Chapter 5

Mathematics and Audio Compact Discs

5

5

5 Mathematics and Audio Compact Discs

In this chapter we give a short description of the mathematical background behind the technology used for compact discs. Since we are dealing with problems arising from a real-word application, we must adapt our assumptions to this particular situation.

In the first section we present a short introduction to digital audio transmission. Some facts about Fourier Series and Fourier Transforms are collected. We describe sampling and filtering of signals from a mathematical point of view, explain analog digital conversion, dither, pulse code modulation and, finally, we prove Shannon's Sampling Theorem, which gives an interpolation formula which expresses the value $f(x)$ of a signal at any time x in terms of its values $f(ns)$ at the discrete points ns for $n \in \mathbb{Z}$, $s > 0$.

First of all, as far as the error-correction in connection with compact discs is concerned, we should notice that errors are not uniformly distributed random errors. In fact, errors tend to occur in bursts, for instance due to manufacturing errors when the compact disc was produced, or due to surface errors arising from scratches or fingerprints while handling the disc. Such a burst error is actually a string of errors happening within a short period of time. In the third section we will investigate how to detect and correct burst errors from a general point of view.

We have already mentioned that in fact two linear codes are applied for the encoding process in the production of a compact disc. This method is known as interleaving of two codes, which is thoroughly described in the third section. This interleaving process allows us to correct burst errors. Since we are using two codes, one of them can be used for error detection. We already know that the error detection rate of a code is larger than its error correction rate. In case the second code detects an error in the received vector, it marks all the corresponding components sent to the first code as erasures (these are errors where the position of the error but not its value is known) and the first code can be applied to correct both errors and erasures. For this reason, we investigate the correction of erasures with linear codes in the second section. In particular, we present an algorithm for correcting erasures with BCH-codes.

In the third section we meet product codes (cf. 2.3.15) again. We describe how they can be used for correction of transmission errors and erasures. Especially products of cyclic codes are analyzed.

Finally, in the last two sections we present all the important facts about the methods used for error detection and error correction both in audio compact discs and in CD-ROM. In particular, the CIRC encoding and decoding is de-

scribed in detail. Furthermore, we explain how to use interpolation in order to deal with errors which can not be eliminated by CIRC. Also the pit/land structure of a track on a CD and the EFM are mentioned.

The second and third section describe independently from the application in compact discs some interesting coding and decoding methods. The first section is mainly devoted to digital audio, the fourth to a detailed description of error detection and correction in audio compact discs. The fifth section gives a short overview how the CD-ROM standard extends the standard of digital audio discs.

5.1

5.1 Fourier Transform, Shannon's Sampling Theorem

In this section we explain how an acoustic signal is transformed into digital data. Usually this process is called *sampling*. In our setting the acoustic signal is a mapping $f : \mathbb{R} \to \mathbb{R}$, where $f(x)$ is the sound pressure at the time x. When dealing with audio data, it is important to analyze which frequencies occur in the signal. For this reason, we give a short introduction to the theory of Fourier Transforms, which describes how to express a signal f as a sum of functions $x \mapsto e^{2\pi i \xi x}$ for $\xi \in \mathbb{R}$. We limit our presentation to the essential facts, since a more complete discussion would require detailed knowledge of the theory of Lebesgue integration, which is beyond the scope of this text. Our presentation of the Fourier analysis is based on [64] and [30]. Technical details about the compact disc system are taken from Pohlmann's book [164]. Further facts about acoustics and audio engineering are taken from [164], [202] and [175].

Let $I \subseteq \mathbb{R}$ be an interval and let $p \geq 1$ be a real number. (The reader should take care that in this section p does not indicate a prime number!) We denote the set of all measurable functions $f : I \to \mathbb{C}$ for which $|f(x)|^p$ is integrable on I by $L^p(I)$. Then $L^p(I)$ together with the norm

$$\|f\|_p := \left(\int_I |f(x)|^p \, dx \right)^{1/p}, \qquad f \in L^p(I),$$

is a complete normed vector space (cf. Exercise 5.1.2 and [64, 15.2.3 Proposition]).

In this setting, $\|f\|_p = 0$ if and only if $f = 0$ almost everywhere on I, in other words, $f(x) = 0$ for all $x \in I \setminus M$, where M is a subset of I of measure 0. The set $\mathcal{N} := \left\{ f \in L^p(I) \mid \|f\|_p = 0 \right\}$ is a subspace of $L^p(I)$. In order to be more precise, instead of $L^p(I)$ we should actually consider the factor space $L^p(I)/\mathcal{N}$.

The set $L^2(I)$ is also equipped with an inner product defined by

$$\langle f, g\rangle = \int_I \overline{f(x)} g(x)\, dx, \qquad f, g \in L^2(I),$$

where $\overline{f(x)}$ means the complex conjugate of $f(x)$. It is easy to show that $L^2(I)$ is a Hermitian inner product space and that

$$\|f\|_2 = \sqrt{\langle f, f\rangle}, \qquad f \in L^2(I).$$

Consequently, $L^2(I)$ is a Hilbert space.

Different notions of convergence are considered. Let $(f_n)_{n\in\mathbb{N}}$ be a sequence with $f_n \in L^p(I)$ and let $f \in L^p(I)$. We say that $(f_n)_{n\in\mathbb{N}}$ *converges uniformly* to f if

$$\lim_{n\to\infty} \sup_{x\in I} |f_n(x) - f(x)| = 0,$$

which can also be expressed as

$$\forall\, \epsilon > 0,\ \exists\, N_\epsilon,\ \forall\, n > N_\epsilon,\ \forall\, x \in I\ :\ |f_n(x) - f(x)| < \epsilon.$$

The sequence $(f_n)_{n\in\mathbb{N}}$ *converges pointwise* to f if

$$\forall\, x \in I,\ \forall\, \epsilon > 0,\ \exists\, N_{\epsilon,x},\ \forall\, n > N_{\epsilon,x}\ :\ |f_n(x) - f(x)| < \epsilon.$$

The sequence $(f_n)_{n\in\mathbb{N}}$ converges to f in $L^p(I)$ if

$$\lim_{n\to\infty} \|f_n - f\|_p = 0.$$

Convergence in $L^1(I)$ is also called *mean convergence*, whereas convergence in $L^2(I)$ is known as *mean quadratic convergence* or *convergence "in energy"*. If $(f_n)_{n\in\mathbb{N}}$ converges uniformly to f in $L^1(I)$ and I is a finite interval, then integration and limit can be interchanged so that

$$\lim_{n\to\infty} \int_I f_n(x)\, dx = \int_I f(x)\, dx.$$

The *support* of a function $f : \mathbb{R} \to \mathbb{C}$ is the closure of the set of all elements $x \in \mathbb{R}$ for which $f(x) \neq 0$:

$$\operatorname{supp}(f) = \operatorname{cl}\{x \in \mathbb{R} \mid f(x) \neq 0\}.$$

A function $f : \mathbb{R} \to \mathbb{C}$ has period $a > 0$ if $f(x) = f(x + a)$ for all $x \in \mathbb{R}$. Especially for $p = 2$ and $I = (0, a)$, we consider the set

$$L^2_{\text{per}}(0, a) := \left\{ f : \mathbb{R} \to \mathbb{C} \;\middle|\; f \text{ has period } a \text{ and } \int_0^a |f(x)|^2\, dx < \infty \right\}$$

which, together with the norm

$$\|f\|_2 := \left(\int_0^a |f(x)|^2\, dx \right)^{1/2},$$

is a normed vector space.

At first we approximate functions $f \in L^2_{\mathrm{per}}(0,a)$ by

5.1.1 **Trigonometric polynomials** A *trigonometric polynomial* of degree N in $L^2_{\mathrm{per}}(0,a)$, $a > 0$, is an expression of the form

$$p(x) := \sum_{n=-N}^{N} c_n e^{2\pi i n x/a}$$

with $c_n \in \mathbb{C}$. We denote by T_N the set of all trigonometric polynomials of degree N. Let e_n be the trigonometric polynomial $x \mapsto e^{2\pi i n x/a}$. Then

$$\langle e_n, e_m \rangle = \begin{cases} a & \text{if } n = m, \\ 0 & \text{if } n \neq m. \end{cases} \quad \text{and} \quad \|e_n\|_2 = \sqrt{a}.$$

Thus, the set $\{e_n \mid -N \leq n \leq N\}$ forms an orthogonal basis of T_N, and T_N is a $(2N+1)$-dimensional space. Moreover, for $p = \sum_{n=-N}^{N} c_n e_n \in T_N$ we have $\langle e_n, p \rangle = c_n \|e_n\|_2^2 = c_n a$, whence

$$c_n = \frac{1}{a}\langle e_n, p \rangle = \frac{1}{a}\int_0^a p(x) e^{-2\pi i n x/a}\, dx, \qquad -N \leq n \leq N.$$

Using the fact that

$$e^{ix} = \cos(x) + i \sin(x), \qquad x \in \mathbb{R},$$

any $p \in T_N$ can be expressed as

$$\begin{aligned} p(x) &= c_0 + \sum_{n=1}^{N} \left(c_n e^{2\pi i n x/a} + c_{-n} e^{-2\pi i n x/a} \right) \\ &= c_0 + \sum_{n=1}^{N} \big((c_n + c_{-n}) \cos(2\pi n x/a) + i(c_n - c_{-n}) \sin(2\pi n x/a) \big). \end{aligned}$$

If we put $a_n := c_n + c_{-n}$ and $b_n := c_n - c_{-n}$ for $n \geq 0$, we obtain

$$p(x) = \frac{a_0}{2} + \sum_{n=1}^{N} \big(a_n \cos(2\pi n x/a) + i b_n \sin(2\pi n x/a) \big), \qquad x \in \mathbb{R},$$

with

$$a_n = \frac{2}{a}\int_0^a p(x) \cos(2\pi n x/a)\, dx$$

and

$$b_n = \frac{2}{a}\int_0^a p(x) \sin(2\pi n x/a)\, dx. \qquad \square$$

Theorem [64, page 30] *Let N be a positive integer, and assume that $a > 0$. For $f \in L^2_{\mathrm{per}}(0,a)$ there exists exactly one trigonometric polynomial* 5.1.2

$$f_N(x) := \sum_{n=-N}^{N} c_n e^{2\pi i n x/a}$$

such that $\|f - f_N\|_2 = \min\{\|f - p\|_2 \mid p \in T_N\}$. The coefficients of f_N are given by

$$c_n = c_n(f) = \frac{1}{a}\int_0^a f(x) e^{-2\pi i n x/a}\, dx, \qquad -N \le n \le N. \tag{5.1.3}$$

For all $N \in \mathbb{N}$ Bessel's inequality

$$\sum_{n=-N}^{N} |c_n|^2 \le \frac{1}{a}\int_0^a |f(x)|^2\, dx$$

holds true. Consequently,

$$\sum_{n=-\infty}^{\infty} |c_n|^2 < +\infty$$

and $c_n = c_n(f) \to 0$ as $|n| \to \infty$. □

Fourier Series Deeper methods from the theory of the Lebesgue integral show [64, 4.3.1 Theorem] that for $f \in L^2_{\mathrm{per}}(0,a)$, $a > 0$, 5.1.4

$$\|f_N - f\|_2 \to 0 \ \text{ as } \ N \to \infty.$$

In other words,

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} c_n e^{2\pi i n x/a} \\ &= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(2\pi n x/a) + i b_n \sin(2\pi n x/a)\right) \end{aligned} \tag{5.1.5}$$

almost everywhere in $\mathbb{R}$, since this is an equality in $L^2_{\mathrm{per}}(0,a)$. From this representation of f we obtain *Parseval's equality*

$$\sum_{n=-\infty}^{\infty} |c_n|^2 = \frac{1}{a}\int_0^a |f(x)|^2\, dx.$$

The coefficients c_n are called *Fourier coefficients* of f. The right hand side of 5.1.5 is the *Fourier series* of f. □

In addition, we take for granted the following result on Fourier series:

Theorem [64, 5.3.1 Theorem] *Assume that f has period $a > 0$, is continuous on $\mathbb{R}$, differentiable on $[0,a]$ with exception of possibly a finite number of points, and that f' is piecewise continuous. Then the Fourier series of f converges uniformly to f on $\mathbb{R}$.* 5.1.6

□

5.1.7 **Example** We have just explained how to approximate a periodic function with trigonometric polynomials. For example consider the periodic function f with period $a = 2\pi$ defined on $I = [-\pi, \pi)$ by

$$f(t) := \begin{cases} -1 & \text{if } -\pi \le t < 0, \\ +1 & \text{if } 0 \le t < \pi. \end{cases}$$

The three approximations for $N = 1, 3, 5$ are given by

$$\begin{aligned} f_1(t) &= \frac{4}{\pi}\sin(t) \\ f_3(t) &= \frac{4}{\pi}(\sin(t) + \frac{1}{3}\sin(3t)) \\ f_5(t) &= \frac{4}{\pi}(\sin(t) + \frac{1}{3}\sin(3t) + \frac{1}{5}\sin(5t)). \end{aligned}$$

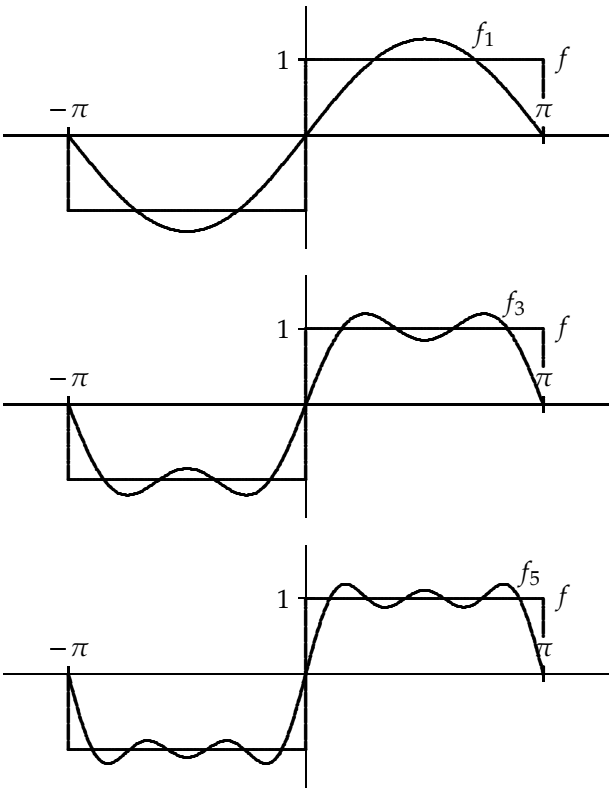


Fig. 5.1 Approximation by trigonometric polynomials

◇

The Discrete Fourier Transform Let f be a periodic function with period $a > 0$ and let N be a positive integer. Let $x_k := ka/N$, $k \in N$, be an evenly spaced subdivision of the interval $[0, a]$. Assume that the values $y_k := f(x_k)$, $k \in N$, are known. Furthermore we assume that the Fourier series of f converges pointwise to f and that **5.1.8**

$$f(x) = \frac{1}{2}(f(x^+) + f(x^-))$$

holds true at points x of discontinuity. Here we have $f(x^+) := \lim_{t \to x,\, t>x} f(t)$ and $f(x^-) := \lim_{t \to x,\, t<x} f(t)$. In order to simplify our notation, we assume that N is odd. For $-(N-1)/2 \le n \le (N-1)/2$, the coefficients of the Fourier expansion of f can be approximated by using the trapezoid formula as

$$c'_n = \frac{1}{a} \sum_{k \in N} y_k \frac{a}{N} e^{-2\pi i n k a/(Na)} = \frac{1}{N} \sum_{k \in N} y_k e^{-2\pi i n k/N}.$$

It can be shown that these c'_n are the Fourier coefficients of the trigonometric polynomial $p \in T_{(N-1)/2}$ which interpolates f at the points ka/N for $k \in N$. If we put

$$Y_n := \begin{cases} c'_n & \text{if } 0 \le n \le (N-1)/2, \\ c'_{n-N} & \text{if } (N+1)/2 \le n \le N-1, \end{cases}$$

then we obtain the two equivalent formulae of the *Discrete Fourier Transform*

$$\begin{aligned} y_k &= \sum_{n \in N} Y_n e^{2\pi i n k/N}, \qquad k \in N, \\ Y_n &= \sum_{k \in N} y_k e^{-2\pi i k n/N}, \qquad n \in N. \end{aligned}$$

□

The Fourier Transform [64, 17.1.3 Theorem] *The* Fourier Transform $\hat{f}$ *of* $f \in L^1(\mathbb{R})$ *is defined by* **5.1.9**

$$\hat{f}(\xi) := \int_{\mathbb{R}} f(x) e^{-2\pi i \xi x}\, dx.$$

The Fourier Transform $\hat{f}$ *of* $f \in L^1(\mathbb{R})$ *is continuous and bounded with*

$$\sup_{x \in \mathbb{R}} |\hat{f}(x)| \le \|f\|_1 \quad \textit{and} \quad \lim_{x \to \pm\infty} \hat{f}(x) = 0.$$

□

The term 2π in the exponent is often omitted in the definition of the Fourier Transform. In this case, the integral is multiplied by the normalization factor $1/\sqrt{2\pi}$.

Comparing $\hat{f}$ with the Fourier coefficients 5.1.3, we deduce that the Fourier Transform allows one to pass from the time domain of the signal f to the frequency domain. The value $|\hat{f}(\xi)|$ is considered to represent the amplitude of the frequency ξ in the signal f.

Some properties of $\hat{f}$ are collected in Exercise 5.1.4 and in the following

5.1.10 Lemma [64, 17.2.1 Theorem]

1. *If $x^k f \in L^1(\mathbb{R})$ for $0 \le k \le n$, then $\hat{f}$ is n times differentiable and the k-th derivative of $\hat{f}$ is*
$$\hat{f}^{(k)}(\xi) = \hat{f}_k(\xi), \qquad 1 \le k \le n,$$
for $f_k(x) := (-2\pi i x)^k f(x)$.
2. *If $f \in L^1(\mathbb{R})$ is n times continuously differentiable and all the derivatives $f^{(k)}$, $1 \le k \le n$, are in $L^1(\mathbb{R})$, then*
$$\widehat{f^{(k)}}(\xi) = (2\pi i \xi)^k \hat{f}(\xi), \qquad 1 \le k \le n.$$
3. *If $f \in L^1(\mathbb{R})$ has bounded support, then $\hat{f}$ is infinitely many times differentiable. We also write $\hat{f} \in C^\infty(\mathbb{R})$.* □

It is not true in general that the Fourier Transform of $f \in L^1(\mathbb{R})$ is again in $L^1(\mathbb{R})$ as the following example shows.

5.1.11 Example Consider, for instance, the function
$$f(x) := \begin{cases} e^{-x} & \text{if } x \ge 0, \\ 0 & \text{if } x < 0, \end{cases}$$
then
$$\begin{aligned} \hat{f}(\xi) &= \int_0^\infty e^{-2\pi i \xi x - x}\, dx = \lim_{R\to\infty} \int_0^R e^{-(2\pi i \xi + 1)x}\, dx \\ &= \lim_{R\to\infty} -\frac{e^{-(2\pi i \xi + 1)R}}{2\pi i \xi + 1} + \frac{1}{2\pi i \xi + 1} = \frac{1}{2\pi i \xi + 1}, \end{aligned}$$
which is not integrable. ◇

Therefore, it is interesting to determine under which conditions $\hat{f} \in L^1(\mathbb{R})$. The next lemma gives sufficient conditions on f.

