the same lengths.

D. *Convolutional encoder/Viterbi decoder*

Convolutional encoders make use of shift registers (flip flops) as memory. They depend not only on the input bits but also the internal states in producing encoded words. The parameters n and k are the same as other block codes; however there are other measures that define the code—constraint length K = m + 1, where m = number of registers in memory, and code rate = k / n. K is varied to control the redundancy of the code. As a representation of the states, trellis structure need to be defined in initialization, in which the internal state transitions differ according to the generator polynomials.

The trellis diagram represents all possible transition from the states from one time instant to the next. The decoder functions by calculating the error in bits with respect to its state transitions per time instant. An example of a similar trellis structure used is shown in Figure 1.

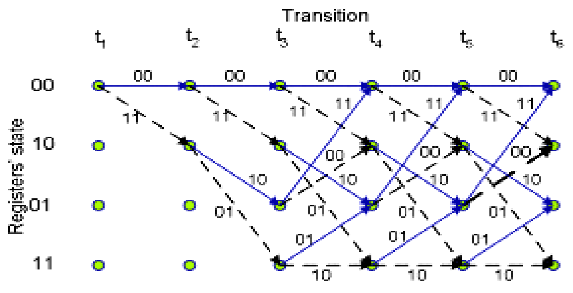

Fig 1. Example Trellis Diagram

In this simulation, two types of trellis structures are experimented – (1) K = 7, g1 = 171(8), g2 = 133(8). Generator polynomials, when not expressed in binary, are typically shown in octal form. In binary, g1 = [1111001] and g2 = [1011011]; and (2) K = 3, g1 = 7(8), g2 = 5(8), or g1 = [111], g2 = [101] in binary form. Because there are two generator polynomials and input of 1 bit, the code rate in both cases is ½ (1 bit in, 2 bits out).

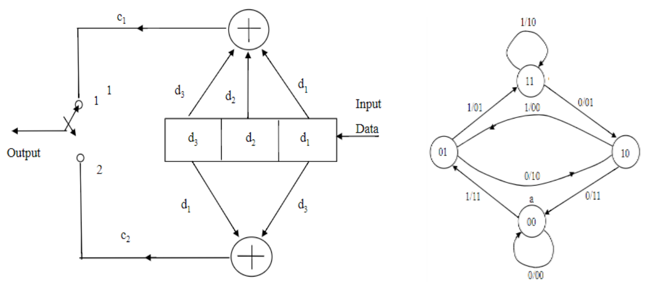

Fig 2. Convolutional Encoder State Diagrams for (2) case

Convolutional encoders are often coupled with Viterbi decoders. Viterbi decoding also utilizes the same trellis structure with trace-back depths specified. In addition, Viterbi algorithm can use ‘hard’ or ‘soft’ decision upon decoding, with truncated or continuous modes. From documentation of vitdec(), the operating modes are defined as shown in Table 2. The decoding process is performed either by following the path through the minimum Hamming distance (‘hard’) or minimizing the log likelihood of the symbols (probability) (‘soft’), and pruning unlikely paths based on the chosen criteria.

Mode	Meaning
continuous	The encoder starts at the all-zeros state. The decoder traces back from the state with the best metric (from trellis). A delay of traceback depth elapses before the first decoded symbol appears in the output. Appropriate when decoder is invoked repeatedly and want to preserve continuity between successive invocations.
truncated	The encoder starts at the all-zeros state. The decoder traces back from the state with the best metric. This mode incurs no delay. Appropriate when you cannot assume the encoder ended at the all-zeros state and do not want to preserve continuity between successive invocations of the decoder.

Table 2. Viterbi decoder modes

Since the continuous mode has some delay in decoding, it should be noted that the bit rate may be attenuated for the mode. Both modes are experimented in comparison with other codes.

