Information Theory

Project Compact Disc Digital Audio

Group 04

2019-2020

Matthias Cami, Marijn De Kerpel, Zeno Dhaene and Giani Ollivier

Supervisors: prof. dr. ir. Heidi Steendam, ir. Johannes Van Wonterghem and ir. Stef Vandermeeren

Contents

1 Introduction

This report contains the answers to the questions from the project assignment. Furthermore results of the implemented parts will be discussed and compared. During this project the encoding and decoding part of the Compact Disc (CD) format of Philips and Sony was investigated. More specificaly during the project a Reed-Solomon encoder and decoder are implemented and investigated. In addition, the encoding and decoding chain as specified for the Compact Disc is implemented and investigated of which a Reed-Solomon code is a part. This will provide insight into how it is possible that an average CD can usually be played without noticeable errors, despite its susceptibility to fingerprints, scratches, etc.

2 Answers

2.1 Audio CD Subcode (5.1)

1. **What is the datarate of the subcode in subcode $\frac{blocks}{s}$? And in $\frac{bit}{s}$?**

   - The datarate of the subcode is 7350 $\frac{blocks}{s}$ or 58800 $\frac{bit}{s}$. This can be derived based on the fact that there are 75 sectors per second, with 98 frames per sector and one subcode symbol of 8 bit per frame.

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.**

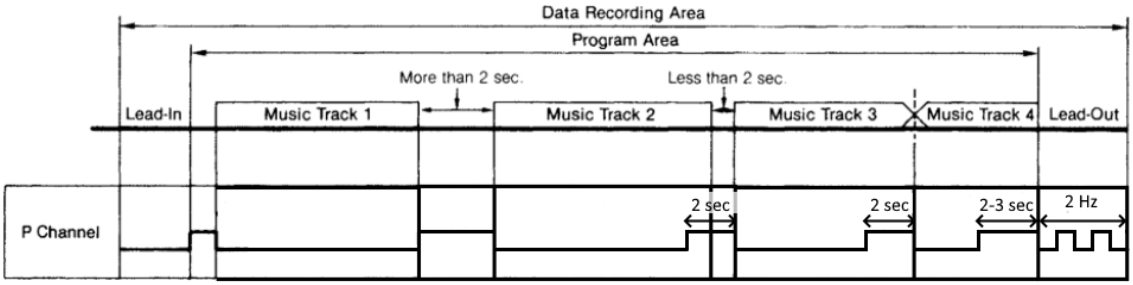


Figure 1: Solution

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?**

   - This information is stored in the TOC (Table of Contents) entry. This entry is stored in the lead-in area of the data recording area, on the Q-track. The data is protected from errors by repeating the TOC entry continuously during the lead-in, and is thus stored more than once.

2.2 Reed-Solomon code (5.2)

2.2.1 Encoding

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

   -To prove that $p(D)$ is a minimal polynomial with root α, the minimal polynomial of α is calculated (which is meant to be 00000010). The fact that $m(x^2) = m(x)^2$ can be taken into account, so that $m_1(x) = (x - \alpha) \cdot (x - \alpha^2) \cdot (x - \alpha^4) \cdot ...$. This is calculated in the static function *question_1()* in the *RScode.m* file.

   The calculated polynomial does indeed equal $p(D)$. By seeing that the polynomial $p(D)$ is the minimal polynomial of α and the polynomial is primitive (as stated

in the assignment), it can be concluded that $p(D)$ is a primitive polynomial with root α. Checking that α is primitive can be done by calculating all powers of α and verifying that none of them are equal to one until α^{255}.

2. **Construct the generator polynomials $g(x) = g_0 + g_1x + g_2x^2 + \ldots + g_{n-k}x^{n-k}$ for the C1 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?**

   -The generator polynomial is calculated in the *makeGenerator(m,t,m$_0$)* function in the RScode.m file. For the C_1 code, $m = 8$, $t = 2$ and $m_0 = 0$ according to the CD specification, the value of m_0 can be derived from the check matrix on figure 11, since the first row only contains the value 1, it can be concluded that it must be one of the roots. This results in the following coefficients: $g_0 = 1, g_1 = 15, g_2 = 54, g_3 = 120, g_4 = 64$, these are calculated (by using the definition of a generator polynomial in RS codes) in the static function *question_2()* in the *RScode.m* file.