5.1.12 Lemma [64, 18.1.2 Theorem] *If f is twice continuously differentiable and if f, f' and f'' are in $L^1(\mathbb{R})$, then $\hat{f} \in L^1(\mathbb{R})$.* □

5.1.13 The Inverse Fourier Transform [64, 18.1.1 Theorem] *If both f and $\hat{f}$ belong to $L^1(\mathbb{R})$, then*
$$f(x) = \int_{\mathbb{R}} \hat{f}(\xi) e^{2\pi i \xi x}\, d\xi$$
for all points x, where f is continuous. This integral is called the Inverse Fourier Transform *of f.* □

A function $f \in L^1(\mathbb{R})$ is called *band limited* if the support of its Fourier Transform $\hat{f}$ is bounded. In other words, if there exists a limiting value $\lambda_c > 0$ such that $\operatorname{supp}(\hat{f}) \subseteq [-\lambda_c, \lambda_c]$, which means that in the signal f no frequencies greater than λ_c occur.

Consequence of the Paley–Wiener Theorem [64, 31.5.2 Theorem and p. 360] **5.1.14**
If $f \neq 0$ is band limited, then $f \in C^\infty(\mathbb{R})$. Hence, f vanishes on no interval of positive length. The assumption that $f \neq 0$ is band limited implies that f is analytic with $\operatorname{supp}(f) = \mathbb{R}$.

If f has bounded support (which means bounded in time), then f cannot be band limited since $\hat{f}$ is a C^∞-function. □

Example Consider the band limited function f with **5.1.15**

$$\hat{f}(\xi) = \begin{cases} 1 & \text{if } |\xi| \leq \lambda_c, \\ 0 & \text{if } |\xi| > \lambda_c. \end{cases}$$

Since f is band limited, it is analytic, thus continuous. By the Inverse Fourier Transform we get

$$\begin{aligned} f(x) &= \int_{\mathbb{R}} \hat{f}(\xi) e^{2\pi i \xi x} \, d\xi = \int_{-\lambda_c}^{\lambda_c} e^{2\pi i \xi x} \, d\xi = \frac{e^{2\pi i \xi x}}{2\pi i x}\Big|_{-\lambda_c}^{\lambda_c} \\ &= \frac{1}{\pi x} \cdot \frac{e^{2\pi i \lambda_c x} - e^{-2\pi i \lambda_c x}}{2i} = 2\lambda_c \frac{\sin(2\pi\lambda_c x)}{2\pi\lambda_c x}. \end{aligned}$$

This is closely related to the *cardinal sine function* $\operatorname{sinc} : \mathbb{R} \to \mathbb{R}$ which is defined by

$$\operatorname{sinc}(x) := \begin{cases} \sin(x)/x & \text{if } x \neq 0, \\ 1 & \text{if } x = 0. \end{cases}$$

It has zeros at $k\pi$ for $k \in \mathbb{Z} \setminus \{0\}$. The set of local extrema of $\operatorname{sinc}(x)$ corresponds to its intersections with the cosine function $\cos(x)$. The main peak of $\operatorname{sinc}(x)$ has width 2π whereas all other peaks have width π.

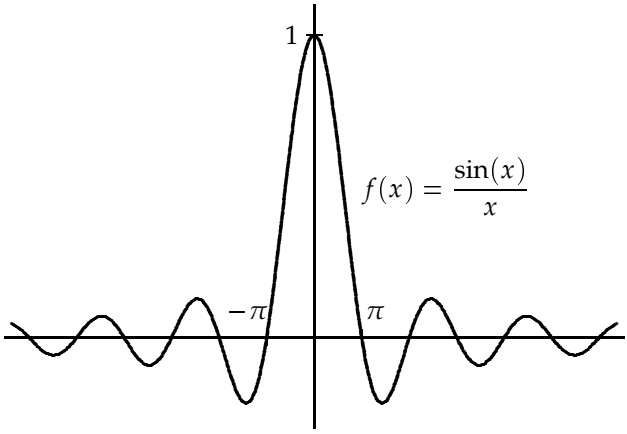
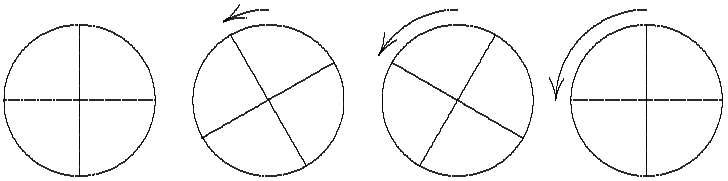


Fig. 5.2 The cardinal sine function

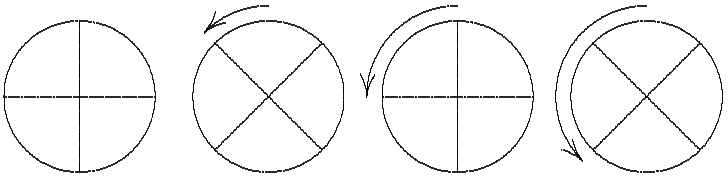
If we consider $\operatorname{sinc}(2\pi\lambda_c x)$ as a function of λ_c we notice that as λ_c increases, the width of the peaks decreases. ◇

5.1.16 **Sampling** Assume that $f\colon \mathbb{R} \to \mathbb{C}$ is a signal and $s > 0$ is a real number. *Sampling* f every s time units means to replace f by the sequence $(f(ns))_{n\in\mathbb{Z}}$. The *sampling rate* or *sampling frequency* is given by $1/s$.

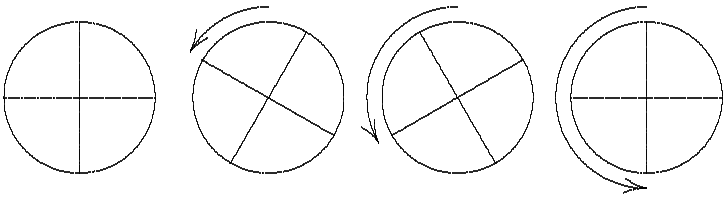
For instance, when watching a film, we are presented a sequence of 15 to 20 pictures per second. From this sampling we get the impression of smoothly moving pictures. But when seeing a turning wheel equipped with spokes, we realize that sometimes the wheel seems to rotate in the right direction, sometimes it is standing still, and sometimes it is turning in the converse direction. This is caused by the sampling of the pictures. Consider, for instance, a turning wheel with 4 spokes, which has a periodic movement with period $\pi/2$. If it is turning slowly enough, i.e. if we take more than two pictures within one period we get the right impression. For example, consider a wheel which is turning anti-clockwise. If it is turning from one sample to the next by $\pi/6$, we get:



Turning between two samplings by $\pi/4$ produces only two different pictures so that we have the impression of a standing wheel which has 8 spokes.



If the wheel is turning even faster, for instance by $\pi/3$ from one sampling to the next, then we have the impression that the wheel is rotating in the opposite direction.



If the wheel is turning even faster we will realize similar phenomena as just described. From this example we deduce that on the one hand some information of the original signal is lost by sampling. But we will see that the loss of information can be neglected for practical purposes, provided the sampling

rate is sufficiently high. On the other hand, digital processing offers many advantages. For instance, real time processing of the data can be done between the arrival of two consecutive samples. In addition, sampling and digital processing also allow us to compress the signal in such a way that it can later be reconstructed without loss of essential data. This technique is known as source coding. For further details we refer to [81].

Shannon's formula is an interpolation formula which expresses the value $f(x)$ of a signal at any time x in terms of its values $f(ns)$ at the discrete points ns, $n \in \mathbb{Z}$, $s > 0$. This theorem is attributed to C.E. Shannon, but was already discovered earlier by E.T. Whittaker [200], J.M. Whittaker [201], Kotel'nikov [114], or Nyquist [160]. For a historical background see also [103] or [141].

Shannon's Sampling Theorem [179] *Assume that $f \in L^1(\mathbb{R})$ is a band limited signal with $\operatorname{supp}(\hat{f}) \subseteq [-\lambda_c, \lambda_c]$. Consider a sampling rate $1/s \geq 2\lambda_c$. If the Fourier Transform $\hat{f}$ is piecewise continuously differentiable on the closed interval $[-\lambda_c, \lambda_c]$, then* **5.1.17**

$$f(x) = \sum_{n=-\infty}^{\infty} f(ns) \frac{\sin \pi(x/s - n)}{\pi(x/s - n)}, \qquad x \in \mathbb{R}. \tag{5.1.18}$$

Proof: We extend the restriction of $\hat{f}$ to the interval $[-1/(2s), 1/(2s))$, which contains $[-\lambda_c, \lambda_c)$, to a continuous, periodic function $\tilde{f}$ with period $a = 1/s$ defined by

$$\tilde{f}(x + n/s) := \hat{f}(x) \text{ for } x \in [-1/(2s), 1/(2s)),\ n \in \mathbb{Z}.$$

Since $\tilde{f}$ is continuous and piecewise continuously differentiable, according to 5.1.6 the Fourier series of $\tilde{f}$ converges uniformly. Consequently, for each $\xi \in [-1/(2s), 1/(2s)) = [-a/2, a/2)$ we have

$$\hat{f}(\xi) = \sum_{n=-\infty}^{\infty} c_n e^{2\pi i n \xi / a},$$

where

$$c_n = \frac{1}{a} \int_{-a/2}^{a/2} \hat{f}(\xi) e^{-2\pi i n \xi / a}\, d\xi = s \int_{\mathbb{R}} \hat{f}(\xi) e^{2\pi i \xi \cdot (-ns)}\, d\xi = s f(-ns)$$

by 5.1.13. Another application of the Fourier Inversion formula yields

$$f(x) = \int_{\mathbb{R}} \hat{f}(\xi) e^{2\pi i \xi x}\, d\xi = \int_{-1/(2s)}^{1/(2s)} \sum_{n=-\infty}^{\infty} c_n e^{2\pi i n \xi s} e^{2\pi i \xi x}\, d\xi.$$

Since the Fourier series converges uniformly, we are allowed to interchange the sequence of integration and summation, obtaining

$$f(x) \;=\; s \sum_{n=-\infty}^{\infty} f(-ns) \int_{-1/(2s)}^{1/(2s)} e^{2\pi i \xi (ns + x)}\, d\xi$$

$$= \sum_{n=-\infty}^{\infty} sf(ns) \int_{-1/(2s)}^{1/(2s)} e^{2\pi i\xi(x-ns)}\, d\xi = \sum_{n=-\infty}^{\infty} f(ns) \frac{\sin \pi(x/s - n)}{\pi(x/s - n)}$$

by an application of 5.1.15. □

The assumptions of the last theorem are satisfied, for instance, in the situation when $f \in L^1(\mathbb{R})$ is band limited and the function $x \mapsto xf(x)$ is also integrable, since then by 5.1.10 $\hat{f}$ is continuously differentiable.

In order to describe a band limited signal f with $\operatorname{supp}(\hat{f}) \subseteq [-\lambda_c, \lambda_c]$ by a sampling with rate $1/s$ we must choose s so that $1/s \geq 2\lambda_c$. This critical value $2\lambda_c$ is called the *Nyquist rate*. Obviously, the largest frequency which can be described properly using a sampling rate $1/s$ is $1/(2s)$. This frequency is also known as the *Nyquist frequency*.

The following picture (see [30, page 62]) shows that if the frequency of a signal is too high for a given sampling rate, then this frequency is not properly described by the sampling. For instance, taking 10 samples per time unit yields $s = 1/10$ and a sampling rate of 10. In this situation we cannot distinguish between the two functions $f_1(x) = \sin(8\pi x)$ and $f_2(x) = \sin(28\pi x)$. The first one produces exactly 4 sine waves per time unit the second one 14 sine waves. If the time unit is a second, then we have a sampling rate of 10 Hz and two frequencies of 4 Hz and 14 Hz, respectively. Since $10 > 2 \cdot 4$, the sampling rate is high enough for describing f_1. According to 5.1.17, the frequency of f_2 is too high for the given sampling rate, so f_2 cannot be properly reconstructed from the sampling.

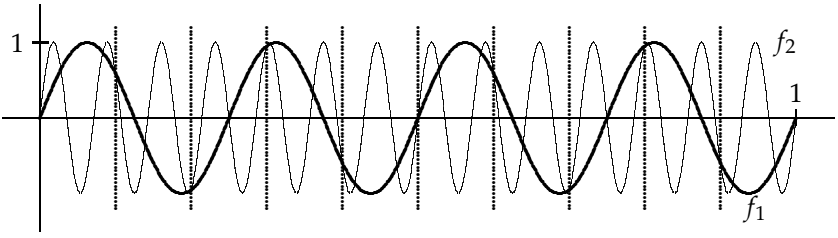


Fig. 5.3 Aliasing

The two functions f_1 and f_2 coincide for all x of the form $k/10$, $k \in \mathbb{Z}$. Hence, sampling f_2 at a sampling rate of 10 produces f_1. This phenomenon is called *aliasing*. In general, if $1/s$ is the sampling frequency and $f > 1/(2s)$ is sampled, then new frequencies f' appear, where $f' = n/s \pm f$ for $n \in \mathbb{Z}$.

In signal processing, $L^2(\mathbb{R})$ models the space of signals which are functions of a continuous variable (usually time) and which have finite energy. So far the Fourier Transform has only been defined for functions in $L^1(\mathbb{R})$, and $L^2(\mathbb{R})$ is not included in $L^1(\mathbb{R})$. In [64, Lesson 22] it is explained how to extend the

Fourier Transform in a natural way to $L^2(\mathbb{R})$. The Fourier Transform in $L^2(\mathbb{R})$ has the major advantage that $\hat{f} \in L^2(\mathbb{R})$ whenever $f \in L^2(\mathbb{R})$.

Filters In order to apply Shannon's Sampling Theorem, we must have a band limited signal in order to determine the correct sampling rate $1/s$. Thus, the natural signal, which is usually not band limited, must be filtered. **5.1.19**

Assume that f is a time limited, piecewise continuously differentiable signal, then by the Inverse Fourier Transform

$$f(x) = \int_{\mathbb{R}} \hat{f}(\xi) e^{2\pi i \xi x} \, d\xi, \qquad x \in \mathbb{R}.$$

Usually a *filter* is described by its *transfer function* $\hat{h}$ for some $h \in L^2(\mathbb{R})$. Applying the filter with transfer function $\hat{h}$ to the signal f should produce (cf. [30, page 202ff]) the signal

$$\tilde{f}(x) = \int_{\mathbb{R}} \hat{f}(\xi)\hat{h}(\xi) e^{2\pi i \xi x} \, d\xi, \qquad x \in \mathbb{R}.$$

The amplitudes $|\hat{f}(\xi)|$ of the original signal are multiplied by the amplitudes $|\hat{h}(\xi)|$ and the phases $\arg(\hat{f}(\xi))$ are changed by adding the phase $\arg(\hat{h}(\xi))$. A complex number $c \neq 0$ may be presented as $c = x + iy$ with $x, y \in \mathbb{R}$, or $c = |c|e^{i \arg(c)}$, where $|c| = \sqrt{x^2 + y^2}$ is a positive real number called the complex modulus of c and $\arg(c)$ is a real number in the interval $(-\pi, \pi]$. It is called the argument of c and can be computed as

$$\arg(x + iy) = \begin{cases} \tan^{-1}(y/x) & \text{if } x \neq 0, \\ -\pi/2 & \text{if } x = 0 \text{ and } y < 0, \\ \pi/2 & \text{if } x = 0 \text{ and } y > 0. \end{cases}$$

An *ideal low-pass filter* suppresses all frequencies greater than a limiting frequency λ_c. It produces a signal $\tilde{f}$ which may be delayed by $x_0 \geq 0$, where the frequencies in $[-\lambda_c, \lambda_c]$ are not changed and all the frequencies outside this interval are canceled. Thus, the transfer function of an ideal low-pass filter is given by

$$\hat{h}(\xi) = \begin{cases} A_0 e^{-2\pi i \xi x_0} & \text{if } |\xi| \leq \lambda_c, \\ 0 & \text{if } |\xi| > \lambda_c, \end{cases}$$

with $A_0 > 0$, and the filtered signal is

$$\tilde{f}(x + x_0) = A_0 \int_{-\lambda_c}^{\lambda_c} \hat{f}(\xi) e^{2\pi i \xi x} \, d\xi, \qquad x \in \mathbb{R}.$$

Moreover, we derive that $\tilde{f}$ is band limited, infinitely many times differentiable, but it is not time limited, i.e. its support is the whole real line. From 5.1.15 we deduce that

$$h(x) = 2A_0\lambda_c \frac{\sin(2\pi\lambda_c(x - x_0))}{2\pi\lambda_c(x - x_0)}.$$

A filter is called *realizable* if for each signal f and each $x_0 \in \mathbb{R}$ the implication

$$\Big(f(x) = 0 \ \text{ for all } \ x < x_0\Big) \Longrightarrow \Big(\tilde{f}(x) = 0 \ \text{ for all } \ x < x_0\Big)$$

holds true. From the explicit form of h given above we derive that an ideal low-pass filter is actually not realizable. The best we can expect is to find realizable filters whose transfer functions approximate the transfer function of an ideal filter. For instance, *Butterworth filters* are realizable approximations of ideal low-pass filters.

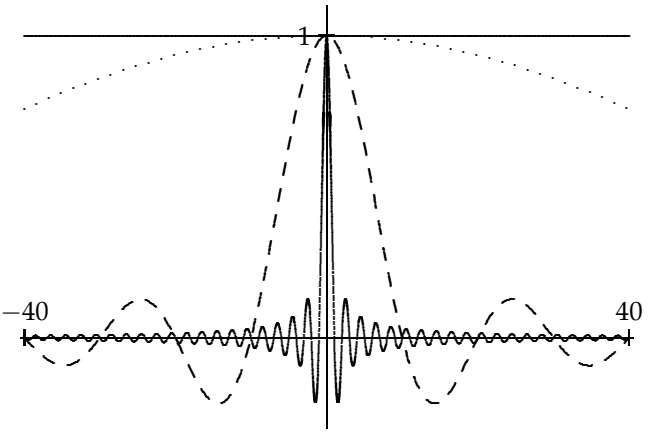
5.1.20 **Impulse** Similarly as in 5.1.15 we compute the Fourier Transform of the time limited constant signal

$$\delta_\epsilon(x) := \begin{cases} 1 & \text{if } |x| \le \epsilon/2, \\ 0 & \text{if } |x| > \epsilon/2, \end{cases} \qquad \epsilon > 0.$$

For $\epsilon = 1/m$, $m \in \mathbb{N}^*$, we obtain

$$m\widehat{\delta_{1/m}}(\xi) = m \int_{\mathbb{R}} \delta_{1/m}(x) e^{-2\pi i \xi x}\, dx = \operatorname{sinc}\Big(\frac{\pi\xi}{m}\Big).$$

In the following picture this function is plotted for $m = 1$ (solid), $m = 10$ (dashed), $m = 100$ (dotted), and $m = 1000$ (solid).



