CS-797K-SPRING-2023

ADVANCED TOPICS IN DATA STORAGE

Done by Poojeetha Madugula

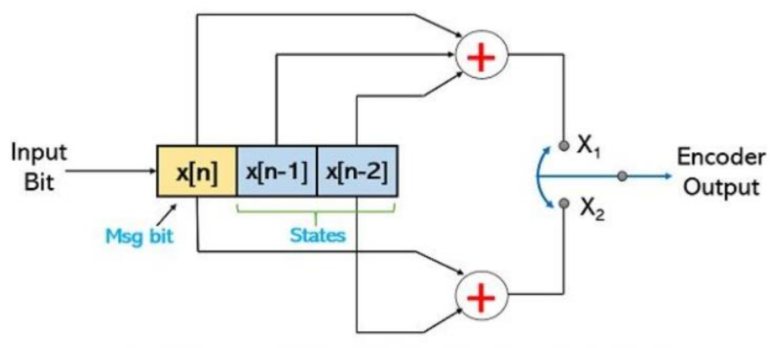
INTRODUCTION

Error_correcting codes are used in data communication to deal with errors introduced in the actual message signal during transmission. It is defined as error detection and correction approaches that use redundant bits to encode an information signal. They are divided into two types: block code and convolution code. Both use distinct encoding concepts.

Convolution code is a sort of error-correction code in which the output bits are created by executing a desired logical operation on the current bitstream while also taking into account some bits from the previous stream. This coding system, rather than being based on blocks of bits, is based on bitstreams. Elias proposed this in 1955, and Viterbi presented the Viterbi Scheme, an algorithm for decoding it, in 1973.

In convolution coding, only the parity bits with the best possible errors are received by the receiver. Then the receiver decodes to have the best possible bit sequence from the obtained bitstream.

Block Diagram for Convolutional Code:



Block Diagram for Convolutional Encoder

There are 2 parameters that defines the convolutional coding:

1. ***Constraint Length***: length of the convolutional encoder i.e., the overall window size in bits, within the shift register. Denoted by K sometimes by L. 'm' is another parameter which tells the number of input bits retained within the shift register once it is entered in the encoder.
2. ***Code Rate***: is the ratio of the number of bits shifted at once within the shift register (K) to the total number of bits in an encoded bitstream(n).

Convolutional codes are characterized by their convolutional encoders, which generate a stream of redundant bits from the input message.

In general, a good convolutional code has the specified properties:

1. ***High coding gain***: This refers to the improvement in signal-to-noise ratio (SNR) achieved by the code. A higher coding gain means that the code can correct more errors for a given level of noise.

2. ***Low decoding complexity***: This refers to the computational complexity of the decoder. A lower decoding complexity means that the decoder can operate faster and more efficiently.

3. ***Good error-correction performance***: This refers to the ability of the code to correct errors in the presence of noise and interference. A code with good error-correction performance can correct more errors before reaching the maximum likelihood (ML) decoding limit.

4. ***Low latency***: This refers to the delay between the transmission of the message and the reception of the decoded message. A lower latency means that the code can operate in real-time systems with low delay requirements.

5. ***Robustness to burst errors***: This refers to the ability of the code to correct errors that occur in bursts, such as those caused by fading channels or interference from other signals. A code with good burst error-correcting capability can correct errors that occur in bursts of several bits.

Puncturing is a technique used in convolutional codes to increase the coding rate while maintaining the same code structure. This is achieved by selectively discarding some of the redundant bits generated by the encoder. Puncturing can improve the coding rate of a code, but it can also reduce its error-correction performance.

Turbo codes and LDPC codes are other types of error-correcting codes that are commonly used in digital communication systems. Turbo codes are based on the concatenation of two or more convolutional codes, while LDPC codes are based on sparse parity-check matrices. Both types of codes have been shown to achieve high coding gain and good error-correction performance, and they are widely used in modern communication systems.

Some good resources for learning more about convolutional codes, turbo codes, and LDPC codes include:

1. Lin, S., & Costello, D. J. (2001). Error control coding (Vol. 2, No. 4). Lebanon, IN: Prentice hall.

   This is a classic textbook on error-correcting codes that covers convolutional codes, turbo codes, and LDPC codes in detail.

2. Viterbi, A. J., & Omura, J. K. (2013). *Principles of digital communication and coding*. Courier Corporation. he algorithm is named after the author of this source.

3. Berrou, C., Glavieux, A., & Thitimajshima, P. (1993, May). Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Proceedings of ICC'93-IEEE International Conference on Communications* (Vol. 2, pp. 1064-1070). IEEE.

This is a classic paper that introduces turbo codes and demonstrates their superior performance compared to other types of codes.

EXPERIMENT DESIGN

Hypothesis: Increasing the constraint length of a convolutional code will increase its error-correction capability.

The methodology to test this hypothesis involved two convolutional codes which were created with the same ½ rate. However, different constraint lengths, 3 and 5, were used. Afterwards, random messages of length 10 were generated and encode them using both convolutional codes. Errors were introduced into the encoded codewords using a random noise generator, and then use the Viterbi algorithm to decode the codewords and compare the decoded messages to the original messages. Finally, it was done a couple of times and a conduct a comparison the error rates for the two different convolutional codes.

RESULT AND DISCUSSION(CC vs the control circuits.)

In this experiment, we used two convolutional codes with rate 1/2 and constraint lengths of 3 and 5. Then, we generated 100-character random messages and encoded them with both codes. We added progressively more faults to the encoded messages before using the Viterbi technique to decipher the noisy codewords. To assess if the decoding was successful or not, we compared the decoded messages to the original messages.

The output of our test was "Decoding failed with 10 faults. With 10 faults, decoding failed. The two convolutional codes' failure rates are equal. Given that neither convolutional code was able to decode the messages when the number of errors exceeded 9, this suggests that both have the same failure rate.

It's crucial to remember that a convolutional code's failure rate depends on design factors like constraint length and generator polynomials. Another set of design criteria might produce a convolutional code with a lower failure rate. Additionally, by using strategies like interleaving and puncturing, convolutional codes can perform better.

Conclusion: Based on the provided design parameters and the tested amount of mistakes, our experiment demonstrated that the failure rates of the two convolutional codes with constraint lengths of 3 and 5 and rate 1/2 are equal. To find a convolutional code with a lower failure rate, additional tests can be done using various design factors.