# Experiment # 5

# Introduction to Error Control Codes

## 1 Purpose

The purpose for this experiment is to provide you with an introduction to the field of error control coding. This will be done in three steps. First, the Binary Symmetric Channel (BSC) model will be introduced. You will vary the probability of error for the channel and observe the bit error rate. Second, a system utilizing repetition code will be modeled and the same BSC will allow you to vary the probability of error and observe once again the bit error rate. Finally, you will model a system utilizing linear block coding and will utilize Hamming Codes for error correction.

At the end of this experiment, you should have a better understanding of:

- The binary symmetric channel and bit error rate (BER);
- Repetition encoding and decoding and the calculation of its BER;
- Linear block codes, generator matrices, parity-check matrices and syndromes.

## 2 Background Reading and Preparation

The relevant reading for this experiment can be found on Chapter 16 of [1] and Chapter 10 of [2]. On the course notes, this experiment will cover from pages 24.1 to 25.6. Another source often referenced is [3].

*Before* coming to the lab, complete the **lab preparation** and hand it to the T.A..

## 3 Equipment

Hardware:

1. One notebook workstation;

Software:

1. `Matlab` Release 12
2. `Simulink` with ECE417 Toolbox

## 4 Experiment

Congratulations! Just a couple more weeks and you will graduate. Hopefully the ECE417 labs will have equipped you with some useful tools for your professional practice. For the second and last time, the approach to this experiment will be "design and simulate" only. All necessary blocks will be found on the ECE417 blockset. If you cannot figure out what some of the blocks actually do, ask the T.A. for assistance.

**All results are to be reported in the spaces provided in this outline. Make sure that the T.A. verifies every result you record.**

### 4.1 Binary Symmetric Channel and Bit Error Rate

In this part of the experiment, you will utilize a binary symmetric channel to introduce errors on your received signal. You will vary the probability of error introduced by the BSC and calculate the bit error rate. The purpose of this exercise is for you to become familiar with how to calculate the bit error rate, knowing all other variables. It is a straight forward procedure, which will provide a parameter for comparison for the other steps of the experiment. The objective of this part of the experiment is to draw a curve of probability of error versus bit error rate.

Start by opening a new model and dragging in a PN sequence generator, a binary symmetric channel and two "to workspace" blocks. Attach the PN sequence generator to the BSC and the output of the BSC to one "to workspace" block. The other "to workspace" block will take the signal from the PN sequence generator. Next, bring into your model a "logical operator" block and a scope. The "logical operator" will take the signal from the PN sequence generator and from the output of the BSC, and perform an XOR operation on them, and send the result to the scope. Remember to set the "to workspace" blocks to "array" format and limit its size to 20000 points. The purpose for this operation is to indicate where the input and the output sequences differ, or where the error is introduced, as well as how many errors are there.

You must set the PN sequence generator to a sequence large enough to make the experiment meaningful relative to the probability of error chosen. This is to say that if the probability of error is 1 to 100, a good size for the incoming sequence is 10000. The generator polynomial `[0 -9 -10 -12 -13]` will give you a pseudo-random repeating sequence of 8191 bits. As for initial states, you can use `[ones(1,13)]`. Use a sample time of 1/8000. Another polynomial you can use is `[0 -4 -8 -13 -14]`, which will generate a sequence with 16383 bits.

Run the model and collect the data. You are to reshape the collected data appropriately and count the errors for every different probability of error chosen for the BSC. You must write a `Matlab` routine that counts errors on the output signal and computes the bit error rate. The bit error rate is calculated by counting the "ones" from the XOR operation and dividing them by the length of the output sequence. To make sure things work, start with *zero* probability of error on the BSC, and run your model. The scope should show a continuous line at zero. Here we go.

*Be prepared to answer questions asked by the T.A. You will be marked on these answers as well.*

- *Considering* $P(\varepsilon) \in (1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128)$, *draw the curve probability of error* *versus bit error rate (output). Build a table as well with the values, so that the TA can check if they are good.*

## 4.2 Repetition Code

Let us now introduce a code to our sequence. The most straight-forward coding technique is repetition coding, which repeats every element of the sequence of bits containing the information. The decoding is done by a technique called *Majority Logic Decoding*. On your ECE417 block library you will find a "Repetition Encoder" and a "Majority Logic Decoder". Put a model together similarly to what you have done in the previous section, this time adding the *enconder* between the PN sequence generator and the BSC, and the *decoder* between the BSC and the output ("to workspace") block. Do not remove the XOR block or the scope; they will come in handy.