Given a limiting frequency λ_c we can find an integer m large enough so that $m\widehat{\delta_{1/m}}$ is approximately equal to 1 on the whole interval $[-\lambda_c, \lambda_c]$. Equivalently, if the signal $\delta_{1/m}$ is short enough, we call it an *impulse*, then $m\widehat{\delta_{1/m}}$ is approximately equal to 1 on this interval. This fact can be used for the

---

5.1.21 **Reconstruction of the original signal from a sampled signal** In order to reconstruct the original signal from a sampled signal we cannot use the formula

5.1.18 of the Sampling Theorem, since computing $f(x)$ for $x \in \mathbb{R}$ would require the knowledge of the sampled data $f(ns)$ for all $n \in \mathbb{Z}$, where f is a function whose support is the whole real line.

If we assume that we know only finitely many sampled values, for instance $f(ns)$ for $-M \le n \le N$, then we can use the sequence of impulses

$$f_{\epsilon,s}(x) := \sum_{n=-M}^{N} f(ns)\delta_\epsilon(x - ns)$$

as an input signal to an ideal low-pass filter described in 5.1.19 with limiting frequency $\lambda_c = 1/(2s)$. If we choose ϵ small enough so that $(1/\epsilon)\widehat{\delta_\epsilon}$ is approximately 1 on $[-\lambda_c, \lambda_c]$, then the output of the ideal low-pass filter is approximately

$$\begin{aligned}
\widetilde{f_{\epsilon,s}}(x) &= \int_{-\lambda_c}^{\lambda_c} \widehat{f_{\epsilon,s}}(\xi)\hat{h}(\xi)e^{2\pi i\xi x}\,d\xi \\
&= A_0 \sum_{n=-M}^{N} f(ns) \int_{-\lambda_c}^{\lambda_c} \widehat{\delta_\epsilon}(\xi)e^{2\pi i\xi(x-(x_0+ns))}\,d\xi \\
&\approx 2\lambda_c A_0 \epsilon \sum_{n=-M}^{N} f(ns)\frac{\sin\pi((x-x_0)/s-n)}{\pi((x-x_0)/s-n)}, \qquad x \in \mathbb{R}.
\end{aligned}$$

Up to the factor $2\lambda_c A_0\epsilon$ and the delay by x_0 this is an approximation of f, converging to $2\lambda_c A_0 \epsilon f(x - x_0)$ if $N, M \to \infty$.

Digital audio transmission As an input we have the time limited audio signal f. Using an approximation of an ideal low-pass filter with maximum frequency λ_c we obtain a band limited signal $\tilde{f}$. Using a sampling rate $1/s \ge 2\lambda_c$ we obtain the sequence of samples $\tilde{f}(ns)$ for $-M \le n \le N$. An analog-digital converter replaces each sample $\tilde{f}(ns)$ by a digital codeword. These codewords are sent through the channel. The receiver produces a sequence of samples $F(ns)$ for $0 \le n \le N + M$ by using a digital-analog converter. Applying an ideal low-pass filter we smooth the sequence of impulses $\sum_{n=0}^{N+M} F(ns)\delta_\epsilon(x - ns)$ and obtain an approximation of the input signal. **5.1.22**

Sound pressure and decibels *Sound power* or *acoustic power* P is a measure of sonic energy per time unit. It is measured in watts, abbreviated by W. The ratio of two sound powers P_1 and P_0 is usually described in decibels by **5.1.23**

$$\mathrm{dB} = 10\log_{10}(P_1/P_0).$$

If no second power P_0 is indicated, then the reference sound power in air is usually taken to be $10^{-12}\,\mathrm{W} = 0\,\mathrm{dB}$. The decibel is a dimensionless "unit".

Since most audio engineers work with voltages and since power is proportional to the square of voltage, we also have

$$\mathrm{dB} = 10\log_{10}(V_1^2/V_0^2) = 20\log_{10}(V_1/V_0).$$

The smallest change of sound power detectable for the human ear corresponds to one decibel. A change of three decibel is noticeable to most people. 10 dB seems to be approximately twice as loud.

The *sound intensity* is defined as the sound power P per unit area. The usual context is the measurement of sound intensity in the air at a listener's location. It is expressed in $\mathrm{W/m^2}$. *Sound pressure* p or *acoustic pressure* is the measurement in Pascal ($\mathrm{Pa} = \mathrm{N/m^2}$) of the average sound wave pressure variations as the sound wave passes by a fixed point. We have

$$p = F/A,$$

where p is the sound pressure in Pascal Pa, F is the force measured in Newton N, and A is the area measured in $\mathrm{m^2}$.

Sound is usually measured by microphones and they respond with voltage approximately proportional to the sound pressure p. Thus, we also have

$$\mathrm{dB} = 20\log_{10}(V_1/V_0) = 20\log_{10}(p_1/p_0).$$

Unless specified otherwise, the reference level for air is chosen as 20 micro Pascal. This is about the limit of sensitivity of the human ear in the most sensitive range of frequency.

If two sound intensities satisfy $p_1 = 2p_0$, then the signal of p_1 is twice as loud as the signal of p_0. This yields a ratio of $20\log_{10}(2) \approx 6.02059913$ dB.

5.1.24 **Analog-digital converter** So far we have been dealing with sound sampling and we have seen that when the sampling rate is high enough it is possible to reconstruct totally a band limited signal from the discrete samples. At each sampling the amplitude of a sound signal is measured. The amplitude is an analog signal which takes infinitely many values. When these values are stored in a digital system, only a finite number of discrete values or steps can be represented by digital numbers of finite length. So, for each measured amplitude value a digital value must be found which approximates the original value as good as possible. *Quantization* is the technique of approximating an analog amplitude by discrete numbers. (For more details see [164, pages 27ff].) By convention, the signal is attached with a sign. This means, that half of the digital values are used for positive amplitudes and half of them for 0 and negative amplitudes. When producing a compact disc each sampling is quantized by a 16 bit word. Hence, there are totally $2^{16} = 65\,536$ values which can be

used to approximate all occurring amplitudes. In general, at each sampling small approximation errors occur. These errors are not bigger than half of a step between two consecutive digital values.

The number of bits used to represent a single sampling value, i.e. the word length, determines the resolution of the quantization.

The *signal-to-noise ratio*, often abbreviated by S/N, is an engineering term for the ratio between the magnitude of a signal and the magnitude of background noise. It is often expressed in terms of the decibel scale. If the incoming signal strength in microvolts is V_s, and the noise level, also in microvolts, is V_n, then the signal-to-noise ratio in decibels is given by the formula

$$\mathrm{S/N(dB)} = 20\log_{10}(V_s/V_n).$$

Signal-to-noise ratios are closely related to the concept of dynamic range. Whereas dynamic range measures the ratio between noise and the greatest undistorted signal on a channel, S/N measures the ratio between noise and an arbitrary signal on the channel, not necessarily the most powerful signal possible. Because of this, measuring signal-to-noise ratios requires the selection of a representative or reference signal. In audio engineering, this reference signal is usually a sine wave, i.e. a plain tone, at a recognized and standardized magnitude (cf. [202]).

In connection with digital audio, the noise is the error signal caused by the quantization of the signal. The *signal-to-error ratio*, in short S/E ratio, is the number of available digital values divided by the maximal quantization error. Above we have just seen that the quantization error is not greater than $1/2$ bit. For instance, the S/E ratio of 16 bit audio is $2^{17} = 131\,072$. Every added bit doubles this ratio and also the number of possible digital values.

Usually the amplitude of a signal is attached with a sign. Therefore, using a quantization with n bits, the range of the digital signal lies between $-(2^{n-1}-1)$ and 2^{n-1}. When a voltage amplitude of $V_{\max}$ is used, then a quantization step is given by

$$\Delta = \frac{V_{\max}}{2^{n-1}}.$$

In terms of decibels, the S/E ratio depends on the word length n as

$$\mathrm{S/E(dB)} = 20\log_{10}\frac{V_{\max}}{\Delta/2} = 20\log_{10}(2^n) = 20n\log_{10}(2) \approx 6.02n.$$

This formula yields for 16-bit audio a S/E ratio of about 96dB. For more details see [156].

Pulse code modulation The amplitude of the signal is stored in *pulse code modulation*, PCM (cf. [164, pages 35ff]). This means that the analog signal is **5.1.25**

represented by the sequence of binary values produced by the analog-digital converter. In PCM format the quantized value of every sampling is stored. In 16-bit audio format we can represent an audio signal by $2^{16} = 65\,536$ discrete levels. As mentioned above, the signal is attached with a sign. Hence, there are 32 768 binary values for representing positive amplitudes and 32 767 binary values for negative amplitudes.

PCM was invented by A.H. Reeves in 1939 (American Patents 2272070, 1942-2, see [156]) and was analyzed and developed as a modulation system from the point of view of communication theory by C.E. Shannon [180]. For 16-bit audio with sampling rate 44 100 Hz the demand on the storage device and speed of the transmission channel is 88 200 Bytes/sec. (Usually 8 bits are combined into one byte, thus each sampling produces 2 bytes of audio information. For an audio compact disc the left and right channel are sampled separately, which yields the above-mentioned number of audio bytes per second for each channel.) This is a "brute force" approach, which is not the most effective way of using the storage device and transmission channel.

"Sampling and quantization are the two fundamental design elements for audio digitization." ([164])

5.1.26 **Dither** As a matter of fact, even though the quantization error of 16-bit audio is quite small, it is obvious that when the signal amplitude decreases, the relative error increases. If the signal level is approximately as big as the signal difference corresponding to two consecutive digital values, then these errors could be audible. In other words, quantization not only loses information, but also causes unexpected problems. The following picture shows a low level sine signal and its quantization which is a square wave. As a square wave, according to 5.1.7, its trigonometric approximation is rich of odd harmonics extending far beyond the sampling frequency. (Harmonics are sine waves the frequencies of which are positive integer multiples of the wave with the smallest frequency.) For this reason, as we already know, low-pass filters must be used in order to obtain a band-limited output signal.

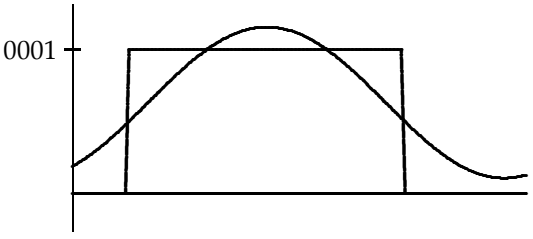


Fig. 5.4 Quantization of a low level sine signal

After quantization of a low amplitude signal the resulting signal differs extremely from the original signal. This effect is also known as *granulation noise*. In high amplitude audio signals this effect is usually not audible.

Granulation noise can be removed by, surprisingly, adding small amounts of analog noise. This noise is called *dither* (cf. [164, pages 32ff]).

Adding low level noise to the previous signal we obtain:

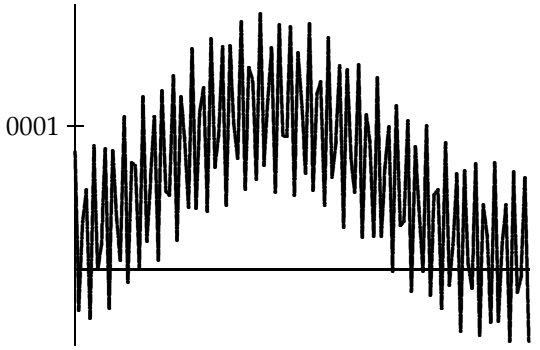


Fig. 5.5 Adding dither to a low level sine signal

After quantization it looks like:

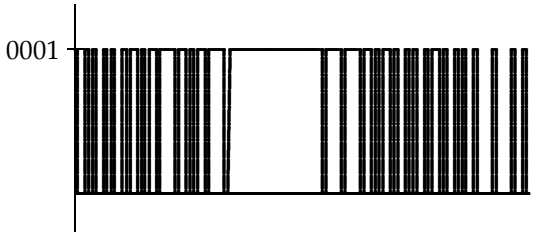


Fig. 5.6 Quantization of the low level sine signal together with dither

Finally, taking the average amplitude of this quantized signal over a certain period of time we obtain

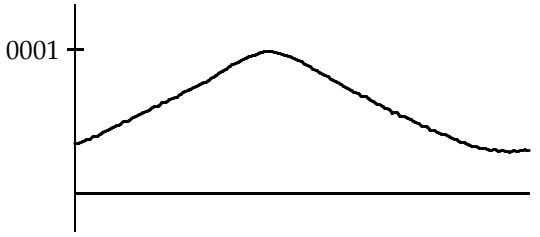


Fig. 5.7 The average amplitude of the quantized low level sine signal with added dither

which is quite similar to the original signal.

Since human beings are unable to hear frequencies which are beyond about 20 000 Hz, in the process of storing audio data on a compact disc all frequencies which are higher than 20 000 Hz are filtered out first. According to the Sampling Theorem, the sampling rate must be at least 40 000 samples per second in order to describe a band limited signal of maximal 20 000 Hz. For audio compact discs, 44 100 samples per second are taken both for the left and the right stereo channel. Each sample value is represented by 16 bits, this is a binary vector of length 16. The choice of these two parameters guarantees that the fidelity of the compact disc system is comparable to the best analog systems. For historical reasons the sampling rate was fixed to 44 100 Hz. Namely, early digital tape recorders used video cassette recorders for storage. Caused by the video refreshing rate and number of pixels per screen used in different video systems, they were able to store the audio information produced by a sampling frequency of maximal 44 100 Hz ([164, page 22]).

In the following sections we describe the methods used for error detection and correction in connection with audio compact discs.

Exercises

E.5.1.1 **Exercise** Show that $L^1(I)$ is a normed vector space.

E.5.1.2 **Exercise** Assume that $p > 1$ is a real number. Verify the details that $L^p(I)$ is a normed vector space.

Hints: For $f, g \in L^p(I)$, show that the inequality

$$|f(x) + g(x)|^p \leq \big(|f(x)| + |g(x)|\big)^p \leq 2^p \big(|f(x)|^p + |g(x)|^p\big)$$

holds almost everywhere on I, and deduce that $f + g \in L^p(I)$.

The proof of the triangle inequality for $\|.\|_p$ is based on *Hölder's Inequality*. Assume that $p, q > 1$ satisfy $1/p + 1/q = 1$. If φ belongs to $L^p(I)$ and ψ belongs to $L^q(I)$, then $\varphi\psi$ is an element of $L^1(I)$ and

$$\|\varphi\psi\|_1 \leq \|\varphi\|_p \|\psi\|_q.$$

In this setting, the triangle inequality is also known as *Minkowski's Inequality*. Determine q such that $1/p + 1/q = 1$. Let $f, g \in L^p(I)$, then $h := |f + g|^{p-1}$ belongs to $L^q(I)$, and $h^q = |f + g|^{q(p-1)} = |f + g|^p = |f + g|\, h \leq |fh| + |gh|$. Applying Hölder's inequality twice, once for $(\varphi, \psi) = (f, h)$ and once for $(\varphi, \psi) = (g, h)$, show that

$$\|f + g\|_p^p \leq \Big(\|f\|_p + \|g\|_p\Big) \|f + g\|_p^{p/q}.$$

Exercise Prove that $L^2(I)$ is an Hermitian inner product space, i.e. the inner product is antilinear in the first argument, linear in the second argument and $\langle g, f\rangle = \overline{\langle f, g\rangle}$. E.5.1.3

Exercise For $f_1, f_2 \in L^1(\mathbb{R})$ and $c_1, c_2 \in \mathbb{R}$ show that E.5.1.4

$$\widehat{c_1 f_1 + c_2 f_2} = c_1 \hat{f}_1 + c_2 \hat{f}_2.$$

Consider $f \in L^1(\mathbb{R})$ and $a \in \mathbb{R}$. Prove that $g(x) := e^{-2\pi i a x} f(x)$ and $h(x) := f(x-a)$ are in $L^1(\mathbb{R})$ and verify

$$\hat{g}(\xi) = \hat{f}(\xi + a) \text{ and } \hat{h}(\xi) = e^{-2\pi i a \xi} \hat{f}(\xi).$$

5.2 Correction of Erasures 5.2

After this short introduction of Fourier Analysis and digital audio we come back to the theory of error correcting codes.

Definition (erasures) An *erasure* is a transmission error where the exact position of the error in the received vector is known (but of course not the exact value of the error). ◇ 5.2.1

Erasures occur, for instance, when the receiver obtained a vector of which it could not read certain components, or when it is known to the decoder that certain positions in a received vector are not valid.

In the sequel when speaking about a transmission error, we use the term *random error* in order to distinguish it from an erasure. The reader can imagine that, in general, it is easier to correct erasures than random errors.

Theorem [104, 3.3.1 Satz] *Let C be an (n,k,d)-code over $\mathbb{F}_q$ and let t, u be non-negative integers. If $d \geq 2t + u + 1$, then it is possible to correct up to t errors and, additionally, u erasures with C.* 5.2.2

Proof: Assume that the receiver has obtained a vector $y \in \mathbb{F}_q^n$ containing exactly u erasures, which are located at $0 \leq i_0 < i_1 < \ldots < i_{u-1} \leq n-1$, and in the remaining $n-u$ coordinates exactly t errors have occurred. Consider the set

$$V := \{v \in \mathbb{F}_q^n \mid v_i = y_i \text{ for } i \notin \{i_0, \ldots, i_{u-1}\}\},$$

then there is exactly one $c \in C$ such that $d(c,v) \leq t$ for a suitable $v \in V$. From the assumptions it is clear that such a codeword c exists. We have to show

that it is uniquely determined. Assuming that there exist $c^{(1)}, c^{(2)} \in C$, with $c^{(1)} \neq c^{(2)}$ and $v^{(1)}, v^{(2)} \in V$ such that $d(c^{(1)}, v^{(1)}) \leq t$ and $d(c^{(2)}, v^{(2)}) \leq t$, we obtain

$$d(c^{(1)}, c^{(2)}) \leq d(c^{(1)}, v^{(1)}) + d(v^{(1)}, v^{(2)}) + d(v^{(2)}, c^{(2)}) \leq t + u + t < d,$$

which is a contradiction to $\operatorname{dist}(C) = d$. □

5.2.3 **Example** If C is a binary code equipped with a decoding algorithm for random errors, then this algorithm can also be used for the correction of erasures.

Assume that $c \in C$ was sent and the received vector $y \in \mathbb{F}_2^n$ contains u erasures in the positions $i_0 < \ldots < i_{u-1}$. Moreover, assume that among the remaining components no more than t transmission errors have occurred and that $d \geq 2t + u + 1$. We consider two particular elements $v^{(0)}, v^{(1)}$ of V. The vector $v^{(0)}$ is obtained from y by replacing all erasures by 0, and $v^{(1)}$ is obtained by replacing them by 1. Since C is a binary code, one of two cases must occur: Either at least half the values at the erasure positions are zero or at least half the values are one. Consequently, in the first case $d(c, v^{(0)}) \leq t + \lfloor u/2 \rfloor$ and in the second case $d(c, v^{(1)}) \leq t + \lfloor u/2 \rfloor$.