3. **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.**

   -The check polynomial is defined as $\frac{x^n-1}{g(x)}$, In this case, the check polynomial can be calculated as $h(x) = (x - \alpha^4) \cdot (x - \alpha^5) \cdot (x - \alpha^6)...(x - \alpha^{254})$. This results in [1 ; 30 ; 145 ; 218 ; 41] for the first 5 coefficients and [237 ; 189 ; 77 ; 111 ; 233] for the last 5 coefficients. These coefficients were calculated in the static function *question_3()* in the RScode.m file with the same values for the variables as the previous question.

4. **What is the minimal Hamming distance for this code. Use the fact that a Reed-Solomon code is an MDS code.**

   -MDS (Maximum distance seperable) code is a code where the minimal Hamming distance between two codewords is the maximal value possible for a linear code of size(n,k). This Hamming distance is equal to $n - k + 1$ which is equal to 3 for C_1 Reed-Solomon code.

5. **Explain why a Reed-Solomon code is -in general- more appropriate for data protection than a binary BCH code.**

   -A BCH code would be faster and easier to implement, but a Reed-Solomon code is a code over GF(2^8) so it applies error correction over bytes instead of over bits like BCH does, for this reason, it can correct concentrated or concatenated burst errors on bits. A sequence of m + 1 consecutive bit errors can affect at most two symbols of size m.

   A Reed-Solomon code is also a quasi-perfect code which yields the best possible performance for given n and k.

2.2.2 Decoding

1. **How many erroneous bits can the C1 Reed-Solomon code always correct? What's the maximum number of correctable bit errors?**

   -The C_1 Reed-Solomon code has $t = 2$, this means that it can correct up to 2 symbols. However, a symbol consists of 8 bits. A symbol is "wrong" from the

moment 1 bit is wrong, and because of this, the amount of erroneous bits that can always be corrected regardless of their position in the codeword, is also 2 (1 bit per symbol and up to 2 corrected symbols). It could be a symbol is wrong because all 8 bits are wrong, the symbol will still be corrected (as long as there are only 2 symbols to be corrected). This gives us 16 bits as the maximum number of guaranteed correctable bit errors (8 bits per symbol and up to 2 corrected symbols).

Note: it is possible in practice that more symbols could be corrected by chance.

2. **Give the relationship between the bit error rate P_b and the symbol error rate P_s for $GF(2^8)$.**

   -The same reasoning as in exercise 11 on polynomial codes applies, the formulas (6.77) and (6.79) can be combined which results in the fact that the bit error probability is smaller than or equal to $\frac{2^{8-1}}{2^8-1} \approx 0.5$ times the symbol error probability.

2.3 Audio encoding in CDs (5.3)

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', 'C1 encoder' and 'Delay of 1 frame'). Which types of errors (random, short burst, long burst) does C_1 protect against? And C_2?**

   - 'Delay of two frames': by delaying the even samples with two frames, the spatial nature of errors is circumvented (at least partially). If two adjacent symbols are corrupted, the delay makes sure that these are in two different frames. This improves the effectiveness of the interpolation or conceilment of an error which could not be corrected by the Cross Interleaved Reed-Solomon Coding (CIRC) decoder.

   - 'Interleaving sequence': this is closely connected to the delay of two frames, making sure that two frames are mixed together to two new frames that are a mix of the two original frames. After that the scrambling of the samples is done by mixing up the connections to the C2 encoder following a specific pattern. This mixing pattern splits the even and odd numbered samples.

   - 'C2 encoder': the C2 encoder uses the scrambled input connections to add an additional 4 bytes or Q-parity symbols. After the C2 encoder has done its work, the samples that appear close to each other (in time) will be stretched over a big area. This means that the C2 decoder will effectively correct long burst errors, when the small random errors have been filtered out by the C1 decoder already. The parity bits are added in the middle which enlarges the distance between odd and even samples a little more than the distance created by the interleaving sequence and the two symbol delay.