Take a closer look at both encoder and decoder. You should be familiar with the structure used on the encoder. Run your system and collect the data. Make sure to start using *zero* probability of error on the BSC, so that you know your system is working. You will need to introduce some delay on the PN sequence path that goes into the XOR block. This is because the encoding and decoding will delay the output sequence relative to the original PN sequence. In order for you to know how much delay you need to introduce, you will have to look at both sequences in `Matlab` and introduce the delay accordingly. Either that or you can do it the *Neanderthal way* and repeatedly change the value of the integer delay block until you see the scope with a continuous line at zero.

- *What does the decoder do? (i.e., how does it work?) How is the BER calculated for this case?*

- *Considering* $P(\varepsilon) \in (1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128)$, *draw the curve probability of error versus bit error rate (output). Build a table as well with the values, so that the TA can check if they are good.*

- *How is this curve different than the one in the previous section? Is this a better or worse performance than no coding at all? What is the price one paid?*

## 4.3 Linear Block Codes

In this section you will experiment with a more complex coding technique. You will design a system which utilizes a [7,4] linear coding scheme. Your model is to be put together similarly to the one using the repetition encoder. This time, the proper blocks to be used are the "Linear Block Encoder" and the "Linear Block Decoder". You will be responsible for providing the proper parameters for these blocks to operate.

If you double-click on the blocks, you will see the sub-system they represent. Within each sub-system, you will find highlighted blocks. These are the ones you are responsible for designing. At this point, you are familiar with Generator Matrices, Parity Check Matrices and Syndromes (so we hope).

Given the Parity Check Matrix in `Matlab` format: $H = [1001011; 0101110; 0010111]$, you will be asked to calculate the other paramaters for the (7,4) code. Your best option is to write a little `Matlab` program that generates all matrices, and run it once. Then all you need to do is to call up the matrices by their label from the blocks where they are supposed to be.

If you want to view your original signal as compared to the received (output) signal, do not forget that you may need a delay block somewhere along the line. Keep the XOR operation block and the scope handy, so that you can view quickly if you are heading towards the right direction.

- *What is the Generator Matrix for this (7,4) linear code?*

- *Assuming a maximum of one error per code word, what is the decoding table for the code? Present the syndrome and the error pattern*

- *How does the decoder work for a maximum of one error per codeword?*

- *Considering* $P(\varepsilon) \in (1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128)$, *draw the curve probability of error versus bit error rate (output). Build a table as well with the values, so that the TA can check if they are good.*

- *On what would you base your decision to utilize repetition code or a (7,4) linear block code? (This is an artsie question; ask the T.A. what he or she would like you to emphasize here.)*

## 5 Going The Extra Mile

Your Engineering (capital E) programme has made you run many extra miles, this time you will have a well-deserved break. You will have many more extra miles to run in your professional practice!

## 6 Conclusion

Since this experiment concludes the laboratory portion of ECE417, it is expected that through your lab practice you were lead to understand better the concepts seen in the theory. In this experiment, the concepts involved were bit error rate calculation, repetition coding and decoding, and linear block coding and decoding. This experiment is a mere introduction to the field of error control coding. Those interested in further advancing in this field should take a graduate course such as ECE1501.

**Obs: The books cited in this outline contain relevant material for the student to better understand the topics pertaining to the experiment. The student should note that by reading the course textbook only, one should be able to successfully perform the experiment. This is to say that it is not necessary for the student to purchase all the references.**

## References

[1] B. P. Lathi, *Modern Digital and Analog Communication Systems*, 3rd Edition. New York: Oxford University Press, 1998.

[2] S. Haykin *Communication Systems*, 4th Edition. Toronto: John Wiley & Sons, Inc., 2001.

[3] S. Lin and D.J.Costello *Error Control Coding*, Chapters 1, 2 and 3 Prentice-Hall, Inc., NJ, 1983