Without loss of generality, assume that $d(c, v^{(0)}) \leq t + \lfloor u/2 \rfloor$. Then it is possible to decode $v^{(0)}$, and $v^{(0)}$ is decoded into the codeword originally sent. If $v^{(1)}$ cannot be decoded, then we are done. Assume that $v^{(1)}$ is decoded into $c' \in C$. If $c = c'$ we are also done. If $c' \neq c$ then

$$\left| \left\{ j \in n \mid c'_j \neq y_j,\ j \notin \{i_0, \ldots, i_{u-1}\} \right\} \right| > t,$$

thus c' cannot be obtained by filling the erasures and correcting up to t nonerased components of y. ◇

For codes over $\mathbb{F}_q$ with $q > 2$ this method is not very useful. In this case it would be necessary to compute all the q^u different vectors of V and find, according to 5.2.2, the existing and uniquely determined vector which can be decoded into a codeword so that at most t nonerased components are changed.

Hence, for BCH-codes we describe another method (cf. [104] and [68]). Let C be a BCH-code of length n over $\mathbb{F}_q$ with designed distance δ. By definition, there exists an integer b and a primitive n-th root of unity ξ such that the variety $V(C)$ contains the consecutive set $\{\xi^b, \ldots, \xi^{b+\delta-2}\}$. For $m = \operatorname{ord}_n(q)$, the primitive n-th root ξ belongs to $\mathbb{F}_{q^m}$. Let t and u be nonnegative integers such that $\delta \geq 2t + u + 1$. Moreover, we assume that the codeword c was sent and that the vector $y \in \mathbb{F}_q^n$ was received. It contains exactly u erasures and we assume that during the transmission $w \leq t$ errors have occurred in the nonerased positions.

From y we derive the vector $\widetilde{y}$ in which all the erased positions are replaced by 0. Thus, we can express $\widetilde{y}$ as the sum

$$\widetilde{y} = c + e + a,$$

where e is the error vector describing the errors which occurred in the nonerased positions, and a is the vector which produces the value 0 at all the erased positions. Then the error vector of $\widetilde{y}$ equals $f := e + a$.

Using the check matrix $\widetilde{\Delta}$ over the extension field $\mathbb{F}_{q^m}$, which was introduced in the proof of 4.3.1, we compute the syndrome s of $\widetilde{y}$ as

$$s = \widetilde{y} \cdot \widetilde{\Delta}^{\top}.$$

We are interested in certain components of s, the *partial syndromes*, which are given by

$$s_j = \sum_{i \in n} \widetilde{y}_i \xi^{ij} = \widetilde{y}(\xi^j) = c(\xi^j) + f(\xi^j) = f(\xi^j), \qquad b \le j \le b + 2t + u - 1. \tag{5.2.4}$$

As in 4.2.2 we consider the vectors $\widetilde{y}, c, f$ as polynomials. Since c is a codeword, $c(\xi^j) = 0$. This allows the computation of the *partial syndrome polynomial*

$$s(x) := \sum_{j \in 2t+u} s_{b+j} x^j.$$

From the received vector y we deduce that the erasures have occurred in positions $i_0, \ldots, i_{u-1}$. Moreover, we assume that the errors in the nonerased positions are located in $i_u, \ldots, i_{u+w-1}$. Actually these positions are not known in the beginning. In connection with these positions we have the *erasure location polynomial* which is given by

$$\lambda(x) := \prod_{j \in u} (1 - \xi^{i_j} x)$$

and the *error location polynomial* given by

$$\sigma(x) := \prod_{j=u}^{u+w-1} (1 - \xi^{i_j} x).$$

So far we know λ, but σ remains to be computed.

Now we turn our attention to the actual values of the errors. Once again, we recall that f_{i_j} for $j \in u + w$ denotes the error occurring at the position i_j when sending c and receiving $\widetilde{y}$. From 5.2.4, it follows that

$$s_\ell = \sum_{j \in u+w} f_{i_j} \xi^{i_j \ell}, \qquad b \le \ell \le b + 2t + u - 1. \tag{5.2.5}$$

Let ω be the unique polynomial of degree less than $2t + u$ satisfying

$$\omega(x) \equiv s(x)\lambda(x)\sigma(x) \bmod \mathrm{I}(x^{2t+u}).$$

It is called the *error evaluation polynomial*.

5.2.6 **Lemma** *Using the notation from above we have*

$$\omega(x) = \sum_{j \in u+w} f_{i_j} \xi^{i_j b} \prod_{r \neq j} (1 - \xi^{i_r} x).$$

Proof: From 5.2.4 and 5.2.5, we get

$$\begin{aligned} s(x)\lambda(x)\sigma(x) &= \left(\sum_{\ell \in 2t+u} \sum_{j \in u+w} f_{i_j} \xi^{i_j(b+\ell)} x^{\ell} \right) \prod_{r \in u+w} (1 - \xi^{i_r} x) \\ &= \sum_{j \in u+w} f_{i_j} \xi^{i_j b} \left(\sum_{\ell \in 2t+u} (\xi^{i_j} x)^{\ell} \right) \prod_{r \in u+w} (1 - \xi^{i_r} x) \\ &= \sum_{j \in u+w} f_{i_j} \xi^{i_j b} \frac{1 - (\xi^{i_j} x)^{2t+u}}{1 - \xi^{i_j} x} (1 - \xi^{i_j} x) \prod_{r \neq j} (1 - \xi^{i_r} x) \\ &\equiv \sum_{j \in u+w} f_{i_j} \xi^{i_j b} \prod_{r \neq j} (1 - \xi^{i_r} x) \bmod \mathrm{I}(x^{2t+u}). \end{aligned}$$

This is a polynomial of degree at most $w + u - 1 < 2t + u$, whence it represents the polynomial ω. □

Since we know the polynomial λ and since its degree is equal to u, we write $s\lambda$ in the form

$$s(x)\lambda(x) = T_1(x) + x^u T_2(x)$$

with $T_1, T_2 \in \mathbb{F}_{q^m}[x]$ and $\deg T_1 < u$. Then

$$s(x)\lambda(x)\sigma(x) - \sigma(x)T_1(x) = x^u \sigma(x) T_2(x),$$

whence

$$\omega(x) - \sigma(x)T_1(x) \equiv x^u \sigma(x) T_2(x) \bmod \mathrm{I}(x^{2t+u}).$$

Consequently, x^u is a divisor of $\omega(x) - \sigma(x)T_1(x)$ and finally we obtain

$$\frac{\omega(x) - \sigma(x)T_1(x)}{x^u} \equiv \sigma(x) T_2(x) \bmod \mathrm{I}(x^{2t}).$$

We denote the left-hand side of this congruence by $\Omega(x)$. It is a polynomial of degree at most $w - 1 \leq t - 1$.

5.2.7 **Lemma** *Let C be a BCH-code of length n over $\mathbb{F}_q$ with designed distance δ. The variety of C contains powers of ξ where $\xi \in \mathbb{F}_{q^m}$ is a primitive n-th root of unity. Consider nonnegative integers t, u such that $u + 2t + 1 \leq \delta$. Assume that the product of the partial syndrome polynomial and the erasure location polynomial is written in the form $s(x)\lambda(x) = T_1(x) + x^u T_2(x)$ with $\deg T_1 < u$. Then there exist relatively*

prime polynomials σ, $\Omega \in \mathbb{F}_{q^m}[x]$ *such that* $\deg \Omega \leq t-1$, $\deg \sigma \leq t$, *and* $\Omega(x) \equiv \sigma(x)T_2(x) \bmod \mathrm{I}(x^{2t})$.

In addition, the polynomials σ *and* Ω *are unique modulo scalars in the following sense. If* σ_1 *and* Ω_1 *are two further polynomials over* $\mathbb{F}_{q^m}$ *which are relatively prime and satisfy* $\deg \Omega_1 \leq t-1$, $\deg \sigma_1 \leq t$, *and* $\Omega_1(x) \equiv \sigma_1(x)T_2(x) \bmod \mathrm{I}(x^{2t})$, *then there exists a constant* $\kappa \in \mathbb{F}_{q^m}^*$ *such that* $\sigma_1 = \kappa\sigma$ *and* $\Omega_1 = \kappa\Omega$.

Proof: From the previous computations we already know that polynomials σ, Ω exist with $\deg \Omega \leq t-1$, $\deg \sigma \leq t$, and $\Omega(x) \equiv \sigma(x)T_2(x) \bmod \mathrm{I}(x^{2t})$. Now we prove that they are relatively prime. If we assume that $\varphi \in \mathbb{F}_{q^m}[x]$ is a common divisor of σ and Ω, then it is also a divisor of $x^u\Omega(x)$, therefore, a divisor of $\omega - \sigma T_1$, thus a divisor of ω. From 5.2.6 it is clear that ω and σ are relatively prime, thus φ is a unit in $\mathbb{F}_{q^m}[x]$, in other words, a nonzero constant, thus σ and Ω are relatively prime.

If σ_1 and Ω_1 also have the desired properties, then

$$\Omega(x)\sigma_1(x) \equiv \sigma(x)T_2(x)\sigma_1(x) \equiv \sigma(x)\Omega_1(x) \bmod \mathrm{I}(x^{2t}).$$

Comparing degrees, we obtain that

$$\Omega\sigma_1 = \sigma\Omega_1.$$

Thus, σ_1 is a divisor of $\sigma\Omega_1$. Since σ_1 and Ω_1 are relatively prime, σ_1 is a divisor of σ. Correspondingly, we derive that σ is a divisor of σ_1, whence there exists a constant $\kappa \in \mathbb{F}_{q^m}^*$ such that $\sigma_1 = \kappa\sigma$. Finally, from $\Omega\kappa\sigma = \Omega\sigma_1 = \sigma\Omega_1$ we deduce that $\kappa\Omega = \Omega_1$. □

So far we have proved that relatively prime polynomials σ and Ω of certain degrees exist, such that $\Omega(x) \equiv \sigma(x)T_2(x) \bmod \mathrm{I}(x^{2t})$. But we still don't know how to compute them. Next we describe a method for doing this which is based upon the

Extended Euclidean Algorithm [104, 3.2.12 Satz] *Consider nonzero polynomials* **5.2.8**
$\varphi, \psi \in \mathbb{F}_q[x]$ *with* $\deg \varphi \geq \deg \psi$. *Put*

$$\begin{array}{lll} f_0 := 1, & g_0 := 0, & r_0 := \varphi, \\ f_1 := 0, & g_1 := 1, & r_1 := \psi, \end{array}$$

and recursively for $i \geq 1$ *determine* q_i, r_{i+1}, f_{i+1}, *and* g_{i+1} *by*

$$\begin{array}{lcl} r_{i-1} & = & q_i r_i + r_{i+1} \quad \textit{with } \deg r_{i+1} < \deg r_i, \\ f_{i+1} & = & f_{i-1} - q_i f_i, \\ g_{i+1} & = & g_{i-1} - q_i g_i. \end{array}$$

Then

$$f_i\varphi + g_i\psi = r_i, \qquad i \geq 0, \tag{5.2.9}$$

and

5.2.10 $$\deg g_{i+1} \leq \deg \varphi - \deg r_i, \qquad i \geq 0.$$

Since $\deg r_{i+1} < \deg r_i$*, after finitely many steps this algorithm terminates with* $r_N \neq 0$ *and* $r_{N+1} = 0$*. Then* r_N *is equal to* $\gcd(\varphi, \psi)$. □

An application of the extended Euclidean Algorithm allows us to find polynomials σ and Ω with the properties described in 5.2.7.

5.2.11 **Sugiyama–Kasahara–Hirasawa–Namekawa algorithm** [188], [189] Let $\varphi(x) := x^{2t}$ and let $\psi(x)$ be the remainder of $T_2(x)$ after division by x^{2t}. Then there exist polynomials $f_N, g_N \in \mathbb{F}_{q^m}[x]$ such that

$$d(x) := \gcd\big(x^{2t}, T_2(x)\big) = \gcd\big(x^{2t}, \psi(x)\big) = f_N(x)x^{2t} + g_N(x)\psi(x).$$

If the assumptions of 5.2.7 are satisfied, then there exist relatively prime solutions Ω and σ of the congruence

$$\Omega(x) \equiv \sigma(x)\psi(x) \bmod \mathrm{I}(x^{2t})$$

with $\deg \Omega \leq t-1$ and $\deg \sigma \leq t$. Thus, there exists a polynomial ϕ such that

$$\Omega(x) = \sigma(x)\psi(x) + \phi(x)x^{2t}$$

and d is a divisor of Ω. Hence $\deg d \leq t-1$. The degrees of the polynomials r_i in 5.2.8 are strictly decreasing, and $\deg r_0 = \deg x^{2t} = 2t$. So, there exists an index $j_0 \in \{1, \ldots, N\}$ such that $\deg r_{j_0-1} \geq t$ and $\deg r_{j_0} < t$. From 5.2.9 and 5.2.10 we obtain

$$f_{j_0}(x)x^{2t} + g_{j_0}(x)\psi(x) = r_{j_0}(x)$$

and $\deg g_{j_0} \leq 2t - \deg r_{j_0-1} \leq t$. Consequently, g_{j_0} and r_{j_0} satisfy the congruence

$$r_{j_0}(x) \equiv g_{j_0}(x)T_2(x) \bmod \mathrm{I}(x^{2t})$$

with $\deg g_{j_0} \leq t$ and $\deg r_{j_0} < t$. Since, furthermore, Ω and σ are required to be relatively prime we set

$$\Omega(x) := \frac{r_{j_0}(x)}{\gcd(r_{j_0}(x), g_{j_0}(x))} \quad \text{and} \quad \sigma(x) := \frac{g_{j_0}(x)}{\gcd(r_{j_0}(x), g_{j_0}(x))}.$$

If the assumptions of 5.2.7 are not satisfied, then it may happen that $T_2 \neq 0$ and $\deg d \geq t$. In this case we cannot find j_0 such that $\deg r_{j_0} < t$ and an uncorrectable error has occurred. □

So far we have computed the partial syndrome polynomial s, the error location polynomial σ, the erasure location polynomial λ, and the error evaluation polynomial

$$\omega(x) = \Omega(x)x^u + \sigma(x)T_1(x).$$

In the next step the error locations $i_u, \ldots, i_{u+w-1}$ are determined by finding the roots of σ. It is obvious that $\kappa \in \mathbb{F}_{q^m}$ is a root of σ if and only if there exists some $j \in \{u, \ldots, u+w-1\}$ such that $1 - \xi^{i_j}\kappa = 0$, which is equivalent to $\kappa = \xi^{-i_j}$. From the representation of the multiplicative inverse of the roots of σ as powers of ξ, we are able to deduce the error locations. In Section 3.5 we were briefly discussing a method how to determine the roots of a polynomial. For the present problem of finding the zeros of the error locator polynomial, we suggest another method. If we are able to determine the polynomial σ, then, according to 5.2.7, we can assume that $\sigma(0) = 1$. Hence, the roots of σ can be determined by

Chien search **5.2.12**

Input: n the length of the code, $\xi \in \mathbb{F}_{q^m}$ a primitive n-th root of unity, and the error location polynomial $\sigma(x) = \sum_{i=0}^{w} \sigma_i x^i$ with $\sigma_0 = 1$.

Output: All roots of σ of the form ξ^j.

(1) We introduce w variables $X_1, \ldots, X_w$. For $1 \le j \le w$ set $X_j := \sigma_j$ and set $i := 0$.

(2) If

$$\sum_{j=1}^{w} X_j = -1,$$

then output ξ^i.

(3) If $i = n$, then STOP.
Otherwise, set $i := i + 1$ and for $1 \le j \le w$ set $X_j := X_j \xi^j$. Goto (2). □

In the i-th step the sum in (2) stands for

$$\sum_{j=1}^{w} \sigma_j \xi^{ij} = \sigma(\xi^i) - 1,$$

whence ξ^i is a root of σ if and only if this sum equals -1.

Finally, we have to compute the error values f_{i_j} for $j \in u + w$ which describe the errors of $\widetilde{y}$ in the i_j-th component. A useful method is given by the

Forney algorithm [55] *For $j \in u + w$ the error f_{i_j} can be computed as* **5.2.13**

$$f_{i_j} = \frac{\xi^{-i_j b}\omega(\xi^{-i_j})}{\prod_{r \ne j}(1 - \xi^{i_r - i_j})} = -\frac{\xi^{-i_j(b-1)}\omega(\xi^{-i_j})}{\rho'(\xi^{-i_j})},$$

where $\rho := \sigma\lambda$.

Proof: From 5.2.6 we obtain

$$\omega(\xi^{-i_j}) = f_{i_j}\xi^{i_j b}\prod_{r\neq j}(1-\xi^{i_r-i_j}),$$

which proves the first equality. Computing the formal derivative of ρ leads to

$$\rho'(x) = \frac{d}{dx}\prod_{j\in u+w}(1-\xi^{i_j}x) = -\sum_{j\in u+w}\xi^{i_j}\prod_{r\neq j}(1-\xi^{i_r}x).$$

Thus,

$$\rho'(\xi^{-i_j}) = -\xi^{i_j}\prod_{r\neq j}(1-\xi^{i_r-i_j}),$$

which proves the second equality. □

Hence, we obtain the error values f_{i_j} for $j \in u+w$, from which together with the error- and erasure locations i_j, the error vectors e and a can easily be determined. By subtraction we obtain the originally sent codeword $c = \widetilde{y} - e - a$.

5.2.14 **Algorithm (Decoding of errors and erasures with a BCH-code)** Let C be a BCH-code with designed distance δ and consecutive subset $\{\xi^b, \ldots, \xi^{b+\delta-2}\}$ of its variety, where $\xi \in \mathbb{F}_{q^m}$ is a primitive n-th root of unity.

Input: A vector y containing u erasures.

Output: A codeword c or an error message. If t denotes the number of errors in the nonerased positions and if $2t \leq \delta - 1 - u$, then the output is the codeword originally sent.

(1) Determine the vector $\widetilde{y}$ by setting the erased components of y all equal to 0, and define $t := \lfloor(\delta - u - 1)/2\rfloor$.

(2) Compute the partial syndromes $s_j = \widetilde{y}(\xi^j)$ for $b \leq j \leq b + 2t + u - 1$.

(3) Determine the partial syndrome polynomial

$$s(x) := \sum_{j\in 2t+u} s_{b+j}x^j.$$

If $s = 0$, then set $c := \widetilde{y}$ and goto (12).

(4) Determine the locations of the erasures $i_0, \ldots, i_{u-1}$ and form the erasure location polynomial

$$\lambda(x) := \prod_{j\in u}(1-\xi^{i_j}x).$$

(5) Write $s(x)\lambda(x)$ in the form $T_1(x) + x^u T_2(x)$ with $\deg T_1 < u$.

(6) Compute relatively prime polynomials $\Omega, \sigma \in \mathbb{F}_{q^m}[x]$ such that $\deg\Omega \leq t-1$, $\deg\sigma \leq t$, and $\Omega(x) \equiv \sigma(x)T_2(x) \bmod \mathrm{I}(x^{2t})$ as described in

5.2.11. If such polynomials do not exist, output an error message and STOP.

(7) Compute by Chien search the roots ξ^{-i_j} of σ for $j = u, \dots, u+w-1$, where $w \le \deg\sigma \le t$, and their multiplicative inverses. Then the error locations are $i_u, \dots, i_{u+w-1}$.