   - 'Delay lines of unequal length': By delaying every symbol by a fixed number of frames, the sequence of the symbols is changed completely, thus making error correction more robust. The corrupted symbols caused by burst errors are spread over multiple frames by the inverse step in the decoder, which helps with correcting them afterwards. The burst error is converted into a collection of seemingly random errors this way (from the view point of C2).

- 'C1 encoder': the C1 encoder reads the symbols from the frames which where scrambled to spread the symbols the C2 encoder has encoded. By encoding these scrambled frames an extra protection layer is added. C1 corrects random errors and detects burst errors.

- 'Delay of 1 frame': this ensures that even if there are two random adjacent symbol errors, there is at most one error per CIRC series, thus making error correction more robust.

- 'Inversion of parity bits': inversing of the parity bits is done to help in the case when there is a large defect. The recorded parity pattern before modulation, if the data is all zero, has only bits with value one. This is also a measure for fixed patterns. It makes it possible to detect bit insertions and deletions because of bad clocking or bit slip.

For the decoding part the inversing steps are done for each step in the encoding part such that the effect of each step is eliminated. If there are non correctable errors this will result in an output that is not equal to the input that was used for the encoding proces, however the most uncorrectable errors will be conceiled by doing interpolation if possible.

2. **What is the maximum duration of a burst (in bits) that can always be corrected if we assume that C2 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 simulation results of your Matlab code and explain.**

-To derive the burst length based on the allowed erasures the shortest distance between 5 successive symbols of 1 frame have to be determined at the end of the CIRC encoder with respect to the C2 encoder (not taking any strategy into account just C2). The first symbol line has a total delay of 1 frame. For the fifth symbol line the total delay is 15 frames. Between the first and fifth symbol the distance however is 15.15 frames (15 frames and 5 symbols), but to be correctable and thus have correct symbols from the fifth symbol and up the actual burst error in raw data (33 byte per frame) may only be 15.12 frames (viewpoint of C2). This corressponds to a burst error of 3992 bit in raw data ($15.12 \cdot 33 \cdot 8$).

Converting this result to the modulated data or channel data which is recorded on CD gives 8888 bits $((15 + \frac{4 \cdot 17}{((1+2 \cdot 12+2 \cdot 4) \cdot 17+27)}) \cdot ((1 + 2 \cdot 12 + 2 \cdot 4) \cdot 17 + 27))$. Knowing that the CD rate is 7350 frames per second makes it possible to calculate the length of the actual scratch. The scratch may have a length of 2.673516 mm $(\frac{15+\frac{68}{588}}{7350} \cdot 1.3)$ for the calculated burst length based on a velocity of 1.3 $\frac{m}{s}$.

- Compared to measurements: the measurements are done based on data existing of 32 symbols per frame (raw data without subcode symbol) therefore the correct value to compare with is a burst of 3872 bits $((15 + \frac{4}{32}) \cdot 32 \cdot 8)$ that is theoretically always correctable if it is assumed that decoder C2 can correct up to 4 erasures. However if the implemented decoding strategy is taken into account this will further reduce as the C1 decoder will flag a full frame as erroneous if more than one error is in a frame. This will result in an always correctable error of maximum 3640 bits $((14 + \frac{7}{32}) \cdot 32 \cdot 8)$ that are erroneous for the implemented case, it can be more depending on the position of the scratch. The actual scratch length would correspond to 2.511986 mm $((\frac{14+\frac{7 \cdot 17}{588}}{7350} \cdot 1.3))$ or 8351 bits $((14 + \frac{7 \cdot 17}{588}) \cdot 588)$.