III. CONSIDERATIONS

A. *Random and Burst Error Correction*

In correcting errors, it is necessary to understand possible types of errors present in the transmitted symbols. First is burst errors—errors that occur in many consecutive bits rather than occurring in bits independently of each other. Such errors can be corrected by Reed-Solomon codes, which operate on alphabet sizes larger than binary (since nonbinary) and this property allows for excellent burst error correction capabilities. Burst errors are bound to occur in clusters, there is a high chance of several binary errors contributing to a single symbol errors. Another type is random errors, which occur randomly in a data stream due to its inherent limitations—this can be well handled by BCH codes.

B. *Tradeoffs*

Upon selecting a coding scheme, tradeoffs among BER performance, time complexity and amount of redundancy needed to be taken into account. As more symbols are sent through the AWGN channel or the block and message length increases, the time complexity follows to increase.

Experimentation with equalizer in place in addition to the error correction codes showed no better results—even worse some times. This is due to correction of burst errors in the symbols through the equalizer, and the equalized signal through decoder may not improve due to “ordered” errors, which are rather harder to correct than dispersed ones.

C. *Equations (SNR, bit rate)*

Bit rate is the ratio of usable bits to total message bits sent. In error correction coding, the code rate determines how many usable bits are received at the end, whereas when using equalizer the training sequence determines the rate. For two-level error control codes, the code rates are multiplied. SNR noise scaling is another measure that needs to be adjusted—

the SNR increased by the code rate needs to be added in calculation, as in Equation 3.

$$\text{Bit rate} = (\text{sampling rate}) * (\text{\# bits/symbol}) * \prod \frac{k}{n},$$
$$\text{SNR (dB)} = \text{Eb/No} + 10*\log(2) + 10*\log(\text{code rate}) - 10*\log(\text{sampling rate}) \text{ (only for BPSK)}$$

Equation 3. Bit rate and SNR adjustment

IV. EXPERIMENTS

The aforementioned error correction coding schemes are experimented with the following parameters:

#	Level	Code Type	Parameters
1	Single-level	Block (3 types)	**n = 15, k =11**
2			**n = 255, k = 247**
3		BCH	**n = 15, k =11**
4			**n = 255, k = 247**
5		RS	**n = 15, k =11**
6			**n = 255, k = 247**
7		Convolutional	code rate = ½, mode = **trunc.**, decision = hard and soft, traceback depth = 100, trellis = (171,133), K = 7
8			code rate = ½, mode = **cont.**, decision = hard and soft, traceback depth = 100, trellis = (171,133), K = 7
9	Two-level	Convolutional with BCH	**n = 15, k =11,** code rate = ½, mode = trunc., decision = hard and soft, traceback depth = 100, trellis = (7, 5), K = 3
10			**n = 255, k =247,** code rate = ½, mode = trunc., decision = hard and soft, traceback depth = 100, trellis = (7, 5), K = 3
11		Convolutional with RS	**n = 15, k =11,** code rate = ½, mode = trunc., decision = hard and soft, traceback depth = 100, trellis = (7, 5), K = 3
12			**n = 255, k =247,** code rate = ½, mode = trunc., decision = hard and soft, traceback depth = 100, trellis = (7, 5), K = 3

Table 3. Parameters for the performed experiments

V. RESULTS

The MATLAB simulation results for the experiments listed above are shown in the following figures and tables.

A. *Single-level code*

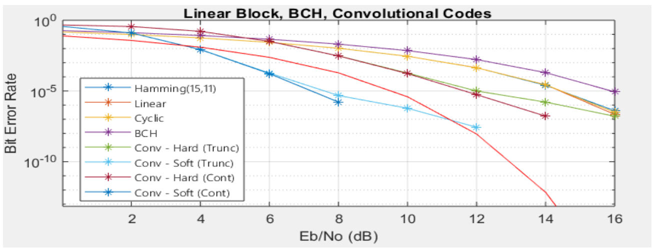

Fig 3. Block, BCH, Convolutional Code (15,11) **(1), (3), (7),** Symbols = 10,000