(8) Set $\rho := \sigma\lambda$ and compute the formal derivative $\rho'(x)$.

(9) Compute the error evaluation polynomial

$$\omega(x) = \Omega(x)x^u + \sigma(x)T_1(x).$$

(10) For $j \in u + w$ determine the errors

$$f_{i_j} = -\frac{\xi^{-i_j(b-1)}\omega(\xi^{-i_j})}{\rho'(\xi^{-i_j})}$$

as indicated by Forney's algorithm.

(11) Set

$$a(x) := \sum_{j\in u} f_{i_j}x^{i_j}, \qquad e(x) := \sum_{j=u}^{u+w-1} f_{i_j}x^{i_j},$$

and $c(x) := \widetilde{y}(x) - a(x) - e(x)$.

(12) Output c. □

This algorithm can also be used for decoding a BCH-code when no erasures occurred. In this case $u = 0$, $\widetilde{y} = y$, $\lambda = 1$, $T_1 = 0$, and $\omega = \Omega$.

Algorithm (Decoding of errors with a BCH-code) Let C be a BCH-code with designed distance δ and consecutive subset $\{\xi^b, \dots, \xi^{b+\delta-2}\}$ of its variety, where $\xi \in \mathbb{F}_{q^m}$ is a primitive n-th root of unity. **5.2.15**

Input: A vector y.

Output: A codeword c or an error message. If t denotes the number of errors and if $2t \le \delta - 1$, then the output is the codeword originally sent.

(1) Set $t := \lfloor(\delta-1)/2\rfloor$ and compute the partial syndromes $s_j = y(\xi^j)$ for $b \le j \le b + 2t - 1$.

(2) Determine the partial syndrome polynomial

$$s(x) := \sum_{j\in 2t} s_{j+b}x^j.$$

If $s = 0$, then set $c := y$ and goto (8).

(3) Compute relatively prime polynomials $\omega, \sigma \in \mathbb{F}_{q^m}[x]$ such that $\deg \omega \leq t-1$, $\deg \sigma \leq t$ and $\omega(x) \equiv \sigma(x)s(x) \bmod I(x^{2t})$ as described in 5.2.11. If such polynomials do not exist, output an error message and STOP.

(4) Compute by Chien search the roots ξ^{-i_j} of σ for $j \in w$, where $w \leq \deg \sigma \leq t$, and their multiplicative inverses. Then the error locations are $i_0, \ldots, i_{w-1}$.

(5) Compute the formal derivative $\sigma'(x)$.

(6) For $j \in w$ determine the errors

$$f_{i_j} = -\frac{\xi^{-i_j(b-1)}\omega(\xi^{-i_j})}{\sigma'(\xi^{-i_j})}$$

as indicated by Forney's algorithm.

(7) Set

$$e(x) := \sum_{j \in w} f_{i_j} x^{i_j} \text{ and } c(x) := y(x) - e(x).$$

(8) Output c. □

At the end of this section we want to discuss two numerical examples. First we illustrate the method of 5.2.3 for binary codes. It was taken from [104, 3.3.2 Beispiel].

5.2.16 **Example** Consider the binary $(15,7,5)$-code with generator polynomial $g(x) = x^8 + x^7 + x^6 + x^4 + 1$. According to 4.2.13, it is a BCH-code with $\delta = 5$, since g is the product of $x^4 + x + 1$ and $x^4 + x^3 + x^2 + x + 1$. Assume that

$$c = (1,0,0,1,1,0,1,0,1,1,1,1,0,0,0)$$

was sent and that the vector

$$y = (1,0,\square,1,1,0,0,0,1,1,1,1,0,0,\square)$$

was received. Thus, two erasures and one error have occurred. Now we set

$$v^{(0)} := (1,0,0,1,1,0,0,0,1,1,1,1,0,0,0)$$

and

$$v^{(1)} := (1,0,1,1,1,0,0,0,1,1,1,1,0,0,1).$$

Using a decoding algorithm for cyclic codes (for instance 5.2.15), we realize that $v^{(0)}$ is decoded to c, whereas $v^{(1)}$ cannot be decoded. ◇

Now we consider a BCH-code over $\mathbb{F}_{11}$. A similar example can be found in [104, 3.3.4 Beispiel].

Example Let C be a BCH-code in the strict sense of length 10 with designed distance $\delta = 6$ over $\mathbb{F}_{11}$. The parameters were chosen so that $\xi = 2$ is a primitive 10-th root of unity. Thus, it is not necessary to do computations in an extension field of $\mathbb{F}_{11}$. **5.2.17**

The receiver has obtained the vector

$$y = (8, \square, 8, \square, 2, \square, 4, 2, 2, 1).$$

Since $u = 3$ erasures have occurred, we can detect at most $t = 1$ error in the nonerased positions. Initially, we set

$$\widetilde{y} := (8, 0, 8, 0, 2, 0, 4, 2, 2, 1).$$

Then we compute the first five components of its syndrome

$$s_{j-1} = \sum_{i=0}^{9} \widetilde{y}_i 2^{ij}, \qquad 1 \le j \le 5,$$

and obtain the partial syndrome polynomial

$$s(x) = 10x^4 + 5x^3 + 2x + 2.$$

The erasures occurred at the positions $i_0 = 1$, $i_1 = 3$ and $i_2 = 5$, whence

$$\lambda(x) = (1 - 2x)(1 - 2^3 x)(1 - 2^5 x) = 5x^3 + 6x^2 + 2x + 1.$$

Now we compute

$$s(x)\lambda(x) = 6x^7 + 8x^6 + 6x^5 + 8x^4 + 5x^3 + 5x^2 + 6x + 2$$

and determine

$$T_1(x) = 5x^2 + 6x + 2 \text{ and } T_2(x) = 6x^4 + 8x^3 + 6x^2 + 8x + 5 \equiv 8x + 5 \bmod \mathrm{I}(x^2).$$

A solution of the congruence

$$\Omega(x) \equiv \sigma(x) T_2(x) \bmod \mathrm{I}(x^2)$$

with $\deg \sigma \le 1$ and $\deg \Omega = 0$ is easily computed as

$$\sigma(x) = x + 9 \quad \text{and} \quad \Omega(x) = 1.$$

The root of σ is $\kappa = 2 = \xi$, whence $\kappa^{-1} = \xi^{-1} = \xi^9$ and the error location $i_3 = 9$. Now we determine

$$\rho(x) = \sigma(x)\lambda(x) = 5x^4 + 7x^3 + x^2 + 8x + 9,$$

$$\rho'(x) = 9x^3 + 10x^2 + 2x + 8$$

and

$$\omega(x) = \Omega(x)x^3 + \sigma(x)T_1(x) = 6x^3 + 7x^2 + x + 7.$$

This allows us to compute the following data

j	0	1	2	3
i_j	1	3	5	9
2^{-i_j}	6	7	10	2
$\omega(2^{-i_j})$	10	6	7	8
$\rho'(2^{-i_j})$	3	2	7	3
f_{i_j}	4	8	10	1

so that $a(x) = 4x + 8x^3 + 10x^5$, $e(x) = x^9$ and

$$c(x) = \widetilde{y}(x) - a(x) - e(x) = 8 + 7x + 8x^2 + 3x^3 + 2x^4 + x^5 + 4x^6 + 2x^7 + 2x^8,$$

which results in the codeword

$$c = (8, 7, 8, 3, 2, 1, 4, 2, 2, 0). \qquad \diamond$$

Exercises

E.5.2.1 **Exercise** Prove 5.2.8. First prove by induction that 5.2.9 is satisfied for $i \geq 0$. Show that $\deg q_i = \deg r_{i-1} - \deg r_i$ for $i \geq 1$. From the monotonicity of $\deg r_i$ deduce by induction that 5.2.10 is satisfied for $i \geq 0$. Finally, show by induction that $\gcd(r_{i+1}, r_i) = \gcd(r_i, r_{i-1}) = \gcd(\varphi, \psi)$ for $i \geq 1$.

E.5.2.2 **Exercise** Consider the code from 5.2.16 and the decoding method described in 5.2.3. Using the algorithm 5.2.15, show that both $v^{(0)}$ and $v^{(1)}$ constructed from

$$y = (0, 0, 0, 0, 1, 0, 0, \square, \square, 0, 0, 0, 0, 0, 0)$$

can be decoded into two distinct codewords $c^{(0)}$ and $c^{(1)}$. Assume that at most one error has occurred in the positions unaffected by erasures. Which of $c^{(0)}$ and $c^{(1)}$ was sent originally? (See [104, Übung 60].)

E.5.2.3 **Exercise** Let C be the BCH-code over $\mathbb{F}_7$ of length 6 with consecutive subset $\{\xi, \ldots, \xi^4\}$ of its variety and designed distance $\delta = 5$, where $\xi = 3 \in \mathbb{F}_7$. The vector $y = (3, 3, \square, 5, \square, 2)$ was received. Which codeword was sent? (See [104, Übung 61].)

5.3 Burst Errors and Interleaving of Codes

5.3

If during the transmission of a codeword at least two errors have occurred, then the error vector can be seen as a burst error.

Definition (burst error) Let C be a code of length n. A *burst error* of length $\ell \leq n$ is an error vector e in which (after some cyclic rearrangement) all the nonzero entries (maybe mixed with some zero entries) occur in ℓ adjacent positions, say among the coordinates $e_i, \ldots, e_{i+\ell-1 \bmod n}$, with nonzero e_i and $e_{i+\ell-1 \bmod n}$. 5.3.1

A burst error of length ℓ is completely described by its *location*, which is the index i of the first nonzero component of e, and by its *pattern*, which can also be read as a polynomial $b(x) := e_i \cdot 1 + e_{i+1 \bmod n} x + \ldots + e_{i+\ell-1 \bmod n} x^{\ell-1}$ of degree $\ell - 1$. ◇

For example the two vectors

$$(0,0,0,0,1,0,1,1,0,0,0,0) \text{ and } (1,0,0,0,0,0,0,0,0,1,1,0)$$

are both burst errors of length 4. The first burst has location 4 and pattern $1 + x^2 + x^3$, the second one has location 9 and pattern $1 + x + x^3$. In order to indicate that the second burst starts somewhere at the end of the vector and is continued at the beginning, we also call it a *wrap around burst*. However, neither the burst error nor its pattern and length are uniquely determined. The first vector is also a wrap around burst with location 6 and pattern $1 + x + x^{10}$, so a burst of length 11, or a wrap around burst with location 7 and pattern $1 + x^9 + x^{11}$. Of course, it makes sense to consider bursts of rather short length.

Again we use the term *random error* instead of *error* in the ordinary sense in order to distinguish it from a burst error. Random errors are sometimes seen as burst errors of length 1.

If the burst error has location i and pattern b of degree $\ell - 1$, then

$$e(x) \equiv x^i b(x) \bmod \mathrm{I}(x^n - 1).$$

Theorem *A cyclic code with generator polynomial of degree t is able to detect all burst errors of length at most t.* 5.3.2

Proof: Let e be a burst error of length at most t, then $e(x) \equiv x^i b(x) \bmod \mathrm{I}(x^n - 1)$ for some i, with $b \neq 0$ and $\deg b < t$. In order to show that e is not a codeword, we prove that e is not a multiple of the generator polynomial g.

On the contrary, if we suppose that $x^i b(x)$ belongs to the code, then there exists a polynomial a such that

$$x^i b(x) \equiv a(x) g(x) \bmod \mathrm{I}(x^n - 1).$$

Hence, $x^n - 1$ is a divisor of $x^i b(x) - a(x)g(x)$, thus g is a divisor of $x^i b(x)$. Since $g(0) \neq 0$, we deduce that g is a divisor of b, which is impossible since $0 \leq \deg b < \deg g$. □

This fact motivates the method of *cyclic redundancy check* decoding (CRC-decoding) which is described below. It is applied for instance in computer networks, where it is important to detect errors. If a block of data was not transmitted correctly, then a transmission error is detected and the corresponding data must be transmitted again. CRC-decoding is also applied for error detection in digital audio applications.

5.3.3 Algorithm (CRC-decoding [104, pages 86ff]**)**
Assume that C is a cyclic code of length n with generator polynomial g of degree $t = n - k$. Moreover, assume that the code was encoded systematically.

Input: A polynomial $y(x) = y_0 + \ldots + y_{n-1}x^{n-1} \in \mathbb{F}_q[x]$.

Output: A vector belonging to $\mathbb{F}_q^k$ or an error message. If a transmission error has occurred which is a burst of length at most t, then this error is definitely detected by this algorithm.

(1) By polynomial division determine $q, r \in \mathbb{F}_q[x]$ such that $y = qg + r$ with $r = 0$ or $\deg r < \deg g$.

(2) If $r \neq 0$ then output an error message.

(3) If $r = 0$ output the information symbols of y. □

Especially the error detection rate of binary CRC-codes is quite high. Many burst errors of even bigger length can be detected by this method.

5.3.4 Lemma *Let C be a binary cyclic code with a generator polynomial of degree t and let $\ell > t$. Assume that all burst errors of length ℓ occur with the same probability. Then the probability that C detects a burst of length ℓ is equal to*

$$\begin{array}{ll} 1 - 2^{-t+1} & \text{if } \ell = t+1, \\ 1 - 2^{-t} & \text{if } \ell > t+1. \end{array}$$

Proof: Assume that e is a burst error with location i and pattern $b(x) = 1 + \ldots + x^{\ell-1}$. As we saw in the proof of 5.3.2, the burst e is not detected by the CRC-algorithm if and only if b is a multiple of g, thus $b = gf$ for some polynomial f of degree $\ell - 1 - t$. Moreover, from $b(0) = g(0) = 1$ we deduce that $f(0) = 1$ must be satisfied.

If $\ell = t + 1$, then there exists exactly one polynomial f with these properties, namely $f = 1$. If $\ell \geq t + 2$, then there are exactly $2^{\ell-t-2}$ polynomials f with these properties. By assumption, each burst pattern b of length ℓ occurs

with the same probability. Since there are exactly $2^{\ell-2}$ burst patterns of length ℓ, we obtain the following probabilities for detecting a burst of length ℓ.

If $\ell = t+1$, a burst of length ℓ is detected with the probability

$$\frac{2^{\ell-2}-1}{2^{\ell-2}} = 1 - 2^{-\ell+2} = 1 - 2^{-t+1}.$$

If $\ell > t+1$, a burst of length ℓ is detected with the probability

$$\frac{2^{\ell-2}-2^{\ell-t-2}}{2^{\ell-2}} = 1 - 2^{-t}. \qquad \square$$

For the correction of burst and random errors we refer the reader to [134] and [68]. Now we analyze the burst error correction ability of linear codes.

Lemma 5.3.5 *A linear code C is able to correct all bursts of length at most ℓ if and only if no codeword different from 0 is the sum of two bursts of length at most ℓ.*

Proof: Assume that C is able to correct all bursts of length up to ℓ and suppose that there exist two burst errors e and e' of length at most ℓ and $c \in C \setminus \{0\}$ such that $c = e + e'$. Then $0 + e' = c - e$. Since C is able to correct all bursts of length at most ℓ, the vector $0 + e$ is decoded into 0, and $c - e$ is decoded into c, whence $c = 0$, which is a contradiction to our assumption.

Conversely, we prove that if it is not possible to correct all bursts of length up to ℓ, then there exists a codeword different from 0 which is the sum of two bursts of length at most ℓ. By assumption, there exist $c, c' \in C$, $c \neq c'$ and two bursts e, e' of length at most ℓ such that $c + e = c' + e'$. Consequently, $e' - e = c - c' \in C \setminus \{0\}$ shows that the codeword $c - c'$ is the sum of e' and $-e$. $\square$

The Reiger-bound 5.3.6 [169] *If C is a linear (n,k)-code over $\mathbb{F}_q$ which is able to correct all bursts of length at most ℓ, then $2\ell \leq n - k$.*

Proof: Since C corrects all bursts of length at most ℓ, according to 5.3.5, no codeword different from 0 is the sum of two bursts of length at most ℓ. Consequently $2\ell < n$, since each vector of length at most 2ℓ is either a burst of length at most ℓ or the sum of two bursts of length at most ℓ. Hence, it is possible to consider the vector space

$$T := \left\{ (v_0, v_1, \ldots, v_{2\ell-1}, 0, \ldots, 0) \in \mathbb{F}_q^n \;\middle|\; v_i \in \mathbb{F}_q,\ i \in 2\ell \right\},$$

which is a subspace of $\mathbb{F}_q^n$ of dimension 2ℓ. Each element of $T \setminus \{0\}$ is either a burst of length at most ℓ or can be written as the sum of two bursts of length at most ℓ. Therefore, $c = 0$ is the unique codeword belonging to T. For $v, v' \in T$,

$v \neq v'$, the cosets $v + C$ and $v' + C$ are distinct elements of $\mathbb{F}_q^n / C$. (Assuming on the contrary that $v + C = v' + C$, then $v - v' \in C \cap T$, thus $v - v' = 0$, whence $v = v'$ a contradiction to our assumption.) Hence, the syndromes of distinct elements of T are distinct. In other words, the mapping $T \to \mathbb{F}_q^{n-k}$, $v \mapsto v \cdot \Delta^\top$, where Δ is a check matrix of C, is injective. Thus $2\ell = \dim(T) \leq \dim(\mathbb{F}_q^{n-k}) = n - k$. □

If an (n,k)-code is able to correct all burst errors of length at most ℓ, then its *burst error correcting efficiency* is defined as

$$\frac{2\ell}{n-k}.$$

From the Reiger-bound it is clear that the burst error correcting efficiency cannot be greater than 1. The minimum distance d of each MDS-code satisfies $d - 1 = n - k$. Hence, we obtain the following

5.3.7 **Corollary** *MDS-codes are able to correct all bursts of length at most* $\lfloor (n-k)/2 \rfloor$. *If* $n - k$ *is even, then their burst error correcting efficiency is equal to* 1. □

Assume that C is a linear code encoded systematically so that the *last* k positions of a codeword are the information symbols. If g is the generator polynomial of C, then the syndrome s of $y \in \mathbb{F}_q^n$ satisfies $s(x) \equiv y(x) \bmod g(x)$. Moreover, it is easily verified that the syndrome of the cyclic shift $x^i y(x)$ is given by $x^i s(x) \bmod g(x)$. In order to correct correctable bursts, the method of error trapping (cf. 4.12.5) can be applied. By the Reiger-bound, a correctable burst is always of length $\ell \leq (n-k)/2$, whence it can always be rotated by a cyclic shift into the first $n - k$ positions.

