Project Information theory
Compact Disc Digital Audio

Arthur Duysens, Kaixin Chang

1 Introduction

This document describes the project assignment for the course on Information theory. In this project, the application of error-correction coding in audio CDs will be studied. This project will be carried out by groups of 3 or 4 people and consists of writing a report (in English), implementing a Reed-Solomon coder/decoder and implementing part of the audio CD system in Python. The report and code should be uploaded to Ufora as a zip-file with the name groupXX.zip with a folder called groupXX containing all the files. **The project deadline is at 23:59 on the 3rd of May.** If you have questions, you can send an e-mail to **arthur.duysens@ugent.be** or **kaixin.chang@ugent.be**.

2 Task division

The project report should contain a section on how you divided the tasks among the group members and who eventually did what. Try to make a fair division! After the project deadline, there will also be an oral defense and a discussion of your task division. For the oral defense it will be assumed that every group member can answer questions about every part of the project. The planning of this will be made available on Ufora.

3 Compact Disc Digital Audio

The compact disc digital audio system, CD for short, was introduced by Sony and Philips in 1980 and commercially introduced in 1982. This, at the time, revolutionary new consumer product was very successful, surpassing the sale of the analog LP and music cassette tapes in less than 10 years after its introduction. The CD system can be considered as a transmission system that brings sound from the studio into the living room. The left and right audio channels are sampled at 44.1 kHz and are represented on the CD as 16 bit linear PCM samples, which are encoded into data bits and modulated into channel bits to be sent along the 'transmission channel' consisting of write laser, master disc, user disc and optical pickup. Imperfections of the disc and noise in the readout

system produce errors in the recovered data. To prevent these errors from impairing the sound at the output of the CD player, error correction coding using Reed-Solomon codes is a part of the CD audio system. In fact, CDs were the first application of Reed-Solomon codes in a mass-produced consumer product. The specifications of the CD audio system are known as the 'Red Book' because of the color of its cover. In the project folder on Ufora you can find a digital version of the CD audio standard.

Figure 1: Compact Disc Digital Audio logo.

4 Cross-Interleaved Reed–Solomon coding (CIRC)

To protect the data on the disc from scratches, fingerprints, dust, etc., audio CDs use a Cross-Interleaved Reed-Solomon coding (CIRC) scheme. A CIRC encoder consists of a Reed-Solomon code C_2 followed by a cross-interleaver that distributes the symbols of a C_2 codeword among several information words of a second Reed-Solomon code C_1. This allows the CIRC scheme to correct both random and burst errors. The interleaver at the input of C_2 and at the output of C_1 further improve the performance of the system. The CIRC decoder undoes the interleave operations and decodes the Reed-Solomon codes. The CD audio standard defines the characteristics of the different interleavers and Reed-Solomon codes.

The standard does not define how decoding should be performed, in this project the strategy described in Algorithm 1 is used. In this algorithm, f is the number of erasure flags at the input of the C_2 decoder.

5 Assignment

5.1 Audio CD Subcode

The audio CD frames do not only contain the CIRC encoded audio samples, but also 27 sync bits and 8 subcode bits. To answer the following questions, you will need to consult the CD audio standard.

1. What is the datarate of the subcode in subcode blocks/s? And in bit/s?
2. The subcode consists of 8 channels (P, Q, R, S, T, U, V and W), complete Figure 3 with the state of the channel P flag bit.

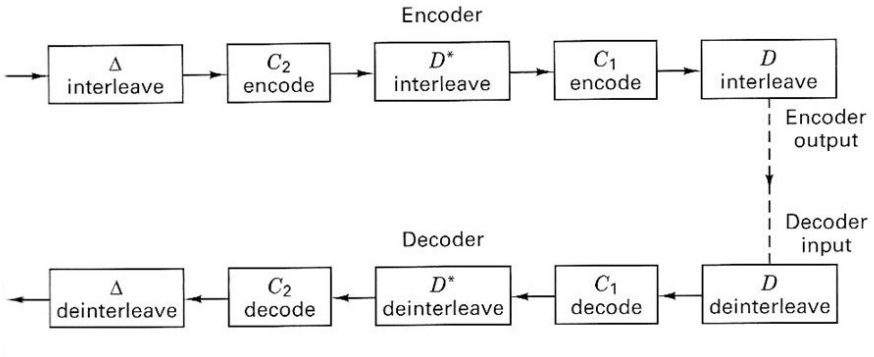

Figure 2: Schematic representation of the CIRC scheme. (The interleaver before C2 and the deinterleaver after C2 are not implemented in this project.)

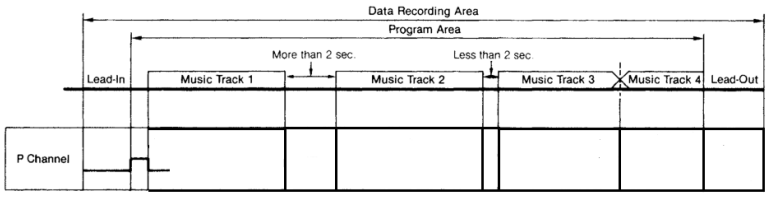

Figure 3: Complete the state of the P Channel flag bit in this figure.

3. How does the CD player know the number of tracks, the starting positions of the tracks and the total playing time of the audio CD? How is this data protected from errors?

5.2 Reed-Solomon code

During CIRC encoding of the audio samples, parity symbols are generated using two shortened Reed Solomon codes in GF(2^8). In this part of the project you will complete the class *RSCode* with functions to construct a generator polynomial, encode Reed-Solomon codewords and correct erroneous codewords. Note that you may **not** use built-in Python implementations for RS coding/decoding or for the construction of the generator polynomial.

