Project Information Theory

Bert Decrop, Kieran De Bruyn, Francis Valcke, Dylan Van Parys

April 2020

1 Task Division

Questions	
Audio CD Subcode	Kieran
RS Code: Encoding	Kieran
RS Code: Decoding	Kieran
Audio encoding in CD's Q1&2	Francis
Audio encoding in CD's Q3&4	Bert
Audio encoding in CD's Q5&6	Dylan
Code	
RSCode	Kieran
AudioCD logic	Francis
AudioCD simulation	Dylan

Table 1: Task Division

2 Audio CD Subcode

2.1 What is the datarate of the subcode in subcode blocks/s? And in bit/s?

From clause 17.3 of the CD audio standard, we learn that the subcode blocks hava a frequency of 75 Hz. Put differently, this means that the datarate in subcode blocks/s is 75 per second. Additionally, every subblock consists of 98 symbols of 8 bits, but the first two symbols are sync symbols. Thus we can conclude that the bitrate of the subcode is

$$75\frac{blocks}{s} * (98-2)\frac{symbols}{block} * 8\frac{bits}{symbol} = 57.6kbit/s \tag{1}$$

As a sanity check, in clause 19.1 of the standard, it is mentioned that the maximum available datarate for channels R to W is 43.2 kbit/s. Only six of the eight channels are used for this number, so this number should be 75% of the datarate we calculated above, which incidentally is true.

2.2 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.

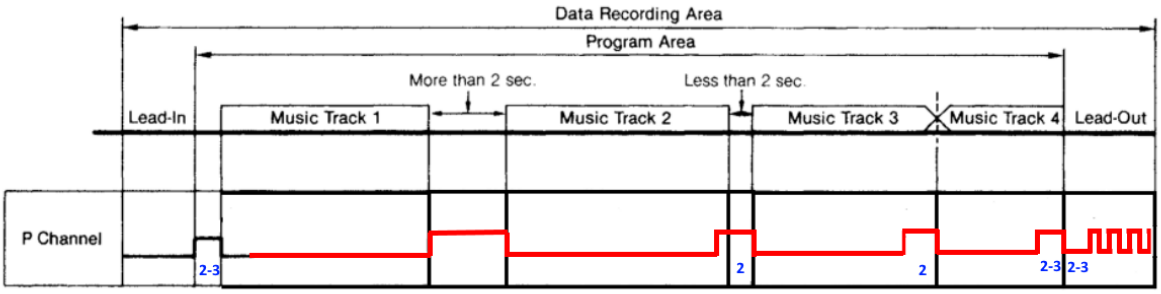

Figure 1: The completed figure of the P channel

In figure 1, the completed state of the channel P flag bit can be seen. The duration of the pulses is annotated in blue, in seconds. The frequency of the block wave at the end is 2 Hz.

2.3 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?

All this information is stored in the Q channel of the subcode, specifically during the lead in track. As mentioned before, the first 2 symbols of a subcode block are for synchronisation. In the bits in the Q channel of the symbol are assigned as in figure 2

S0, S1	CON-TROL	1	00	POINT	MIN	SEC	FRAME	ZERO	P MIN	P SEC	P FRAME	CRC
		ADR	TNO									

Figure 2: The allocation for the bits of the subcode in the Q channel during the lead in track

During the lead in track, when the value at POINT is A1, the value at PMIN is the track number of the last track, thus the CD player knows the number of tracks.
When the value at POINT is A2, the values at PMIN, PSEC and PFRAMES is the starting point of the lead out track, thus the CD player knows the total playing time of the audio. When the value at POINT is a number between 01 and 99, the values at PMIN, PSEC and PFRAMES point to the starting point of the track with that number. Thus, the CD player can compile a table of contents with the starting points of all the tracks on the CD. This data is protected in two ways. Firstly, this table of contents is played continually (and thus multiple times) during the lead in track. Secondly, the values in each line in the table of contents is repeated three times before moving on to the next line.