By doing measurements it is clear the correctable burst varies based on the position or thus the content of the track. The maximum correctable burst achieved during measurements is one of 3689 bits long. The measured results for the varying positions that in this case were tested, were in fact always higher than 3640 bits, so the theoretical always correctable value is certainly achieved in this case based on the measurements. The fact that it depends on the position is logical as the position determines how well the burst error is aligned with the frames. Here it will depend on the used strategy how much frames the C1 decoder will flag. It should be mentioned that the measurement in fact not exactly represents a real burst error of the scratch length, values that where zero before scratching are not affected by doing a scratch so the actual added errors in bits can be lower or higher depending on the position (data varies). If the measurement is done by inverting the bits to have a real burst error of 3640 bits this gives exactly this result as a minimum for multiple tested positions, however higher results are also again achieved.

How well the CIRC decoding performs with regards to random errors and bursts is defined by the chosen decoding strategy as it defines how both decoders C1 and C2 cooperate. One decoding strategy will for example handle better random errors while an other one can better cope with burst errors.

3. **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.**

   - This works because there is high temporal connectivity between adjacent samples. It is very unlikely that one sample differs much from adjacent samples. Because the sampling frequency is so high (44.1 kHz), it can also easily be assumed that the frequency of the samples inbetween two samples follow a constant increase or decrease. This means that linear interpolation is possible between two relatively adjacent samples, which in turn makes the actual interpolation computationally efficient and correct. If the number of symbols that need interpolation gets too high however, the quality of the correction will diminish drastically. If 8 samples need interpolation, for example, the samples must follow a linear pattern for $\frac{8}{44100}$ seconds. For longer interpolations it makes sense that the interpolation won't be correct anymore. If only a few interpolations are needed this won't be noticed by the listener and the conceilment of errors is effective. Further it should be remarked that this will work not as good for music with much high frequency details or components as by doing interpolation in fact this can be seen as virtualy lowering the sampling rate which will thus remove high frequency details.

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

   - To start there will first be taken a look at how many samples are marked as an erasure for the different configurations (Fig. 2a). Of course, configuration zero doesn't do any CIRC decoding, so there can be seen that there are zero samples marked as an erasure. For the other three configurations it is visible that the CIRC decoder marks the most samples as an erasure for the longest scratch, while configurations two and three have about the same amount of erasures marked, which indicates that the CIRC decoder notices the most errors in the data. The lack of erasures for configuration 1 for a scratch length of 3000 bit and lower

is because this configuration can always fully correct these errors (see minimum burst). Configuration 2 and 3 have for these shorter scratches again about the same amount of samples marked as erasures. This should be visible in the upper graph of figure 2.

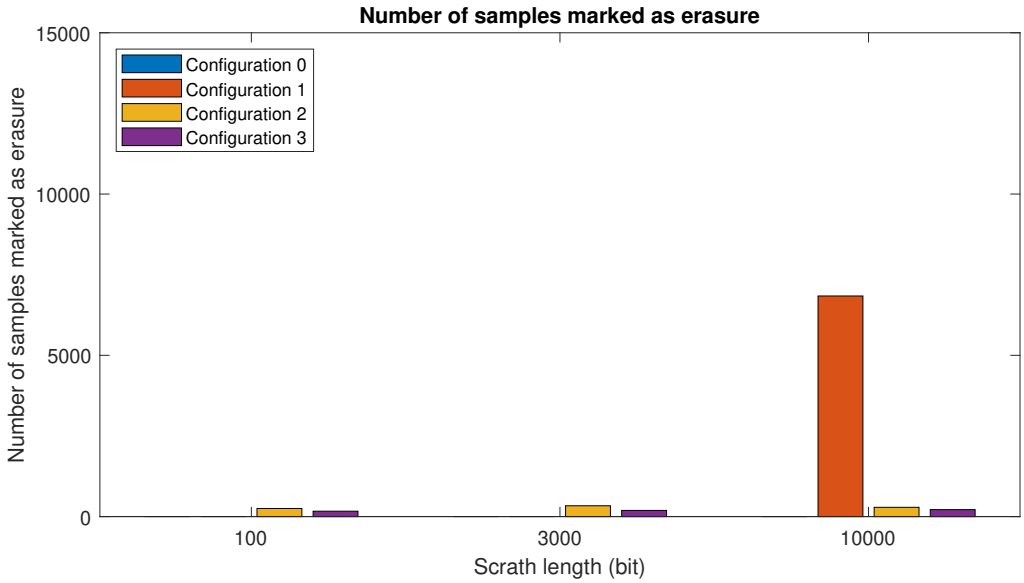
In the second graph indicating the number of undetected errors (Fig. 2b) the expectations are clearly visible: the CIRC decoder provides the best result with regards to detected errors. Configurations two and three perform about the same, but perform barely better than configuration zero.