Now we describe a method which allows us to deal with long burst errors. Let C be a linear (n,k)-code over $\mathbb{F}_q$ and let λ be a positive integer. Consider the set of all λ-tuples of codewords of C, i.e.

$$C^\lambda = \{ f \mid f : \lambda \to C \},$$

which is an $\mathbb{F}_q$ vector space. Its elements can be represented as matrices. Assume that $f(i) = c^{(i)} = (c_0^{(i)}, \ldots, c_{n-1}^{(i)})$ for $i \in \lambda$, then f can be written as a matrix

$$f = \begin{pmatrix} c^{(0)} \\ c^{(1)} \\ \vdots \\ c^{(\lambda-1)} \end{pmatrix} = \begin{pmatrix} c_0^{(0)} & \cdots & c_{n-1}^{(0)} \\ c_0^{(1)} & \cdots & c_{n-1}^{(1)} \\ \vdots & \vdots & \vdots \\ c_0^{(\lambda-1)} & \cdots & c_{n-1}^{(\lambda-1)} \end{pmatrix},$$

where the codewords of C are the rows of this matrix. Now the main idea of interleaving is, to read this matrix columnwise from top to bottom and from left

to right. For this reason, the symbols of λ consecutive codewords are mixed. Finally, we identify the matrix f with the vector

$$(c_0^{(0)}, c_0^{(1)}, \ldots, c_0^{(\lambda-1)}, \ldots, c_{n-1}^{(0)}, c_{n-1}^{(1)}, \ldots, c_{n-1}^{(\lambda-1)})$$

of length λn over $\mathbb{F}_q$. The identification of $\lambda \times n$ matrices over $\mathbb{F}_q$ with vectors of length λn over $\mathbb{F}_q$ is a vector space isomorphism

$$\phi : \mathbb{F}_q^{\lambda \times n} \to \mathbb{F}_q^{\lambda n}.$$

The image of C^λ under this isomorphism is indicated by $C^{(\lambda)}$.

Definition (λ-way interleave) The code $C^{(\lambda)}$ is called the *λ-way interleave* of the code C. ◇ **5.3.8**

The proof of the next theorem is left to the reader.

Theorem *Let C be a linear (n, k, d, q)-code, and let λ be a positive integer. Then the λ-way interleave $C^{(\lambda)}$ is a linear $(\lambda n, \lambda k, d, q)$-code.* □ **5.3.9**

Theorem *If the linear code C is able to correct all bursts of length at most ℓ, then the λ-way interleave $C^{(\lambda)}$ corrects all burst errors of length at most $\lambda\ell$.* **5.3.10**

Proof: Assume that $e \in \mathbb{F}_q^{\lambda n}$ is a burst error of length at most $\lambda\ell$. Applying the inverse isomorphism ϕ^{-1}, this burst error is split over the λ rows of $\phi^{-1}(e)$. In each row at most ℓ consecutive symbols are effected, whence in each row the bursts can be corrected by C. Thus, e is corrected by $C^{(\lambda)}$.

Conversely, choose a burst of length $\ell + 1$ in $\mathbb{F}_q^n$ which cannot be corrected by C. If we replace ℓ consecutive symbols in an arbitrary row of $\phi^{-1}(c)$ for some $c \in C^{(\lambda)}$ by this burst, then we obtain a burst of length $\lambda\ell + 1$ in $C^{(\lambda)}$ which cannot be corrected. □

Corollary *Let C be a code over $\mathbb{F}_{2^m}$. If C allows one to correct bursts of length at most ℓ, then C is able to correct bursts of maximal $(\ell - 1)m + 1$ bits, and $C^{(\lambda)}$ corrects burst errors of length up to $(\lambda\ell - 1)m + 1$ bits.* □ **5.3.11**

Since C and $C^{(\lambda)}$ have the same minimum distance, interleaving does not increase the random error correction abilities of a code. As a matter of fact, depending on how the errors are distributed among the λ interleaves, more than $(d - 1)/2$ errors might be corrected by $C^{(\lambda)}$.

Theorem *Let C be a cyclic code of length n with generator polynomial g over $\mathbb{F}_q$. Then $g(x^\lambda)$ is the generator polynomial of $C^{(\lambda)}$. Hence, the λ-way interleave is also cyclic.* **5.3.12**

Proof: Let $c = (c_0, \ldots, c_{n\lambda-1})$ be an element of $C^{(\lambda)}$, then

$$\phi^{-1}(c) = \begin{pmatrix} c_0 & c_\lambda & \cdots & c_{(n-1)\lambda} \\ c_1 & c_{\lambda+1} & \cdots & c_{(n-1)\lambda+1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{\lambda-1} & c_{2\lambda-1} & \cdots & c_{n\lambda-1} \end{pmatrix}.$$

Each row

$$c^{(i)} := (c_i, c_{\lambda+i}, \ldots, c_{(n-1)\lambda+i}), \quad i \in \lambda,$$

is a codeword of C, thus $g(x)$ is a divisor of

$$c^{(i)}(x) := c_i + c_{\lambda+i}x + \ldots + c_{(n-1)\lambda+i}x^{n-1}, \quad i \in \lambda.$$

The polynomial

$$c(x) = c_0 + c_1 x + \ldots + c_{\lambda n-1}x^{\lambda n-1}$$

is of the form

$$c(x) = c^{(0)}(x^\lambda) + xc^{(1)}(x^\lambda) + \ldots + x^{\lambda-1}c^{(\lambda-1)}(x^\lambda).$$

Since g is a divisor of $c^{(i)}(x)$ in $\mathbb{F}_q[x]/\mathrm{I}(x^n - 1)$, it is obvious that $g(x^\lambda)$ is a divisor of $c(x)$ in $\mathbb{F}_q[x]/\mathrm{I}(x^{\lambda n} - 1)$. Hence, each codeword of $C^{(\lambda)}$ is a multiple of $g(x^\lambda)$.

Conversely, we know that g is a divisor of $x^n - 1$ and that $k = \dim(C) = n - \deg g$. Thus, $g(x^\lambda)$ is a divisor of $x^{n\lambda} - 1$, and the cyclic code generated by $g(x^\lambda)$ is of dimension $n\lambda - \deg g(x^\lambda) = n\lambda - (n-k)\lambda = k\lambda$. This is the dimension of $C^{(\lambda)}$. Since $C^{(\lambda)}$ is contained in the cyclic code generated by $g(x^\lambda)$, and both codes have the same dimension, they are equal. Thus, $C^{(\lambda)}$ is the cyclic code generated by $g(x^\lambda)$. □

The minimum distance of an interleaved code can be increased significantly by adding check symbols across interleaves. Usually product codes (cf. 2.3.15) are used for this construction.

Assume that C_i is an (n_i, k_i, d_i, q)-code, for $i = 1, 2$. Without loss of generality, we assume that the codes are systematically encoded, so that the first k_i symbols in each codeword form an information set. Then the product $C_1 \otimes C_2$ is an $(n_1 n_2, k_1 k_2, d_1 d_2, q)$-code. Its elements are represented as $n_1 \times n_2$-matrices over $\mathbb{F}_q$. First write down k_1 rows containing codewords of C_2. Then consider each of the n_2 columns as an information set of a codeword of C_1. For each column compute the remaining $n_1 - k_1$ check symbols and attach them at the

bottom of the column. Finally, we obtain a matrix of the form

$k_1 \times k_2$ information symbols	$k_1 \times (n_2 - k_2)$ checks on rows
$(n_1 - k_1) \times k_2$ checks on columns	$(n_1 - k_1) \times (n_2 - k_2)$ checks on rows and columns

5.3.13

containing $k_1 k_2$ information symbols in the upper left corner. The code C_1 is also called the *column code* or *outer code*, whereas C_2 is the *row code* or *inner code*. As a matter of fact, all the rows of this matrix, i.e. also the last $n_1 - k_1$ rows, are codewords of C_2 (cf. Exercise 5.3.3). For the encoding process it is not important whether first rows and then columns, or first columns and then rows of the matrix 5.3.13 are determined. Usually the components of this matrix are finally read in columns. Thus from the matrix

$$\begin{pmatrix} c_{00} & c_{01} & \cdots & c_{0,n_2-1} \\ c_{10} & c_{11} & \cdots & c_{1,n_2-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n_1-1,0} & c_{n_1-1,1} & \cdots & c_{n_1-1,n_2-1} \end{pmatrix}$$

we obtain the vector

$$(c_{00}, c_{10}, \ldots, c_{n_1-1,0}, c_{01}, c_{11}, \ldots, c_{n_1-1,1}, \ldots, c_{0,n_2-1}, c_{1,n_2-1}, \ldots, c_{n_1-1,n_2-1}).$$

There are various methods for decoding product codes. The conventional decoding is done in two steps, which are called *inner* and *outer decoding*. The inner decoding, also known as *row decoding*, is used both for error correction of short errors and for error detection. If errors were detected and not corrected in a row, then all the symbols of this row are marked as erasures. The outer decoder, also known as *column decoder*, is provided with information on erasures by the inner decoder, whence its main task is the correction of these erasures. Moreover, it is possible to use it for further error correction as well. If the outer decoder cannot correct the erasures, then in applications like compact discs an *error concealment* must be applied.

Example Consider the product code $C_1 \otimes C_2$ constructed from an extended binary Hamming-code C_1 with systematic generator matrix 5.3.14

$$\Gamma_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix},$$

a binary cyclic code C_2 with generator polynomial

$$g_2(x) = (1+x)(1+x+x^3) = 1+x^2+x^3+x^4$$

and systematic generator matrix

$$\Gamma_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

We already know that $d_1 = d_2 = 4$, whence $C_1 \otimes C_2$ is a binary $(56, 12, 16)$-code. Since the generator polynomial g_2 is of degree 4, the code C_2 detects all bursts of length ≤ 4 and, moreover, it detects bursts of length greater than 4 with probability at least $7/8$. For encoding the information

$$111\ 101\ 010\ 011$$

we insert it in form of rows into the array

$$M_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

Then, by using the systematic generator matrix Γ_2, we compute the four codewords of C_2 whose first three components are given by the rows of M_0, obtaining the rows of

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Finally, by using Γ_1, we compute the seven codewords of C_1 (written as columns) whose first four components are given by the columns of M_1. This way we obtain

$$M_2 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Reading M_2 column by column yields the vector

$$11001100\ 10110100\ 11010010\ 01111000\ 01100110\ 10101010\ 00011110.$$

After transmission we obtain the following vector

$$\underline{0}1011\underline{0}00\ \underline{0}0\underline{1}0\underline{0}\underline{0}00\ \underline{0}1010010\ 01111000\ 01100110\ 10101010\ \underline{1}0011110.$$

For better readability the transmission errors are underlined. In this vector we can find a burst of length 17 in position 0 with pattern 10010100100101001 and a random error in position 48. Rewriting this vector as a matrix we obtain

$$M_2' = \begin{pmatrix} \underline{0} & \underline{0} & \underline{0} & 0 & 0 & 1 & \underline{1} \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ \underline{1} & \underline{0} & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ \underline{0} & \underline{0} & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In the present example we use C_2 just for error detection. The first row of M_2' contains a burst of length 4 which will be detected by C_2, whence all entries of this row are marked as erasures. Also the errors in the two other rows are detected by C_2, and the elements in these rows are also marked as erasures. This way we obtain the matrix

$$M_2'' = \begin{pmatrix} \square & \square & \square & \square & \square & \square & \square \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ \square & \square & \square & \square & \square & \square & \square \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ \square & \square & \square & \square & \square & \square & \square \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since $d_1 = 4$, it is possible to correct three erasures in each codeword of C_1, whence each column of M_2'' can be corrected. Consequently we are able to reconstruct the originally sent vector.

Just using interleaving of one code it would be impossible to correct these errors. However, if the errors were distributed so that a fourth row of M_2' would be infected, then our decoding strategy would fail. For this reason, usually the code C_2 is also used for correction of short errors. Moreover, further interleaving of codewords of the product code $C_1 \otimes C_2$ protects better against burst errors. ◇

A similar example can be found in [104, 3.5.3 Beispiel].

5.3.15 **Example** For error correction in a DVD a product code of two Reed–Solomon-codes over $\mathbb{F}_{2^8}$ is used. The column code C_1 is a $(208, 192, 17)$-code with generator polynomial

$$g_1(x) = \prod_{i=0}^{15}(x + \xi^i),$$

where ξ is a root of the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1 \in \mathbb{F}_2[x]$. The row code C_2 is a $(182, 172, 11)$-code with generator polynomial

$$g_2(x) = \prod_{i=0}^{9}(x + \xi^i).$$

For further details see [49]. ◇

The error- and burst-correcting properties of product codes are described in

5.3.16 **Theorem** [134, pages 275ff] *Assume that C_i is a linear code of length n_i with minimum distance d_i which can correct all bursts of length at most ℓ_i, for $i = 1, 2$. Then:*

1. *The product code $C_1 \otimes C_2$ is capable of correcting*
$$t := \left\lfloor \frac{d_1 d_2 - 1}{2} \right\rfloor$$
random errors.
2. *There exist decoding methods such that $C_1 \otimes C_2$ corrects all bursts of length up to* $\max\{n_1\ell_2, n_2\ell_1\}$.
3. *There exists a decoding algorithm for the product code $C_1 \otimes C_2$ which allows the correction of all random errors of weight at most t and of all burst errors of length at most* $\max\{n_1 t_2, n_2 t_1\}$ *with* $t_i := \lfloor (d_i - 1)/2 \rfloor$.

Proof: 1. The first assertion is trivial since the minimum distance of $C_1 \otimes C_2$ is $d = d_1 d_2$.

2. Assume that the elements of a codearray are transmitted in columns. If during the transmission a burst error of length at most $n_1\ell_2$ has occurred, after rearranging the received data in an $n_1 \times n_2$-array, the elements of the burst error lie in at most $\ell_2 + 1$ columns. Each row of the array is affected with a burst of length not greater than ℓ_2. Thus, these bursts can be corrected by C_2, whence $C_1 \otimes C_2$ corrects all bursts of length up to $n_1\ell_2$. If $n_2\ell_1 > n_1\ell_2$, then assume that the elements of a codearray are transmitted in rows. Similar arguments prove that under these assumptions $C_1 \otimes C_2$ is capable of correcting all bursts of length at most $n_2\ell_1$.

3. Without loss of generality, we assume that $n_1 t_2 \geq n_2 t_1$ and set $\ell := n_1 t_2$. If the received vector y contains a random error of weight at most t we apply

the decoding method of 1. Otherwise, we suppose that y contains a burst error e of length not greater than ℓ which is not a random error of weight at most t. Rearranging the elements of y as an $n_1 \times n_2$-array, the burst error affects at most $t_2 + 1$ columns, where each row of this array contains at most t_2 errors. Consequently, these errors can be corrected by C_2. Finally, we want to show that the syndromes of errors can be used in order to determine which decoding strategy should be applied. For doing this, we prove that the syndrome of e, a burst of length ℓ which is not a random error of weight at most t, is different from the syndromes of all correctable random errors. Supposing, on the contrary, that the syndrome of e coincides with the syndrome of a correctable random error e', then the difference $e - e'$ is a codeword of $C_1 \otimes C_2$. By assumption, $e - e' \neq 0$. Hence, each nonzero row of $e - e'$ is of weight at least d_2. Thus, it consists of at least $d_2 - t_2$ nonzero entries of e' and at most t_2 nonzero components of e. Since there are at most t random errors in e', there exist at most $\lfloor t/(d_2 - t_2) \rfloor$ rows of e' containing at least $d_2 - t_2$ nonzero entries. Thus,

$$\mathrm{wt}(e-e') \leq \left\lfloor \frac{t}{d_2 - t_2} \right\rfloor t_2 + t \leq \frac{tt_2}{d_2 - t_2} + t = t\left(\frac{t_2}{d_2 - t_2} + 1\right) = t\frac{d_2}{d_2 - t_2} < 2t,$$

since

$$\frac{d_2}{d_2 - t_2} = \frac{d_2}{d_2 - \lfloor (d_2 - 1)/2 \rfloor} \leq \frac{2d_2}{2d_2 - d_2 + 1} < 2.$$

Consequently, $e - e'$ is a nonzero codeword of $C_1 \otimes C_2$ of weight less than $2t < d$, which is a contradiction. □

As already mentioned, there exist many other decoding methods for product codes. For instance, first determine all the rows and columns where errors have occurred, and flag all those entries lying both in infected rows and columns. Then use the conventional decoding strategy. But instead of erasing complete rows, just the flagged symbols in these rows will be erased.

The decoding process can be iterated. After having decoded rows and columns, start the decoding process once again. This method is especially useful for decoders with soft input and output. The word "soft" indicates that each data symbol is attached with a measure, usually an element of the real interval $[0, 1]$, indicating its reliability. See the vast literature on concatenated codes, e.g. [56], and on Turbo codes, e.g. [9], [12], and many others.

The effectiveness of the decoding method quite often depends on the situation where it is applied:

Example Consider the product code of two Hamming-codes. Its minimum distance is $3 \cdot 3 = 9$, whence it is possible to correct up to 4 errors. We will compare three different decoding methods in two different situations: **5.3.17**

Method 1: First correct all correctable rows with the row code, and then correct all correctable columns with the column code.

Method 2: Use the row code for error detection and mark all infected rows as erasures. With the column code try to decode errors and erasures.
Situation 1: Assume that four errors are located in the positions

$$(i_1, j_1), \quad (i_1, j_2), \quad (i_2, j_1), \quad (i_2, j_2)$$

of the codearray with $i_1 \neq i_2$, $j_1 \neq j_2$. Method 1 introduces in the rows indexed with i_1 and i_2 and also in the columns indexed with j_1 and j_2 a third error, whence the error cannot be corrected. Method 2 allows one to realize that errors have occurred in the rows i_1 and i_2, whence these rows are erased. Since there are exactly two erasures in each column, the erased symbols can be computed, and the errors are corrected.
Situation 2: Now assume that the four errors have occurred in four different rows and columns, thus there are exactly four rows and four columns containing one error. Method 1 corrects the errors in the four erroneous rows, whence there are no errors left for the column decoding. Thus, all the errors were corrected. Method 2 allows one to realize that errors have occurred in four rows. The symbols in each of these rows are erased. Since there are four erased symbols in each column, the column decoder cannot correct these errors.
The errors in both situations can be corrected by using
Method 3: Correct each row using a decoder for the row code. For each row i, remember ν_i, the number of symbols corrected. Larger values of ν_i correspond to rows which are more likely to have been miscorrected. Uncorrectable rows are tagged with $\nu_i = \infty$ and all symbols in these rows are immediately erased. Then correct the columns using an errors-and-erasures correction method. For $j \in n_2$ attempt to decode column j. If decoding fails because the column is not correctable, or if decoding succeeds but changes a symbol in an unerased row, some of the row decodings were incorrect. In this case, erase the two least reliable unerased rows (rows with the largest values of ν_i), and repeat the decoding for this column. ◇

There are many other ways of interleaving codewords. The construction of the direct product $C_1 \otimes C_2$ can also be described as follows: First interleave k_1 codewords of C_2. Then divide the interleaved vector into n_2 rows each of length k_1, extend each of these rows to a codeword of C_1 and append the additional symbols at the end of each column. In general, any combination of interleaving methods and encoding with respect to two (or more) codes is called *cross interleaving*.

Now we describe the interleaving applied for the error protection in compact discs. The method is called *cross interleaved Reed–Solomon-codes*, for short CIRC. It is a combination of three interleaving processes and encoding with respect to two Reed–Solomon-codes. CIRC involves another form of interleaving, namely, interleaving with delay $d \geq 1$, which is described below:

Ordinary n-fold interleaving of a code of length n yields blocks consisting of exactly n interleaved codewords, thus each block contains n^2 symbols. Moreover, each codeword is part of exactly one block. Interleaving with delay $d \geq 1$ is another method for interleaving a finite sequence of codewords $(c^{(r)})_{0 \leq r \leq N}$ with $c^{(r)} = (c_{rn}, c_{rn+1}, \ldots, c_{rn+(n-1)})$. As we will immediately see using interleaving with delay $d \geq 1$, a single codeword does not belong to a single block, as it is the case with product codes. Each block contains exactly n symbols which belong to n codewords. Each codeword in this sequence starts a new block and completes another block. With this method each codeword $c^{(r)}$ is spread over n different blocks.