3 Reed-Solomon code

3.1 Encoding

3.1.1 Let α be a primitive element of GF(2^8). Consider the primitive polynomial given by: $p(D) = D8 + D4 + D3 + D2 + 1$. Show that p(D) is a primitive polynomial with root α for the field GF(2^8)

A primitive polynomial of GF(2^8) has to be irreducible over GF(2). This means that no non constant polynomial with highest degree 4 can cleanly divide in p(x). There are $2^5 - 2$ (0 and 1) $= 30$ non constant polynomials with highest degree of 4 and coefficients in GF(2). By dividing p(x) by them all and observing that all of these divisions have a nonzero remainder, we can conclude that p(x) is irreducible.
For p(x) to be a primitive polynomial of α, α has to be a root of p(x). if α is a root of p(x), then also $\alpha^(2^i)$ for $i = 1...7$ have to be roots of p(x). By evaluating

$$\prod_{i=1}^{7}(x - \alpha^{2^i}) \tag{2}$$

in α, a primitive element of GF(2^8), we obtain indeed p(x).

3.1.2 Construct the generator polynomials $g(x) = g_0 + g_1x + g_2x^2 + ... + g(n-k)x^(n-k)$ 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_1 is a (32, 28) code over GF(2^8). To obtain the value of m_0, we know that the check matrix of a Reed-Solomon code contains the roots of the generator polynomial to different powers. When we look at the check matrix of the C_1 code (figure 11 of the CD standard), we can see that these roots are 1, α, α^2, α^3. This means that m_0 is 0. By using the makeGenerator function we wrote, we obtain:

$$g(x) = x^4 + \alpha^{75}x^3 + \alpha^{249}x^2 + \alpha^{78}x^1 + \alpha^6 \tag{3}$$

3.1.3 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.

We know that

$$g(x)h(x) = x^{255} - 1 \tag{4}$$

So by using the deconv function in matlab, we can check that h(x) is of the form

$$h(x) = x^{251} + \alpha^{75}x^{250} + \alpha^{163}x^{249} + \alpha^{131}x^{248} + \alpha^{143}x^{247} + ...+ \\ \alpha^{125}x^4 + \alpha^{116}x^3 + \alpha^{151}x^2 + \alpha^{66}x^1 + \alpha^{249} \tag{5}$$

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

An MDS code is a code that reaches the Singleton bound. This bound states that the number of codewords in a binary block code of length n and minimal Hamming distance d is smaller than 2^{n-d+1}. Since there are 2^k codewords in a Reed Solomon code and the Singleton bound is reached, this means that.

$$2^k = 2^{n-d+1} \Leftrightarrow d = n - k + 1 \tag{6}$$

This means the minimal Hamming distance for this code is 5.

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

Because Reed-Solomon code is a non binary code. In a binary code, a symbol of k bits gets encoded into a codeword of n bits. In the case of a Reed-Solomon code in GF(2^8) an information word of k symbols gets encoded into a codeword of n symbols. Both have a redundancy ratio of (n-k)/k, but a burst error of 8 bits in a binary code translates to 8 bit errors that the code has to account for. In the case of a Reed Solomon code, an 8 bit burst error can translate to only a single erroneous symbol (or at most two) . Because burst errors are really common (e.g. scratches), we can conclude that -generally- a Reed Solomon code is more appropriate than a binary BCH code when it comes to data protection.

3.2 Decoding

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

The C_1 code uses 4 parity symbols and thus has an error correcting capability t of 2. This means that the code can always correct 2 bit errors. If the bit errors are distributed such that only two symbols are erroneous, up to 16 bit errors can be corrected.

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

The upper bound of the symbol error rate is given by equation 7, as found in the course notes

$$P_s \leq \sum_{i=t+1}^{n} \frac{i+t}{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{7}$$

with p the probability that the channel introduces a symbol error, t the error correcting capability and n the block length. An upper bound on the bit error rate is given by equation 8.

$$P_b \leq \sum_{j=1}^{K} \frac{j}{K} \binom{K}{j} \frac{1}{2^K - 1} \sum_{i=t+1}^{n} \frac{i+t}{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{8}$$

if we assume a BCH code in GF(2^K). Elaborating this, if we assume the worst case for both P_s and P_b, we find:

$$P_b = \frac{2^{K-1}}{2^K - 1} P_s \tag{9}$$

or in our case, where K = 8

$$P_b = \frac{128}{255} P_s \approx \frac{P_s}{2} \tag{10}$$

4 Audio encoding in CDs

4.1 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?