Next up, there was looked at the random error decoding performance. For each of the three configurations the probability is plot that a sample is marked as an erasure (Fig. 3a), and the chance that an interpolation failed (Fig. 3b). Again, it can be seen that the first configuration which does the full CIRC decoding process detects more samples as erasures compared to the other two configurations.

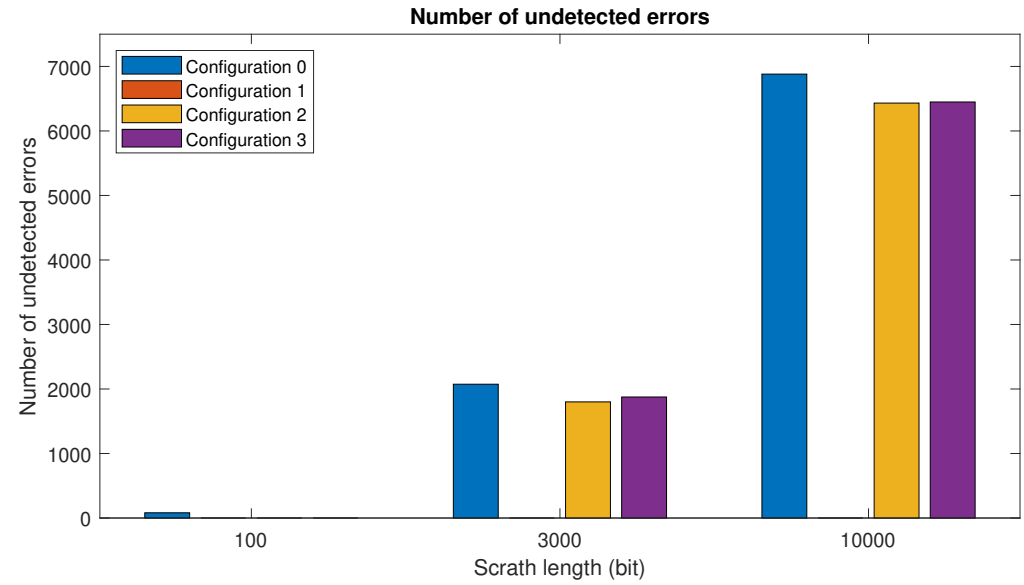
For the interpolation failure rate it can be seen that the CIRC decoder performs considerably better than configuration 2 for a low source error probability. On the other hand configuration 3 performs as good for a low error probability in the source as configuration 1, but it performs better for higher probability based on purely the interpolation failure rate. However it should be noted that configuration 1 in fact performs better than configuration 3 because it detects errors better or thus more errors (the samples marked as erasure influence the interpolation failure rate), which automatically results in a higher chance that the interpolation goes wrong if there are more interpolations needed. Based on the same reasoning configuration 2 also outperforms configuration 3 as it also detects more errors than configuration 3 and thus the interpolation failure rate is also higher compared to configuration 3. Configuration 1 and 2 have zero undetected errors for all tested source error probabilities, this is not the case for configuration 3.

Configuration 2 has in fact a better overal random error performance than configuration 1, it is able to correct random errors more effective. This is the case as it marks less erasures for higher source error probabilities than configuration 1 while having also zero undetected errors. On the other hand configuration 1 performs better for lower source error probabilities up to about 0.8 percent.

As mentioned earlier the decoding strategy of CIRC determines the performance for bursts and random errors as it was already shown the burst performance is quite good, the implemented strategey performs also quite well for random errors. However it performs not the best, but configuration 2 does.