Interleaving with delay d means that the codewords $c^{(r)}$ are inserted as certain diagonals of an array of n rows. For $i \in n$ put c_i, the i-th component of $c^{(0)}$, into the i-th row and the di-th column of this array. If the codeword $c^{(r)}$ is already inserted, then the components of $c^{(r+1)}$ are placed exactly one column to the right from the corresponding components of $c^{(r)}$. For instance, for $d = 1$ we obtain an array of the form:

$$\begin{array}{cccccccc}
c_0 & c_n & & \cdots & c_{(n-2)n} & c_{(n-1)n} & c_{nn} & c_{(n+1)n} & \cdots \\
 & c_1 & c_{n+1} & \cdots & c_{(n-3)n+1} & c_{(n-2)n+1} & c_{(n-1)n+1} & c_{nn+1} & \cdots \\
 & & c_2 & \ddots & & & & & \\
 & & & \ddots & & & & & \\
 & & & & c_{n-2} & c_{n+n-2} & c_{2n+n-2} & c_{3n+n-2} & \cdots \\
 & & & & & c_{n-1} & c_{n+n-1} & c_{2n+n-1} & \cdots
\end{array}$$

Of course blank fields at the beginning and at the end of this array must be filled with zeros. Finally, the interleaves are read as columns of the form

$$\begin{pmatrix} c_{rn} \\ c_{(r-1)n+1} \\ \vdots \\ c_{(r-(n-2))n+n-2} \\ c_{(r-(n-1))n+n-1} \end{pmatrix}.$$

In the general case for $d \geq 1$, *interleaving with delay d* yields blocks of the form

$$\begin{pmatrix} c_{rn} \\ c_{(r-d)n+1} \\ \vdots \\ c_{(r-(n-2)d)n+n-2} \\ c_{(r-(n-1)d)n+n-1} \end{pmatrix}. \qquad \textbf{5.3.18}$$

For $i \in n$ the i-th component of the word $c^{(r)}$ stands in the $r + id$-th column. Thus, the components of a single codeword occur in n blocks distributed over

$(n-1)d+1$ blocks. To be more precise, the components occur in the blocks

$$\begin{pmatrix} c_{rn} \\ c_{(r-d)n+1} \\ \vdots \\ c_{(r-(n-2)d)n+n-2} \\ c_{(r-(n-1)d)n+n-1} \end{pmatrix}, \begin{pmatrix} c_{(r+d)n} \\ c_{rn+1} \\ \vdots \\ c_{(r-(n-3)d)n+n-2} \\ c_{(r-(n-2)d)n+n-1} \end{pmatrix}, \dots, \begin{pmatrix} c_{(r+(n-1)d)n} \\ c_{(r+(n-2)d)n+1} \\ \vdots \\ c_{(r+d)n+n-2} \\ c_{rn+n-1} \end{pmatrix}.$$

By increasing d, symbols of a single codeword are spread over longer sequences of interleaved symbols, whence they are better protected against burst errors. On the other hand, deinterleaving becomes more difficult and time consuming, since more symbols must be read and kept in the memory before they can be collected to the original codewords of C.

Finally, at the end of this section we analyze certain relations between product codes and cyclic codes. These considerations are not necessary for understanding the encoding of compact discs, but they are interesting for their own sake. The reason is that the product of two cyclic codes is a cyclic code again, if the lengths of the codes are relatively prime. (See also [139, Ch. 18 §2].)

5.3.19 Lemma *Let A be an arbitrary alphabet and let n_1, n_2 be positive integers. By $A^{n_1 \times n_2}$ we denote the set of all $n_1 \times n_2$ matrices over A. If G_i is a subgroup of the symmetric group S_{n_i} for $i = 1, 2$, then the mapping*

$$(G_1 \times G_2) \times A^{n_1 \times n_2} \to A^{n_1 \times n_2} \; : \; \big((\sigma, \pi), (a_{ij})\big) \mapsto \big(a_{\sigma^{-1}(i), \pi^{-1}(j)}\big)$$

defines an action of the direct product $G_1 \times G_2$ on $A^{n_1 \times n_2}$. □

The proof of this lemma is left as an exercise for the reader.

Consider the alphabet $A = \mathbb{F}_q$, C_i a cyclic code of length n_i over $\mathbb{F}_q$ and G_i the cyclic group generated by the cycle $(0, 1, \dots, n_i - 1) \in S_{n_i}$ for $i = 1, 2$. Then for each $(\sigma, \pi) \in G_1 \times G_2$ we have

$$(c_{ij}) \in C_1 \otimes C_2 \iff (c_{\sigma^{-1}(i), \pi^{-1}(j)}) \in C_1 \otimes C_2.$$

In other words, $G_1 \times G_2$ is contained in the automorphism group of $C_1 \otimes C_2$. Moreover, if n_1 and n_2 are relatively prime, then the direct product $G_1 \times G_2$ is a cyclic group of order $n_1 n_2$.

Now we represent a codeword $c = (c_{ij}) \in C_1 \otimes C_2$ as

$$c(x, y) = \sum_{i \in n_1} \sum_{j \in n_2} c_{ij} x^i y^j + \mathrm{I}(x^{n_1} - 1, y^{n_2} - 1) \in \mathbb{F}_q[x, y] / \mathrm{I}(x^{n_1} - 1, y^{n_2} - 1).$$

Then $x \cdot c(x, y)$ and $y \cdot c(x, y)$ describe cyclic shifts of the rows and columns of $c = (c_{ij})$.

Assuming again that n_1 and n_2 are relatively prime, by the Chinese Remainder Theorem (cf. 3.5.15 and Exercise 3.5.11), we obtain that for each $(i,j) \in n_1 \times n_2$ there exists exactly one $\phi(i,j) \in n_1 n_2$ such that

$$\phi(i,j) \equiv i \bmod n_1 \text{ and } \phi(i,j) \equiv j \bmod n_2. \qquad \textbf{5.3.20}$$

Lemma 5.3.21 *Assume that n_1 and n_2 are relatively prime positive integers, and consider $a, b \in \mathbb{Z}$ such that $an_1 + bn_2 = 1$. Then:*

1. $\phi(i,j) \equiv jan_1 + ibn_2 \bmod n_1 n_2$.
2. *There exist integers $\tilde{a} > 0$ and $\tilde{b} \leq 0$ such that $\tilde{a}n_1 + \tilde{b}n_2 = 1$. If $n_1 > 1$ and $\tilde{a} > 0$, then $\tilde{b} < 0$. Moreover,* $\gcd(a,b) = \gcd(\tilde{a}, \tilde{b}) = 1$.
3. $\mathrm{I}(x^n - 1, y^m - 1) = \mathrm{I}(x^n - 1) + \mathrm{I}(y^m - 1)$ *for arbitrary $n, m > 0$.* □

The proof is left to the reader.

We claim that it is possible to rewrite $c(x,y)$ in terms of a single variable z by replacing $x^i y^j$ by $z^{\phi(i,j)}$.

Lemma 5.3.22 *Assume that n_1 and n_2 are relatively prime positive integers, and consider $a, b \in \mathbb{Z}$ such that $an_1 + bn_2 = 1$. Let $\varphi\colon \mathbb{F}_q[x,y] \to \mathbb{F}_q[z]/\mathrm{I}(z^{n_1 n_2} - 1)$ be the homomorphism defined by*

$$x \mapsto \varphi(x) := z^{bn_2} + \mathrm{I}(z^{n_1 n_2} - 1), \quad y \mapsto \varphi(y) := z^{an_1} + \mathrm{I}(z^{n_1 n_2} - 1).$$

Then:

1. $\varphi(x^i y^j) = z^{\phi(i,j)} + \mathrm{I}(z^{n_1 n_2} - 1)$ *for all $(i,j) \in n_1 \times n_2$.*
2. *φ is surjective,* $\ker \varphi = \mathrm{I}(x^{n_1} - 1, y^{n_2} - 1)$ *and*

$$\Phi\colon \mathbb{F}_q[x,y]/\mathrm{I}(x^{n_1} - 1, y^{n_2} - 1) \to \mathbb{F}_q[z]/\mathrm{I}(z^{n_1 n_2} - 1)$$
$$\Phi(f(x,y) + \mathrm{I}(x^{n_1} - 1, y^{n_2} - 1)) := \varphi(f(x,y))$$

is a ring-isomorphism.

Proof: 1. For $(i,j) \in n_1 \times n_2$ we have

$$\begin{aligned} \varphi(x^i y^j) &= \left(z^{ibn_2} + \mathrm{I}(z^{n_1 n_2} - 1)\right)\left(z^{jan_1} + \mathrm{I}(z^{n_1 n_2} - 1)\right) \\ &= z^{jan_1 + ibn_2} + \mathrm{I}(z^{n_1 n_2} - 1) = z^{\phi(i,j)} + \mathrm{I}(z^{n_1 n_2} - 1). \end{aligned}$$

2. From the definition of φ it is obvious that $\mathrm{I}(x^{n_1} - 1, y^{n_2} - 1) \subseteq \ker \varphi$. Assume, conversely, that $f(x,y) \in \mathbb{F}_q[x,y]$ belongs to $\ker \varphi$. It can be expressed as

$$f(x,y) = \sum_{i \in n_1} \sum_{j \in n_2} f_{ij} x^i y^j + \tilde{f}(x,y),$$

where $\tilde{f}(x,y) \in \mathrm{I}(x^{n_1}-1, y^{n_2}-1)$. Since φ is a homomorphism, we deduce

$$0 + \mathrm{I}(z^{n_1 n_2}-1) = \varphi(f(x,y)) = \sum_{i\in n_1}\sum_{j\in n_2} f_{ij} z^{\phi(i,j)} + \mathrm{I}(z^{n_1n_2}-1).$$

Since ϕ is a bijection between $n_1 \times n_2$ and $n_1 n_2$, all the coefficients f_{ij} vanish for $(i,j) \in n_1 \times n_2$ and, consequently, $f(x,y) = \tilde{f}(x,y) \in \mathrm{I}(z^{n_1n_2}-1)$. Obviously, φ is surjective, whence Φ is an isomorphism. □

This way we obtain

$$\Phi(c(x,y) + \mathrm{I}(x^{n_1}-1, y^{n_2}-1)) = \sum_{i\in n_1}\sum_{j\in n_2} c_{ij} z^{\phi(i,j)} + \mathrm{I}(z^{n_1n_2}-1),$$

which allows us to determine $c(z)$ as

$$c(z) = \sum_{i\in n_1}\sum_{j\in n_2} c_{ij} z^{\phi(i,j)},$$

as was claimed.

If $c \in C_1 \otimes C_2$, then $zc(z) + \mathrm{I}(z^{n_1n_2}-1) = \Phi(xy)\Phi(c(x,y)) = \Phi(xyc(x,y))$ and $xyc(x,y) \in C_1 \otimes C_2$, whence $C_1 \otimes C_2$ is an ideal in $\mathbb{F}_q[z]/\mathrm{I}(z^{n_1n_2}-1)$. In other words, using an appropriate order of the canonical basis vectors of $C_1 \otimes C_2$, the product code is cyclic: We associate $c' \otimes c'' \in C_1 \otimes C_2$ with the vector

$$c := (c_{\phi^{-1}(0)}, \ldots, c_{\phi^{-1}(n_1n_2-1)}),$$

where $c_{ij} = c'_i c''_j$ for $i \in n_1$, $j \in n_2$.

This proves the first assertion of

5.3.23 Theorem [34], [135] *Assume that C_i is a cyclic linear (n_i, k_i, d_i, q)-code with generator polynomial g_i and check polynomial h_i for $i = 1, 2$. Suppose that $n_1 > 1$ and n_2 are relatively prime and that $a > 0$ and $b \leq 0$ are integers such that $an_1 + bn_2 = 1$. Then:*

1. *The product code $C_1 \otimes C_2$ is a cyclic code.*
2. *The generator polynomial of $C_1 \otimes C_2$ is*
$$g(z) = \gcd\left(z^{n_1n_2}-1, (z^{\ell n_1 n_2} g_1(z^{bn_2}))g_2(z^{an_1})\right)$$
with $\ell = 2(-b)$.
3. *The check polynomial of $C_1 \otimes C_2$ is*
$$h(z) = \gcd\left(z^{mn_1n_2} h_1(z^{bn_2}), h_2(z^{an_1})\right)$$
with $m = -b$.
4. *If e_i is the idempotent generator of C_i for $i = 1, 2$, then $\Phi(e_1 e_2)$ is the idempotent generator of $C_1 \otimes C_2$.*

Proof: 2. The integer ℓ is chosen so that $z^{\ell n_1 n_2} f(z^{bn_2})$ is a polynomial in z for all polynomials f of degree not greater than $2n_1$. Assume that $c = (c_{ij})$ belongs to $C_1 \otimes C_2$. Let $c(z)$ be the uniquely determined polynomial of degree less than $n_1 n_2$ associated with c. The i-th row of c belongs to C_2, whence adding suitable multiples of $y^{n_2} - 1$ we obtain a polynomial

$$\zeta_i(y) \equiv \sum_{j \in n_2} c_{ij} y^j \bmod \mathrm{I}(y^{n_2} - 1), \qquad i \in n_1,$$

such that the generator polynomial $g_2(y)$ is a divisor of $\zeta_i(y)$ in $\mathbb{F}_q[y]$. Similarly, the j-th column of c belongs to C_1, whence adding suitable multiples of $x^{n_1} - 1$ we obtain a polynomial

$$\sigma_j(x) \equiv \sum_{i \in n_1} c_{ij} x^i \bmod \mathrm{I}(x^{n_1} - 1), \qquad j \in n_2,$$

such that the generator polynomial $g_1(x)$ is a divisor of $\sigma_j(x)$ in $\mathbb{F}_q[x]$. It is always possible to find polynomials σ_j of degree less than $2n_1$. Consequently, after multiplying with $z^{\ell n_1 n_2}$ we obtain

$$c(z) \equiv \sum_{i \in n_1} \zeta_i(z^{an_1}) z^{ibn_2 + \ell n_1 n_2} \equiv \sum_{j \in n_2} \sigma_j(z^{bn_2}) z^{jan_1 + \ell n_1 n_2} \bmod \mathrm{I}(z^{n_1 n_2} - 1).$$

Thus, $c(z)$ can be expressed in two ways

$$\begin{aligned} c(z) &= q(z)(z^{n_1 n_2} - 1) + \sum_{i \in n_1} \zeta_i(z^{an_1}) z^{ibn_2 + \ell n_1 n_2}, \\ c(z) &= \tilde{q}(z)(z^{n_1 n_2} - 1) + \sum_{j \in n_2} \sigma_j(z^{bn_2}) z^{jan_1 + \ell n_1 n_2}. \end{aligned}$$

For this reason $\gcd(z^{n_1 n_2} - 1, g_2(z^{an_1}))$ and $\gcd(z^{n_1 n_2} - 1, g_1(z^{bn_1}) z^{\ell n_1 n_2})$ are divisors of $c(z)$ for all $c \in C_1 \otimes C_2$. Hence,

$$L(z) := \operatorname{lcm}\big(\gcd(z^{n_1 n_2} - 1, g_2(z^{an_1})), \gcd(z^{n_1 n_2} - 1, g_1(z^{bn_1}) z^{\ell n_1 n_2})\big)$$

is a divisors of the generator polynomial $g(z)$.

Now assume that $c_1 = (c_0^{(1)}, \ldots, c_{n_1 - 1}^{(1)})$ and $c_2 = (c_0^{(2)}, \ldots, c_{n_2 - 1}^{(2)})$ are the codewords of C_1 and C_2 corresponding to the generator polynomials $g_1(x) = \sum_{i \in n_1} c_i^{(1)} x^i$ and $g_2(y) = \sum_{i \in n_2} c_i^{(2)} y^i$. Then

$$\begin{aligned} (c_1 \otimes c_2)(z) &\equiv \sum_{i \in n_1} \sum_{j \in n_2} c_i^{(1)} c_j^{(2)} z^{jan_1 + ibn_2 + \ell n_1 n_2} \bmod \mathrm{I}(z^{n_1 n_2} - 1) \\ &\equiv z^{\ell n_1 n_2} \sum_{i \in n_1} c_i^{(1)} z^{ibn_2} \sum_{j \in n_2} c_j^{(2)} z^{jan_1} \bmod \mathrm{I}(z^{n_1 n_2} - 1) \\ &\equiv z^{\ell n_1 n_2} g_1(z^{bn_2}) g_2(z^{an_1}) \bmod \mathrm{I}(z^{n_1 n_2} - 1). \end{aligned}$$

After adding suitable multiples of $z^{n_1 n_2} - 1$ to $(c_1 \otimes c_2)(z)$, we deduce that $g(z)$ is a divisor of $(c_1 \otimes c_2)(z)$. Since $g(z)$ is also a divisor of $z^{n_1 n_2} - 1$, it follows that $g(z)$ divides $z^{\ell n_1 n_2} g_1(z^{bn_2}) g_2(z^{an_1})$, whence it is a divisor of

$$G(z) := \gcd\big(z^{n_1 n_2} - 1, z^{\ell n_1 n_2} g_1(z^{bn_2}) g_2(z^{an_1})\big).$$

Summarizing, so far we have deduced that $L(z) \mid g(z) \mid G(z)$. Finally, we want to prove that $L(z) = G(z)$. The polynomials $L(z)$ and $G(z)$ have the same irreducible factors. If $L(z)$ were a proper divisor of $G(z)$, then there exists an irreducible factor of $z^{n_1 n_2} - 1$ which is both a factor of $z^{\ell n_1 n_2} g_1(z^{bn_2})$ and $g_2(z^{an_1})$ which occurs in $G(z)$ with a greater multiplicity than in $L(z)$. Then necessarily $n_1 n_2 = p^s n$, where p is the characteristics of $\mathbb{F}_q$, $s > 0$, and $\gcd(n, p) = 1$. Thus $z^{n_1 n_2} - 1 = (z^n - 1)^{p^s}$. Since n_1 and n_2 are relatively prime, either p is a divisor of n_1 or of n_2. If $p \mid n_1$, then $n_1 = p^s n_1'$ and $g_2(z^{an_1}) = \big(g_2(z^{an_1'})\big)^{p^s}$. Hence, each common factor of $z^{n_1 n_2} - 1$ and $g_2(z^{an_1})$ occurs with the multiplicity p^s both in L and G. If $p \mid n_2$, then $n_2 = p^s n_2'$ and $z^{\ell n_1 n_2} g_1(z^{bn_2}) = \big(z^{\ell n_1 n_2'} g_1(z^{bn_2'})\big)^{p^s}$. Hence, each common factor of $z^{n_1 n_2} - 1$ and $z^{\ell n_1 n_2} g_1(z^{bn_2})$ occurs with the multiplicity p^s both in L and G. This proves the second assertion.