4.1.1 Delay of 2 frames

The first operation in the encoder is called **scrambling**. Scrambling is done by applying a 2 frame delay for the even samples and mixing up the connections to the C2 encoder. The delay of two frames serves to

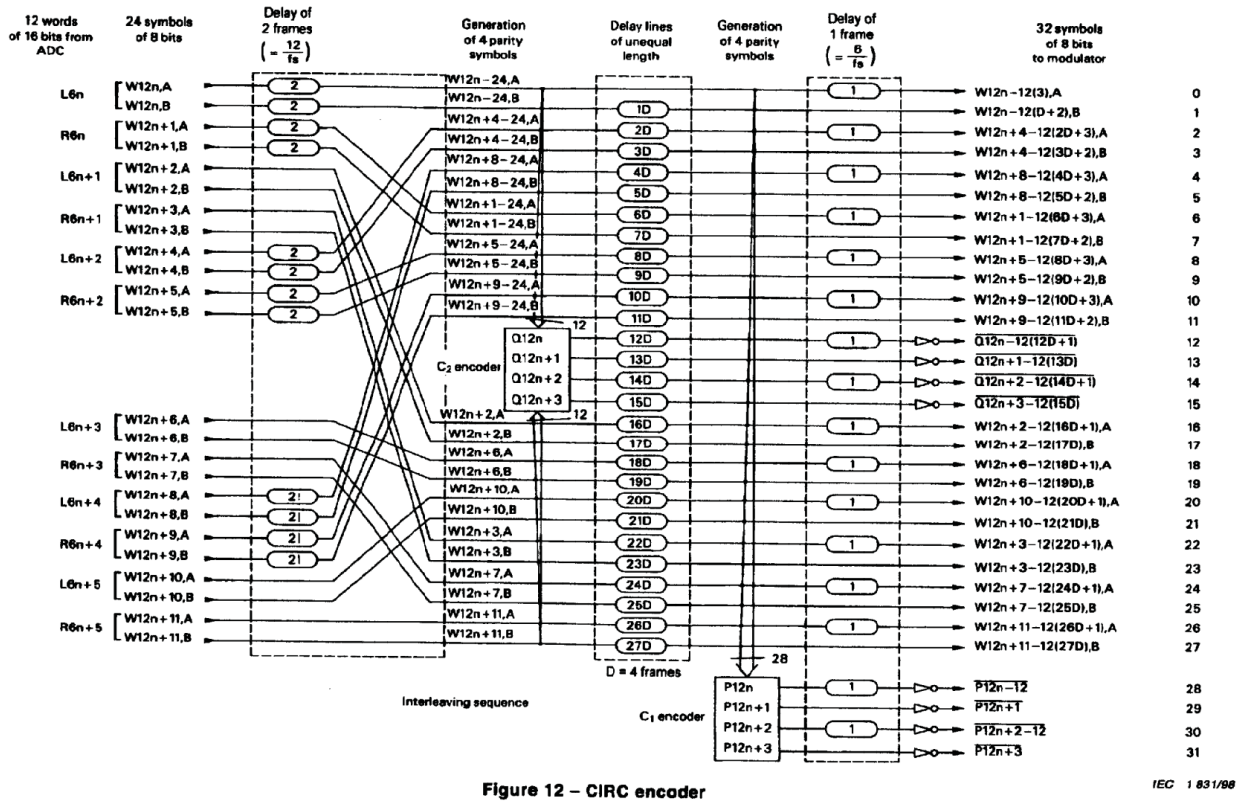

Figure 3: CIRC encoder from the standard

increase the chance of successful interpolation when a burst error occurs. By spreading the samples this way, the probability that either the even or uneven samples are intact increases for a given frame after a burst error occurs that affects up to two frames. The remaining good samples can then be used to approximate the erroneous samples.

4.1.2 Interleaving sequence

The interleaving sequence is also part of the **scrambling** phase. All of the even samples are separated from the uneven samples. Within these new groups the left audio channel is also separated from the right audio channel. This step is done with the same reasoning as the delay of two frames for the even samples, trying to put as much distance between left/right channel and even/uneven words as possible.

4.1.3 C_2 encoder

24 'Scrambled' symbols, each one byte in size, now serve as the input for the C_2 encoder. This step applies a (28,24) Reed-Solomon code to the the 24 symbols. The encoder generates four 8-bit parity symbols called Q words and inserts them between the even and uneven samples. This step provides the first layer of redundancy and is able to decode frames with up to 4 erroneous symbols.

4.1.4 Delay lines of unequal length

A delay of four frames is applied to each of the output lines of the C_2 encoder except the first line. These delays distribute the C_2 code over a total of 109 frames. The goal here is again to protect against burst errors. As the C_2 code protects against a total of 4 erroneous symbols, and these symbols are spread over 16 frames, this approach can handle burst errors of up to 16 frames in size.