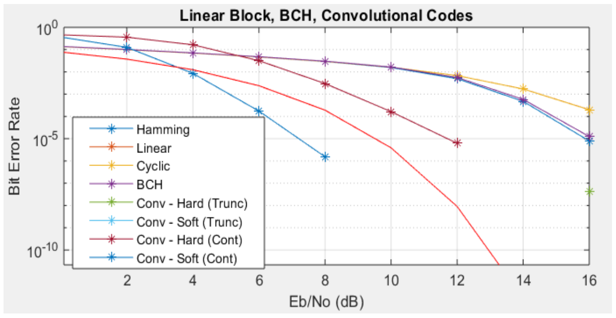

Fig 4. Block, BCH, Convolutional Code (255,247) **(2), (4), (8),** symbols sent = 10,000 * 'trunc' = 'cont' this case (overlapped)

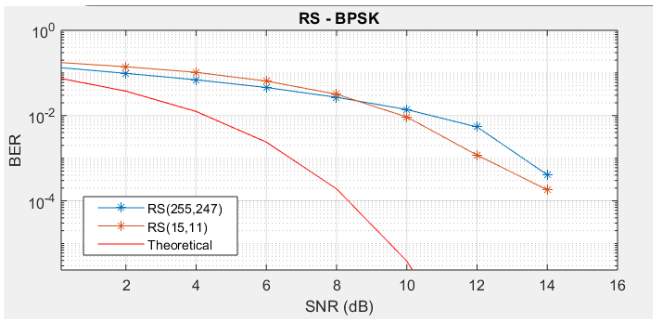

Fig 5. RS Code (15,11) and (255,247) **(5), (6)**, Symbol sent = 1,000

- BER data at 12 dB (15,11)

Data for (15,11)

Code Type	BER		Bit Rate
Hamming block	4.358	e-4 (at 12 dB)	0.73
Generic linear block	4.358	e-4 (at 12 dB)	0.73
Cyclic block	4.358	e-4 (at 12 dB)	0.73
BCH	1.671	e-3 (at 12 dB)	0.73
RS	5.417	e-3 (at 12 dB)	0.73
Convolutional (1/2) – Hard (Trunc)	9.973	e-6 (at 12 dB)	0.50
Convolutional (1/2) – Soft (Trunc)	4.755	e-6 (at 8 dB)	0.50
Convolutional (1/2) – Hard (Cont)	5.468	e-6 (at 12 dB)	0.495
Convolutional (1/2) – Soft (Cont)	1.633	e-6 (at 8 dB)	0.495

* continuous mode has delay(=traceback depth), reducing the bit rate

- BER data at 12 dB (255,247)

Data for (255,247)

Code Type	BER		Bit Rate
Hamming block	4.964	e-3 (at 12 dB)	0.97
Generic linear block	6.814	e-3 (at 12 dB)	0.97
Cyclic block	6.798	e-3 (at 12 dB)	0.97
BCH	5.593	e-3 (at 12 dB)	0.97
RS	1.182	e-3 (at 12 dB)	0.97
Convolutional (1/2) – Hard (Trunc)	6.195	e-6 (at 12 dB)	0.50
Convolutional (1/2) – Soft (Trunc)	1.539	e-6 (at 8 dB)	0.50
Convolutional (1/2) – Hard (Cont)	6.195	e-6 (at 12 dB)	0.495
Convolutional (1/2) – Soft (Cont)	1.539	e-6 (at 8 dB)	0.495