1. Encoding.

   (a) Let α be a primitive element of GF(2^8). Consider the polynomial

$\underline{C_1\text{Decoder}}$
if zero or one error
then modify at most one symbol accordingly
else assign erasure flags to all symbols of the received word
$\underline{C_2\text{Decoder}}$
if zero or one error
then modify at most one symbol accordingly
else
if $f > 2$
then copy C_2 erasure flags from C_1 erasure flags
elseif $f = 2$ **and** error correction successful
then modify the symbols accordingly
else assign erasure flags to all symbols of the received word

Algorithm 1: Decoding strategy.

given by: $p(D) = D^8 + D^4 + D^3 + D^2 + 1$. Show that $p(D)$ is a primitive polynomial with root α for the field GF(2^8).

(b) Construct the generator polynomial $g(x) = g_0 + g_1x + g_2x^2 + \ldots + g_{n-k}x^{n-k}$ for the C_1 Reed-Solomon code described in the standard. Give the coefficients g_i for $i = 0..n-k$ (in GF(2^8)). What is the value of m_0?

(c) Give the corresponding check polynomial $h(x)$. It is not necessary to write down the complete polynomial, giving the first and last 5 terms is sufficient.

(d) What is the minimal Hamming distance for this code. Use the fact that a Reed-Solomon code is an MDS code.

(e) Explain why a Reed-Solomon code is -in general- more appropriate for data protection than a binary BCH code.

(f) Implement the functions *makeGenerator()* and *encode()*.

2. Decoding.

(a) How many erroneous *bits* can the C_1 Reed-Solomon code always correct? What's the maximum number of correctable bit errors?

(b) Give the relationship between the bit error rate P_b and the symbol error rate P_s for GF(2^8) (assuming the bit errors are not correlated).

(c) Implement the function *decode()*. You can use any of the algorithms that are discussed in the course notes.

5.3 Audio encoding in CDs

In the last part of the project, you will complete the class *AudioCD* with the necessary functions to allow conversion of a PCM audio file to the corresponding data bits and to allow recovering of the original audio file from an impaired version of these data bits. Note that in this part of the project, the

encode and decode functions from the Python reedsolo package are allowed to be used as the encoder and decoder from Section 5.2 would be too slow and can't handle the combined recovery of errors and erasures. Look at the functions *C3_enc_8_parity()* and *C3_dec_8_parity()* for an example of how to use the Python implementations.

1. Explain the function of every component of the CIRC structure in figures 12 and 13 of the standard ('Delay of 2 frames', 'Interleaving sequence', 'C_2 encoder', 'Delay lines of unequal length', 'C_1 encoder' and 'Delay of 1 frame'). Which types of errors (random, short burst, long burst) does C_1 protect against? And C_2? Please give short explanation.

2. Complete the functions in the *AudioCD* class. Decoding has to be performed as described in Section 4 of the assignment. In Python you need to determine the number of detected errors from the list of erasure positions and the third output of the reedsolo decoder. Note that this third output contains the list of positions of the errata (errors **and** erasures) and that if the value of the data at an erasure position is correct, this position is not added to the list.

3. What is the maximum duration of a burst (in bits) that can always be corrected if we assume that C_2 can correct up to 4 erasures? Translate this to the maximum width of a scratch if the scanning velocity of the laser is 1.3 m/s. (Hint: the audio samplerate is 44.1 kHz.) Compare this with the simulation results of your Python code and explain.

4. At the output of the CIRC decoder, an interpolator masks the unrepairable errors using linear interpolation of a maximum of 8 subsequent samples. Describe, qualitatively, why this works and what the relation with the sample frequency of the system is.

5. Compare the performance of the 4 configurations (see *AudioCD* class) under the following circumstances:

   (a) A scratch of 100, 3000 and 10000 databits wide, repeated with a period of 600000 bits (the scratch is encountered at every rotation of the disc).

   (b) Random bit errors with a bit error probability p between 0.05 and 0.001 generated by the Python command the *numpy.logspace(-1-math.log10(2),-3,10)*. Plot the probability that an audio sample has been flagged as an erasure (before interpolation) and the probability that an audio sample could not be interpolated, for configurations 1, 2 and 3 (configuration 0 performs no error correction). Note: think about the scale (linear or logarithmic) you use for the axes!

   Explain your observations!

6. After CIRC encoding, the databits are modulated with 'EFM' before being written to the disk. What is EFM and why is it used? Could the EFM demodulator give extra information to the CIRC decoder to improve its error correcting capability?

6 Python implementation

On Ufora the files *RSCode.py* and *AudioCD.py* are provided, these define the classes of the same name. For this project, you need to complete some functions to make the classes functional. It is important that the functions use the correct input and output. This is always described in detail in the functions. This is necessary such that the files you submit can be tested for proper functionality. Note that all files will also be tested for plagiarism. For *RSCode.py*, you must use the galois package (https://galois.readthedocs.io/en/stable/index.html). For this part, you will need the galois.GF() and galois.Poly() classes. The remainder of a polynomial division can be determined with the % operator. For *AudioCD.py*, you must use the encode and decode function of the reedsolo package (https://github.com/tomerfiliba/reedsolomon). Note that the elements of the data array you pass to the encode and decode function must be **(unsigned) bytes**. With a numpy array you can e.g. achieve this with *.astype('B')*. Make sure to provide sufficient comment in your code. Both classes also contain a static *test()* function that you can use to test your implementation.

Note:

- Please comment your code well for better readability.
- Please do not modify the input and output arguments of the functions.

Good luck!