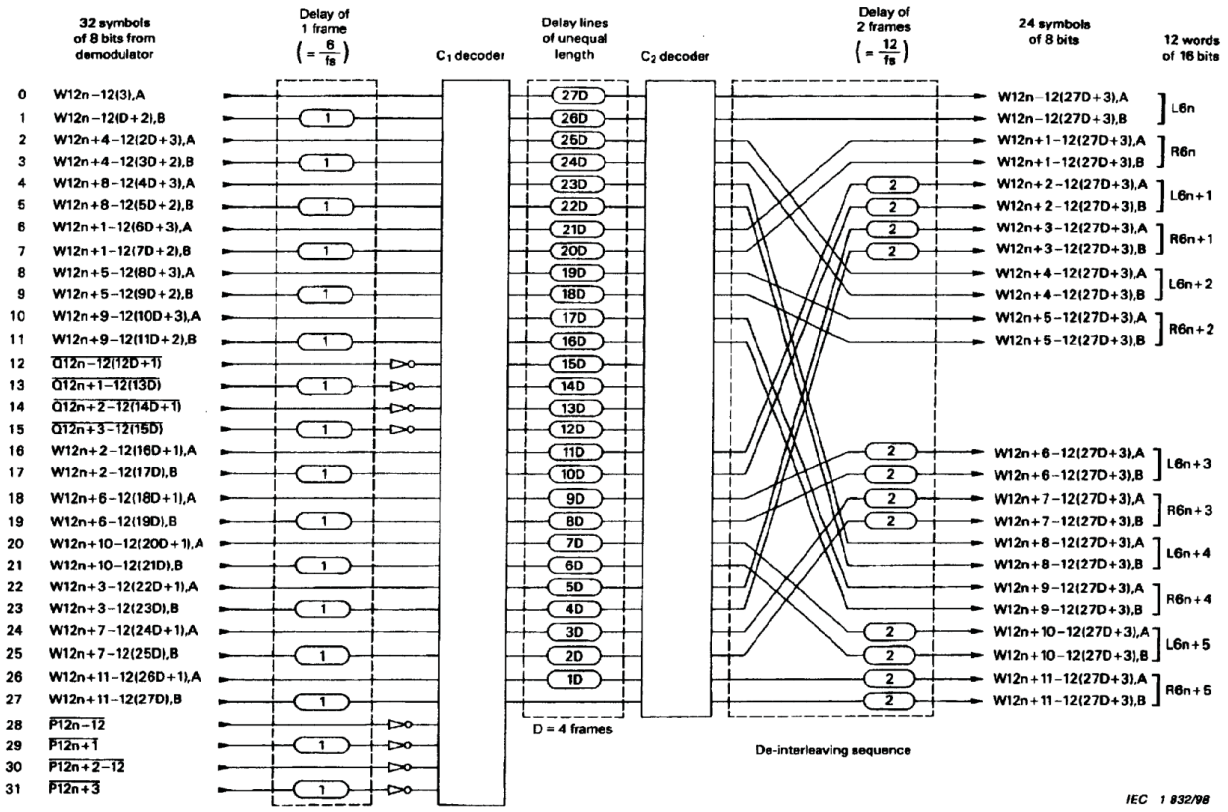

Figure 13 – CIRC decoder

Figure 4: CIRC decoder from the standard

4.1.5 C_1 encoder

The next step is to further encode the 28 symbols, each from different frames, using a (32,28) Reed-Solomon code in the C_1 encoder. The goal of this code is to protect against single symbol errors. During decoding, the C1 decoder keeps track of the number of detected errors, and passes this information to the C2 decoder.

4.1.6 Delay of 1 frame

The last step in the encoder, and also the last interleaving stage, is to apply a 1 frame delay on all even symbols of the 32 output symbols of the C_1 encoder. The reasoning behind these 1 frame delays is to protect against random errors. This protection is due to the output words being spread out over two frames. This prevents random errors from disrupting more than one symbol in one word, even if two adjacent symbols in one frame are erroneous. The parity symbols also get inverted in this step. This provides the parity symbols with zero data. This assists data readout during areas with muted audio program. [1][2]

4.2 Complete the functions in the AudioCD class. Decoding has to be performed as described in Section 4 of the assignment. The number of detected errors is an output of the comm.RSDecoder step() function, a -1 signals an uncorrectable error.