* convolutional – hard (trunc) and (cont) overlapped, and also for soft

B. Two-level code

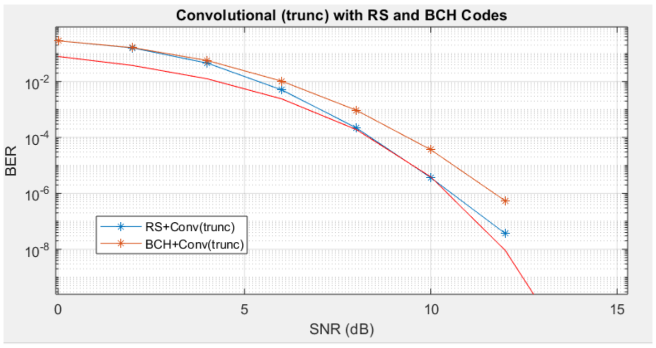

Fig 6. Two-level Convolutional BCH and RS Code (15,11) **(9), (11)**

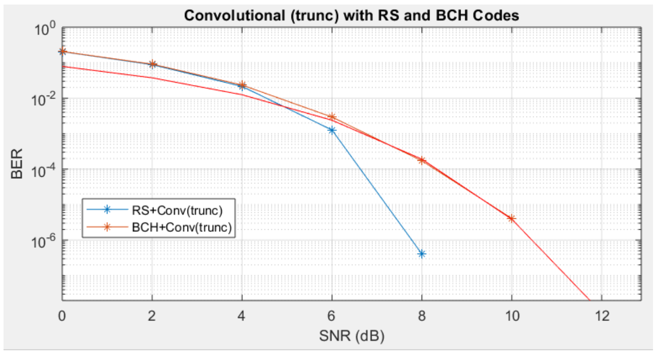

Fig 7. Two-level Convolutional BCH and RS Code (255,247) **(10), (12)**

- BER data for (15,11) and (255,247), truncated mode

Data

Code Type	**BER**	**Bit Rate**
Convolutional (1/2) – RS (15,11)	3.636 e-8 (12 dB)	0.37
Convolutional (1/2) – RS (255,247)	4.049 e-7 (8 dB)	0.48
Convolutional (1/2) – BCH (15,11)	5.364 e-7 (12 dB)	0.37
Convolutional (1/2) – BCH (255,247)	3.872 e-6 (10 dB)	0.48

* mode for convolutional: truncated, nSymbols = 1e3

C. Analysis

- For single-level coding scheme, the convolutional code reached the desired target of 10e-6 at 12 dB SNR. Hard decision mode was relatively easier to achieve with smaller number of symbols transmitted than soft decision mode, and thus with 10,000 symbols it did not reach a nonzero value at 12 dB with soft decision. With the same number of symbols sent, BCH code performance was not outstandingly better than linear block codes; with one order less symbols sent, RS code was not much better either. Possible improvements could result from specifying the generator polynomial to the BCH encoder/ decoder, or not specifying it for RS code (since the opposite was conducted). The bit rate for the convolutional codes, although it resulted in better BER, was as low as its code rate, which is lower than the bit rate for other block codes used. This implies that there is inverse relationship between BER and bit rate. The bit rate for continuous mode was lower than the truncated mode because there was supposedly a delay of length equal to the traceback length, and that is not usable at the end until the delay arrives; thus it was accounted for in the bit rate calculation.
- For two-level code, the truncated mode for convolutional codes are used. The two-level error correction code works by coding with either BCH or RS code in the outer part, and convolutional coding the inner part of the processing loop. The results show slightly better BER than single-level for that reason; however, the bit rate decreases as its price. Since it is double-coded, its time complexity is much higher than single, and therefore only thousand samples were tried; it is evidenced in Fig 7., where it fails to achieve nonzero BER at higher than 10 dB SNR, whereas in Fig 6., it does achieve until 12 dB SNR.

D. Coding Gains

- There seems to be not much coding gain in SNR for code types other than soft-decision convolutional codes. This is clearly evidenced in Fig 7., in which BER of order of 10e-7 is achieved with 2 dB coding gain with Conv(1/2)+RS(255,247) than with Conv(1/2)+BCH(15,11).

VI. Conclusion

The target BER of 10e-6 was achieved by the convolutional encode – Viterbi decode scheme.

Acknowledgment

Thank you professor Keene, for helping to guide through various error correction coding schemes, calculation of bit rates, and also other advice forgotten to be mentioned.