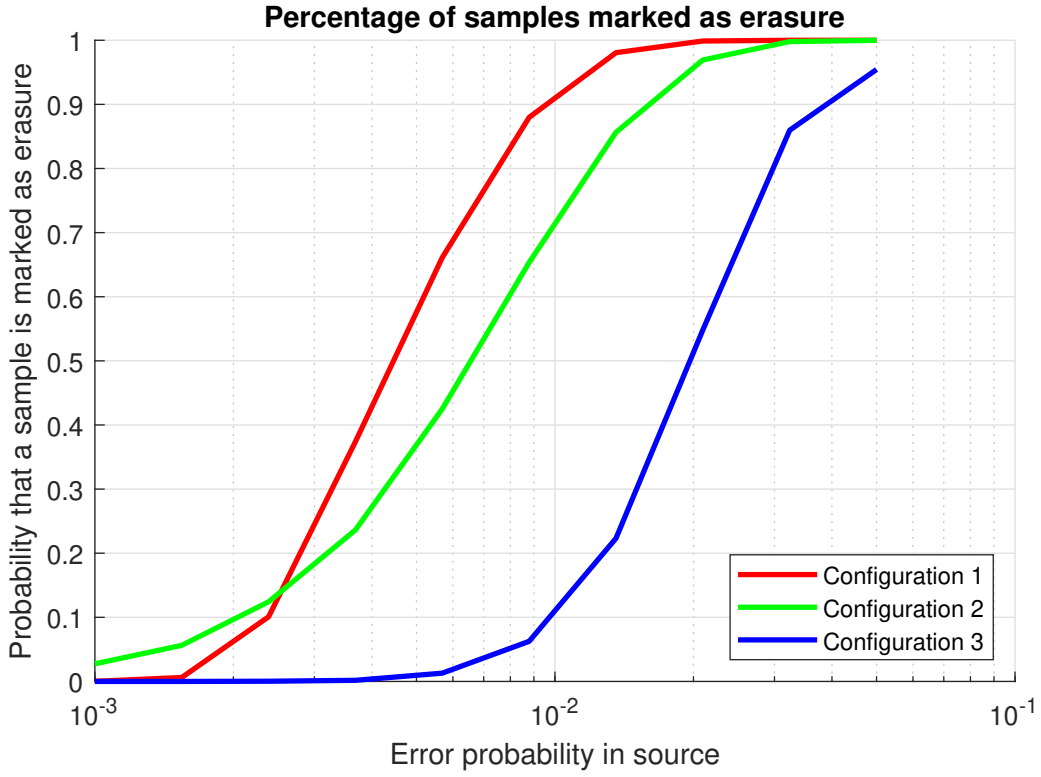

Number of samples marked as erasure
Configuration 0
Configuration 1
Configuration 2
Configuration 3
Number of samples marked as erasure
15000
10000
5000
0
100
3000
10000
Scrath length (bit)

(a)