See AudioCD.m file

4.3 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 sample rate is 44.1 kHz.) Compare this with simulation results of your Matlab code and explain.

As said before, if the C2 code protects against a total of 4 erroneous symbols, and these symbols are spread over 16 frames, then this approach can handle burst errors of up to 16 frames in size. This means that the maximum burst length in bit is 16 frames * 32 symbols/frame * 8 bits/symbol = 4096 bit. This is however a very simple model, and we are probably glossing over some details.

To calculate the scratch width we first calculate the bit density. With an audio sample rate of 44.1kHz, every second 44100 samples are read from the disk. Each sample is 32 bit long. The CIRC encoder will add 8 symbols per block of 24 symbols. This corresponds with a code rate of R = 24/32. With this we can calculate the number of bits per second that is read from the disk: we have 44100 samples/second * 32 bit/sample * 32/24 = 1881600 bits/second.

We also know that the scanning velocity of the laser is 1.3 m/s. Out of these results, follows the bit density: (1881600 bits/second) / (1.3 m/second) = 1447385 bits/m. From this, we can easily calculate the scratch width as 4096 bits / (1447385 bits/m) = 0.00283 m = 2.83 mm.

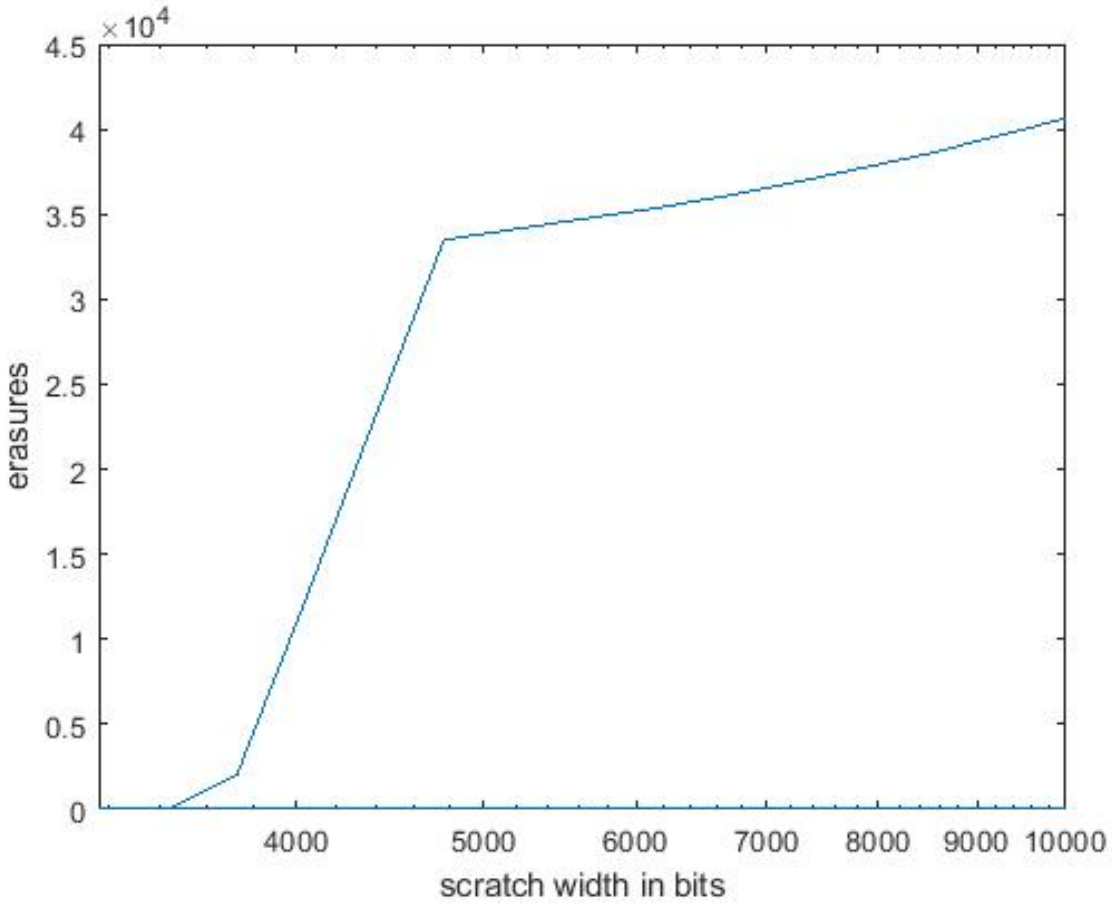

Figure 5: Our matlab result for this is 3690 bits, so this is less than we expect, but that we anticipated this because of our simplified model.

Scratch test results			
Configuration	100 bits	3000 bits	10000 bits
0	(0, 0, 0.0002)	(0, 0, 0.0050)	(0, 0, 0.0165)
1	(0, 0, 0)	(0, 0, 0)	(0.0487, 0.0479, 0)
2	(0.0006, 0.0004, 0)	(0.0008, 0, 0.0045)	(0.0007, 0, 0.0160)
3	(0.0004, 0, 0)	(0.0006, 0, 0.0046)	(0.0006, 0, 0.0160)

Table 2: Results of scratch tests. Entries have the form (e, i, u) which is a tuple of the erasure flag possibility, interpolation failure possibility and the undetected error probability

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

During the encoding process, the even and uneven sample are separated. So when one sample contains an error, it is likely that the adjoining samples are unharmed. This means that if there occurs an error after incorrect decoding, the wrongly bytes can be interpolated using the 8 surrounding bytes. When increasing the sample frequency, the error occurs within a shorter period of time, making the error less audible.

4.5 Compare the performance of the 4 configurations (see AudioCD class) under the following circumstances:

4.5.1 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).

From the scratch test results visible in table 2 one can quickly induce that burst error correction is where CIRC encoding proves its worth. It vastly outperforms the other 2 configuration where no interleaving is performed. These two other configurations are hardly any better than configuration 0, leading one to conclude that interleaving indeed greatly improves scratch resistance for CD encoding, as one would have expected.

4.5.2 Random bit errors with a bit error probability p in an equidistant logarithmic range between 0.05 and 0.001. 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!

In figures 6 and 7 correspondingly erasure probability and interpolation failure probability are shown for a logarithmic range of bit error probabilities and the 3 possible configurations. These plots indicate that doing double RS encoding leads to both a higher erasure flagging and a higher interpolation failure probability than single RS(32, 24) (config 1 vs 3). This means that there is a cost to RS concatenation for random bit errors.

4.5.3 Conclusion

Although CIRC performed worst of all configurations on random bit errors (although this difference diminished once the bit error probability got small enough), the fact is that it truly outperforms the other configurations for the correction of burst errors. In modern day applications a typical random bit error rate would be around the order of magnitude of 10^{-5}[2] meaning that the benefit of the performance on burst errors far outweighs the worse performance on random bit errors. This can thus lead us to conclude that CIRC outperforms the other configuration in modern day real world applications.