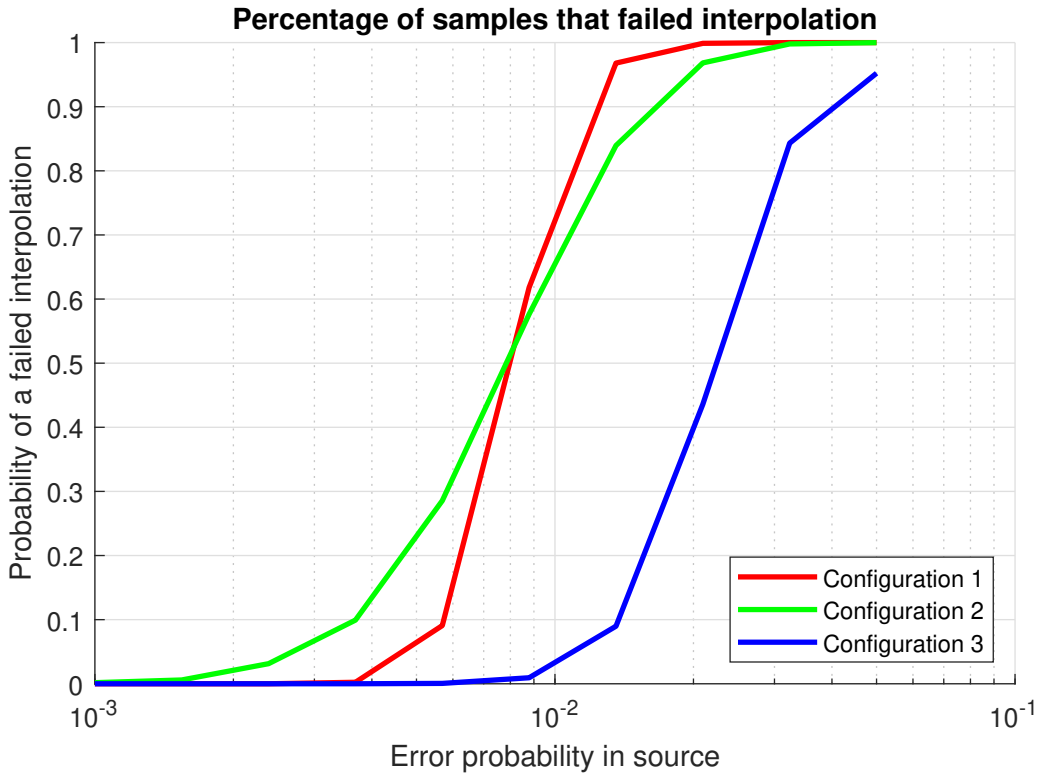

Number of undetected errors
Configuration 0
Configuration 1
Configuration 2
Configuration 3
Number of undetected errors
7000
6000
5000
4000
3000
2000
1000
0
100
3000
10000
Scrath length (bit)

(b)

Figure 2


Percentage of samples marked as erasure
Probability that a sample is marked as erasure
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10-3
10-2
10-1
Error probability in source
Configuration 1
Configuration 2
Configuration 3

(a)


Percentage of samples that failed interpolation
Probability of a failed interpolation
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10-3
10-2
10-1
Error probability in source
Configuration 1
Configuration 2
Configuration 3

(b)

Figure 3

5. **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?**

   - Long series of zeros should be avoided when encoding the data. The reason for this is that the decoder needs clock synchronization, which tends to drift when there is no change in the signal for a long period. This is why EFM modulation is needed. It transforms 8 bits from the CIRC encoder into 14 bits. These 14 bit blocks are chosen in such a way that they contain a small amount of transitions between 0 and 1, while avoiding a complete lack of transitions. These blocks are merged using 3 merging bits, of which two serve the purpose of avoiding successive 1's, while the third merging bit is used for clock synchronization. It can be 0 or 1, depending on its two neighbours which have the same value. The clock synchronization bit is always the opposite of its two neighbours.

   Codewords are also chosen in a manner that the edit distance between any two codewords is as large as possible. This helps the CIRC decoder decoding the signal since the decoded word (with or without corruption) can be mapped to the closest entry in the EFM modulation table. Furthermore the eye pattern received from demodulation gives also an indication on the quality of the demodulated data (certainty).

References

[1] H. Nakajima and H. Ogawa, *Compact Disc Technology*. Ohmsha, 1992.

[2] J. Maes, M. Vercammen, and L. Baert, *Digital Audio Technology: A Guide to CD, MiniDisc, SACD, DVD(A), MP3 and DAT*. Focal, 2001.

[3] K. Pohlmann, *The Compact Disc: A Handbook of Theory and Use*. Computer Music and Digital Audio Series, A-R Editions, 1989.

[4] M. Talbot-Smith, *Audio Engineer's Reference Book*. Taylor & Francis, 2013.

[5] S. Stan, *The CD-ROM Drive: A Brief System Description*. Springer US, 2013.

[6] S. Wicker and V. Bhargava, *Reed-Solomon Codes and Their Applications*. Wiley, 1999.

3 Division of labour

The part where was focused on per person.

- Common: Audio CD Subcode (5.1)
- Matthias Cami: Reed-Solomon code (5.2)
- Marijn De Kerpel: Reed-Solomon code (5.2)
- Zeno Dhaene: Audio encoding in CDs (5.3)
- Giani Ollivier: Audio encoding in CDs (5.3)