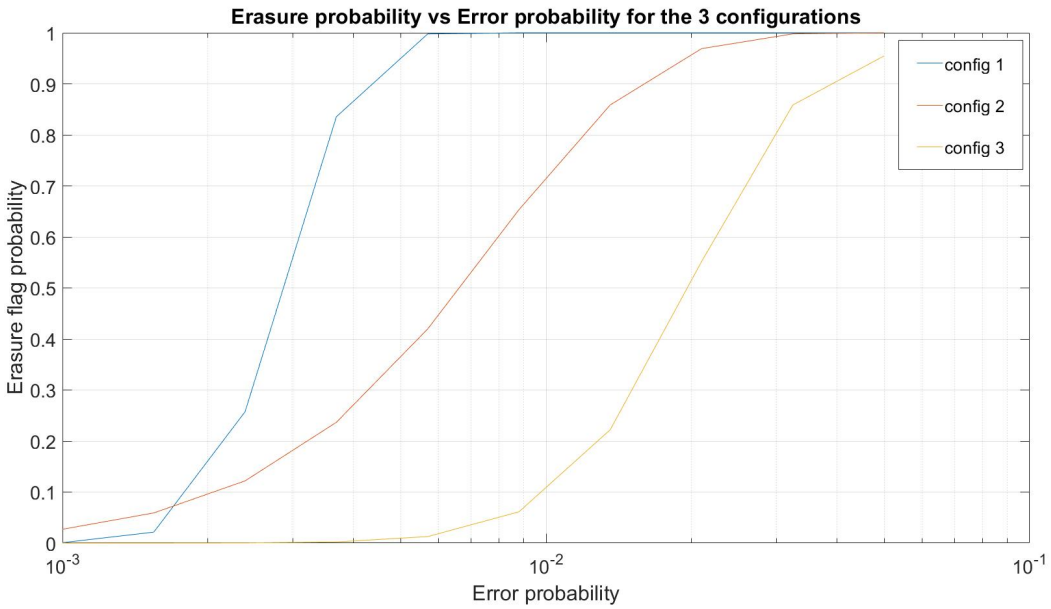

Figure 6: Error probability vs Erasure probability for 3 configurations

4.6 After CIRC encoding, the data bits 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?

EFM is an encoding scheme in which (as the name suggests) 8 bit blocks are mapped on to 14 bit blocks of data. EFM is a type of run-length limited code which in simple terms means that there are limits to the runlength of a sequence of zeroes (both and upper and lower bound). This is useful to improve for example signal properties during transmission (a lower bound on the sequence of zeroes could prohibit rapid changes in frequency in FM for example) or in the case of CD provide useful properties for the sequence of pits and lands. In EFM the lower bound is 2 and the upper bound is 10. For this (2, 10)-RLL code and the 14 bits available this means that there are 267 14 bit symbols available which fulfil the (2, 10) lower and upper bound. This is sufficient to cover the 8 bit symbol space and thus translation can be done using a LUT. EFM is thus used in CD encoding/decoding to optimise the pit/land sequence of laser etching since 1's are encoded as a transition from a pit to land or land to pit. Bounds on the sequence of zeroes thus induce bounds on the distance between these transitions.[3]

At the EFM demodulation component of the system the RLL validity of the symbols can be checked. If the RLL properties should not hold true, this information could be passed on to the first decoder of CIRC to increase error correction capabilities (for example maybe the EFM demodulator will already translate it to the most probable original 8 bits, however the first decoder should be informed about this).

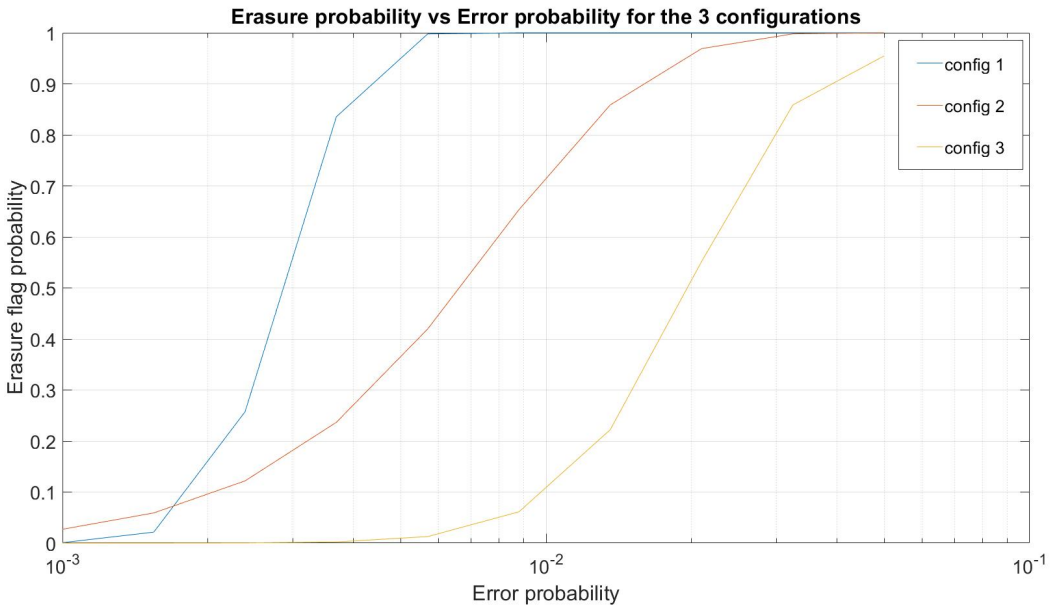

Figure 7: Error probability vs Interpolation failure probability for 3 configurations

References

[1] LuC. Theunissen LuC. Baert and Guido Vergult (Eds.). *Digital Audio and Compact Disc Technology*. Newnes, 2 sub edition, 1992.

[2] Ken C. Pohlmann. *The Compact Disc Handbook*. Oxford University Press, 1989.

[3] HIROSHI OGAWA and Kees Schouhamer Immink. Efm the modulation method for the compact disc digital audio system. 1982.