3. From the representation of the generator polynomial g in 2. and Bézout's Identity (cf. Exercise 3.1.6), we derive that

$$g(z) = f_1(z)(z^{n_1 n_2} - 1) + f_2(z) z^{\ell n_1 n_2} g_1(z^{bn_2}) g_2(z^{an_1})$$

for some $f_1, f_2 \in \mathbb{F}_q[z]$. We want to prove that $h(z)$ is a divisor of $z^{mn_1 n_2} h_1(z^{bn_2})$ and of $h_2(z^{an_1})$. Since $g(z)h(z) = z^{n_1 n_2} - 1$, it is enough to show that $z^{n_1 n_2} - 1$ is a divisor of $g(z) z^{mn_1 n_2} h_1(z^{bn_2})$ and of $g(z) h_2(z^{an_1})$. The first assertion is proved by

$$g(z) z^{mn_1 n_2} h_1(z^{bn_2}) =$$
$$f_1(z)(z^{n_1 n_2} - 1) z^{mn_1 n_2} h_1(z^{bn_2}) + f_2(z) z^{\ell n_1 n_2} g_1(z^{bn_2}) g_2(z^{an_1}) z^{mn_1 n_2} h_1(z^{bn_2}) =$$
$$f_1(z)(z^{n_1 n_2} - 1) z^{mn_1 n_2} h_1(z^{bn_2}) + f_2(z) g_2(z^{an_1}) z^{(\ell + m + b) n_1 n_2} (1 - z^{-bn_1 n_2})$$

what follows from 4.2.3. Indeed, $z^{n_1 n_2} - 1$ is a factor of the first and of the second summand, since $b < 0$. Similarly, we prove that $z^{n_1 n_2} - 1$ is a divisor of $g(z) h_2(z^{an_1})$.

Thus, $h(z)$ is a divisor of $H(z) := \gcd\big(z^{mn_1 n_2} h_1(z^{bn_2}), h_2(z^{an_1})\big)$. Now we prove that $H(z)$ is a divisor of $z^{n_1 n_2} - 1$. If ξ is a root of H in a suitable extension field, then $\xi \neq 0$, since $h_2(0) \neq 0$. Consequently $h_1(\xi^{bn_2}) = 0 = h_2(\xi^{an_1})$. This implies that ξ^{bn_2} is a root of $z^{n_1} - 1$, and ξ^{an_1} is a root if $z^{n_2} - 1$. Consequently $\xi^{bn_1 n_2} = 1 = \xi^{an_1 n_2}$. In other words, $(\xi^{n_1 n_2})^a = 1 = (\xi^{n_1 n_2})^b$, from which we finally obtain that $1 = (\xi^{n_1 n_2})^{(n_1 a + n_2 b)} = \xi^{n_1 n_2}$. Hence, ξ is a root of $z^{n_1 n_2} - 1$. It is easy to prove that $h_2(z^{an_1})$ divides $z^{an_1 n_2} - 1$ and $z^{mn_1 n_2} h_1(z^{bn_2})$ divides $z^{-bn_1 n_2} - 1$. If ξ is a root of $H(z)$, then the minimal polynomial M_ξ of ξ over $\mathbb{F}_q$ is an irreducible factor of $z^{n_1 n_2} - 1$, of $z^{an_1 n_2} - 1$ and of $z^{-bn_1 n_2} - 1$. Since a and b are relatively prime, the multiplicity of M_ξ in $H(z)$ is not greater than the multiplicity of M_ξ in $z^{n_1 n_2} - 1$. Therefore, $H(z)$ is a divisor of $z^{n_1 n_2} - 1$. The decomposition of $z^{n_1 n_2} - 1$ into linear factors is completely described in

Exercise 5.3.6. Each root of H can be expressed as the product $\alpha\beta$ of roots of 1 of order n_1 and n_2.

The common roots of $z^{mn_1n_2}h_1(z^{bn_2})$ and $z^{n_1n_2}-1$ are of the form $\alpha\beta$ where α is a root of h_1 and $\beta^{n_2}=1$. Similarly, the common roots of $h_2(z^{an_2})$ and $z^{n_1n_2}-1$ are of the form $\alpha\beta$ where β is a root of h_2 and $\alpha^{n_1}=1$.

Assume that $p \nmid n_1n_2$. Then there exist exactly k_1 distinct roots α of h_1 and k_2 distinct roots β of h_2. Moreover, $\alpha\beta$ is a root of H if and only if α is a root of h_1 and β is a root of h_2. Hence, each pair (α,β) of these roots determines uniquely a root $\alpha\beta$ of H. Consequently $\deg H = k_1k_2$ and, therefore, $H=h$, since h is the check polynomial of a code of dimension k_1k_2.

Assume that $p \mid n_2$. Then $\gcd(an_1,p)=1$. From Exercise 5.3.7 we deduce: If β is a root of h_2 of multiplicity r and $\alpha^{n_1}=1$, then $\alpha\beta$ is a root of $h_2(z^{an_1})$ of the same multiplicity r. Consequently, $\alpha\beta$ is a root of H if and only if α is a root of h_1 and β is a root of h_2. Moreover, the multiplicity of $\alpha\beta$ as a root of H is at most the multiplicity of β as a root of h_2. Hence, $\deg H \le k_1k_2$ and, therefore, $H=h$.

Finally, assume that $p \mid n_1$. Then $\gcd(bn_2,p)=1$. For $\xi \neq 0$ it is easy to prove that ξ is a root of $z^{mn_1n_2}h_1(z^{bn_2})$ of multiplicity r if and only if ξ^{-1} is a root of $h_1(z^{-bn_2})$ of multiplicity r. Similar arguments as above show that also in this case $H=h$.

4. In order to prove the last assertion we derive

$$\Phi\big(e_1(x)e_2(y)\big)^2 = \Phi\big(e_1(x)^2e_2(y)^2\big) = \Phi\big(e_1(x)e_2(y)\big),$$

whence $\Phi\big(e_1(x)e_2(y)\big)$ is an idempotent element of $\mathbb{F}_q[z]/\mathrm{I}(z^{n_1n_2}-1)$. Assume that $f(z) \in \mathbb{F}_q[x]/\mathrm{I}(z^{n_1n_2}-1)$. Then there exists a unique $\tilde{f}(x,y) \in \mathbb{F}_q[x,y]/\mathrm{I}(x^{n_1}-1,y^{n_2}-1)$ such that $f(z)=\Phi\big(\tilde{f}(x,y)\big)$. Since n_1 and n_2 are relatively prime there exist $f_1(x) \in \mathbb{F}_q[x]/\mathrm{I}(x^{n_1}-1)$ and $f_2(y) \in \mathbb{F}_q[y]/\mathrm{I}(y^{n_2}-1)$ such that $\tilde{f}(x,y)=f_1(x)f_2(y)$. Since e_i is a generator of C_i, for $i=1,2$, there are $r(x) \in \mathbb{F}_q[x]/\mathrm{I}(x^{n_1}-1)$ and $s(y) \in \mathbb{F}_q[y]/\mathrm{I}(y^{n_2}-1)$ such that $f_1(x) = e_1(x)r(x)$ and $f_2(y)=e_2(y)s(y)$. Consequently,

$$\begin{aligned} f(z) &= \Phi\big(\tilde{f}(x,y)\big) = \Phi\big(f_1(x)f_2(y)\big) \\ &= \Phi\big(e_1(x)r(x)e_2(y)s(y)\big) = \Phi\big(e_1(x)e_2(y)\big)\Phi\big(r(x)s(y)\big), \end{aligned}$$

which finishes the proof. □

Example [139, Ch. 18 §2] Let C_1 be the cyclic binary $(3,2,2)$-code with generator polynomial $g_1(x)=x+1$, check polynomial $h_1(x)=x^2+x+1$, and idempotent generator $e_1(x)=x^2+x$. And let C_2 be the cyclic binary $(5,4,2)$-code with generator polynomial $g_2(y)=y+1$, check polynomial $h_2(y)=y^4+y^3+y^2+y+1$, and idempotent generator $e_2(y)=y^4+y^3+y^2+y$. **5.3.24**

Since 3 and 5 are relatively prime, $C_1 \otimes C_2$ is a cyclic $(15,8,4)$-code. With $a = 2$ and $b = -1$ we get $\ell = 2$, $m = 1$, and the generator polynomial

$$g(z) = \gcd\big(z^{15}-1, z^{30} g_1(z^{-5}) g_2(z^6)\big) = z^7 + z^6 + z^5 + z^2 + z + 1.$$

Moreover, the check polynomial of $C_1 \otimes C_2$ is

$$h(z) = \gcd\big(z^{15} h_1(z^{-5}), h_2(z^6)\big) = z^8 + z^7 + z^5 + z^4 + z^3 + z.$$

Of course $h(z) = (z^{15}+1)/g(z)$. Finally, the idempotent generator can be determined by

$$\begin{aligned} e(z) &= \Phi(e_1(x)e_2(y)) = \Phi(x^2y^4 + xy^4 + x^2y^3 + xy^3 + x^2y^2 + xy^2 + x^2y + xy) \\ &= z + z^2 + z^4 + z^7 + z^8 + z^{11} + z^{13} + z^{14}. \end{aligned}$$ ◇

However, not all cyclic codes are products of cyclic codes (cf. [139, Ch. 18 §3]). Let C be a minimal binary cyclic (n,k)-code with $n = n_1 n_2$, $\gcd(n_1 n_2, 2) = 1$, $\gcd(n_1, n_2) = 1$, and $n_1 > 1$, $n_2 > 1$. Since C is minimal, its check polynomial h is irreducible, and its roots are of the form $\xi, \xi^2, \ldots, \xi^{2^{k-1}}$, where ξ is an n-th root of 1 in $\mathbb{F}_{2^k}$. Moreover, we assume that h is a primitive polynomial, whence ξ is a primitive element of $\mathbb{F}_{2^k}^*$.

Since $\gcd(n_1, n_2) = 1$, there exist integers a, b such that $an_1 + bn_2 = 1$. Define $\alpha := \xi^{bn_2}$, $\beta := \xi^{an_1}$, and let k_1, k_2 be the least integers for which $\alpha \in \mathbb{F}_{2^{k_1}}$ and $\beta \in \mathbb{F}_{2^{k_2}}$. Clearly k_1 and k_2 are divisors of k. In fact, $k = \operatorname{lcm}(k_1, k_2)$. Based on these assumptions we can prove the next

5.3.25 Theorem [139, Ch. 18 §3] *Let C be a minimal binary cyclic (n,k)-code with $n = n_1 n_2$, $\gcd(n_1 n_2, 2) = 1$, $\gcd(n_1, n_2) = 1$, and $n_1 > 1$, $n_2 > 1$. There exist binary cyclic (n_i, k_i)-codes C_i such that $C = C_1 \otimes C_2$ if and only if $\gcd(k_1, k_2) = 1$.*

Proof: Since Φ is a ring isomorphism, we obtain from $c \in C$ an $n_1 \times n_2$-array (f_{ij}) by

$$\Phi^{-1}\big(c(z)\big) = \sum_{i \in n_1} \sum_{j \in n_2} f_{ij} x^i y^j + \mathrm{I}(x^{n_1}-1, y^{n_2}-1) = f(x,y) + \mathrm{I}(x^{n_1}-1, y^{n_2}-1).$$

Moreover, $\Phi^{-1}(z^{an_1} + \mathrm{I}(z^{n_1 n_2}-1)) = y + \mathrm{I}(x^{n_1}-1, y^{n_2}-1)$ and analogously $\Phi^{-1}(z^{bn_2} + \mathrm{I}(z^{n_1 n_2}-1)) = x + \mathrm{I}(x^{n_1}-1, y^{n_2}-1)$. Since C is cyclic, $z^{an_1} c(z) \in C$ for all $c \in C$, whence $yf(x,y) + \mathrm{I}(x^{n_1}-1, y^{n_2}-1) = \Phi^{-1}\big(z^{an_1} c(z)\big) \in \Phi^{-1}(C)$, and similarly $xf(x,y) + \mathrm{I}(x^{n_1}-1, y^{n_2}-1) \in \Phi^{-1}(C)$. Therefore, the two sets

$$C_1 := \left\{ (f_{0j}, \ldots, f_{n_1-1,j}) \;\middle|\; j \in n_2,\ \sum_{i \in n_1} \sum_{j \in n_2} f_{ij} x^i y^j + \mathrm{I} \in \Phi^{-1}(C) \right\}$$

and

$$C_2 := \left\{ (f_{i0}, \ldots, f_{i,n_2-1}) \;\middle|\; i \in n_1,\ \sum_{i \in n_1} \sum_{j \in n_2} f_{ij} x^i y^j + \mathrm{I} \in \Phi^{-1}(C) \right\}$$

are cyclic codes of length n_1 and n_2, where $\mathrm{I} = \mathrm{I}(x^{n_1} - 1, y^{n_2} - 1)$.

For α and β as above, we have $\alpha^{n_1} = \beta^{n_2} = 1$ and $\xi = \alpha\beta$. Hence, the roots of h are

$$\alpha\beta, (\alpha\beta)^2, (\alpha\beta)^4, \ldots, (\alpha\beta)^{2^{k-1}}.$$

We still have to determine the dimensions of these codes. Our claim is that $\dim(C_i) = k_i$ for $i = 1, 2$, where k_1, k_2 are the least integers for which $\alpha \in \mathbb{F}_{2^{k_1}}$ and $\beta \in \mathbb{F}_{2^{k_2}}$. We prove that $\alpha, \alpha^2, \ldots, \alpha^{2^{k_1-1}}$ are k_1 zeros of the check polynomial h_1 of C_1 and $\beta, \beta^2, \ldots, \beta^{2^{k_2-1}}$ are k_2 zeros of the check polynomial h_2 of C_2. Let $\beta_0 \in \mathbb{F}_{2^{k_2-1}}$ be an n_2-th root of 1 not belonging to $\{\beta, \beta^2, \ldots, \beta^{2^{k_2-1}}\}$, and let α_0 be any n_1-th root of 1. Then $\alpha_0\beta_0$ is a root of $g(z)$, since $g(z) = (z^n + 1)/h(z)$. Let $f(x, y) = r_0(y) + r_1(y)x + \ldots + r_{n_1-1}(y)x^{n_1-1}$ correspond to a nonzero codeword of C, where r_i are codewords of C_2. Then

$$f(\alpha_0, \beta_0) = \sum_{i \in n_1} r_i(\beta_0)\alpha_0^i = 0.$$

This holds true for any root α_0 satisfying $\alpha_0^{n_1} = 1$. Since there exist n_1 different α_0, the values $r_i(\beta_0)$, $i \in n_1$, satisfy a system of n_1 linear homogeneous equations. The coefficient matrix is a Vandermonde matrix, thus it is regular and $r_i(\beta_0) = 0$ for $i \in n_1$. Therefore, β_0 is a root of each codeword of C_2, whence it is a root of the generator polynomial g_2, and consequently not a root of h_2.

Since $g(\alpha^{2^j}\beta^{2^j}) \neq 0$ for $j \in k_2$, there exist codewords $r_i \in C_2$ such that $r_i(\beta^{2^j}) \neq 0$. Whence, the roots of h_2 are exactly given by β^{2^j} for $j \in k_2$. Similarly, the roots of h_1 can be determined.

In conclusion we obtain: If $\gcd(k_1, k_2) = 1$, then $k = \mathrm{lcm}(k_1, k_2) = k_1k_2$ and $C = C_1 \otimes C_2$. Conversely, if $C = C_1 \otimes C_2$, then $k = \mathrm{lcm}(k_1, k_2) = k_1k_2$, whence $\gcd(k_1, k_2) = 1$. □

Examples [139, Ch. 18 §3] **5.3.26**

1. Let C be the cyclic binary $(15, 4, 8)$-code with generator polynomial $g(z) = z^{11} + z^8 + z^7 + z^5 + z^3 + z^2 + z + 1$ and idempotent generator $e(z) = zg(z)$. Let $n_1 = 3$ and $n_2 = 5$. The check polynomial of C is irreducible of degree 4 and its roots are of the form ξ, ξ^2, ξ^4, ξ^8 with $\xi^{15} = 1$. Then $a = 2$, $b = -1$, $\alpha = \xi^{-5} = \xi^{10}$, $\beta = \xi^6$, $\alpha^4 = \alpha$, $\beta^{16} = \beta$, $k_1 = 2$, and $k_2 = 4$. Since $\gcd(2, 4) \neq 1$, the product $C_1 \otimes C_2$ is different from C. Indeed C_1 is a $(3, 2, 2)$-code and C_2 is a $(5, 4, 2)$-code.

2. Let C be the $(21,6,8)$-product code of Exercise 5.3.8 where $\xi, \xi^2, \xi^4, \xi^8, \xi^{16}, \xi^{32} = \xi^{11}$ are the roots of its check polynomial, with $\xi^{21} = 1$. Then $n_1 = 3$, $n_2 = 7$, $a = 5$, $b = -2$, $\alpha = \xi^{-14} = \xi^7$, $\beta = \xi^{15}$, $k_1 = 2$, $k_2 = 3$, and C is the product code of C_1 and C_2 as given in Exercise 5.3.8. ◇

Exercises

E.5.3.1 **Exercise** Prove 5.3.9.

E.5.3.2 **Exercise** Prove 5.3.11.

E.5.3.3 **Exercise** Prove that all the rows of a matrix representing an element of the product code $C_1 \otimes C_2$ are elements of C_2.

E.5.3.4 **Exercise** Prove 5.3.19.

E.5.3.5 **Exercise** Assume that n_1 and n_2 are relatively prime positive integers, σ and π are given as in 5.3.19 and ϕ satisfies 5.3.20. Show that

$$\phi\big(\sigma(i), \pi(j)\big) \equiv \phi(i,j) + 1 \bmod n_1 n_2.$$

E.5.3.6 **Exercise** Let p be a prime and assume that n_1 and n_2 are relatively prime positive integers. Prove the following assertions.

1. If $\gcd(p, n_1 n_2) = 1$, then
$$z^{n_1 n_2} - 1 = \prod_{i \in n_1} \prod_{j \in n_2} (z - \alpha^i \beta^j),$$
where α and β are primitive roots of 1 of order n_1 and n_2, respectively.
2. If $p \mid n_1 n_2$ assume without loss of generality that p is a divisor of n_1, and $n_1 = n_1' p^s$ with $s > 0$ and $\gcd(n_1', p) = 1$. Then
$$z^{n_1 n_2} - 1 = \Big(\prod_{i \in n_1'} \prod_{j \in n_2} (z - \alpha^i \beta^j) \Big)^{p^s},$$
where α and β are primitive roots of 1 of order n_1' and n_2, respectively.

E.5.3.7 **Exercise** Let $h \in \mathbb{F}_q[x]$ be a divisor of $x^m - 1$ and $n \in \mathbb{N}$ with $\gcd(n, m) = 1$ and $\gcd(n, p) = 1$, where p is the characteristic of $\mathbb{F}_q$. Prove that all the roots of $h(x^n)$ are of the form $\alpha\beta$, where α is a root of h and $\beta^n = 1$. If, moreover, α occurs with the multiplicity r in h, then each $\alpha\beta$ occurs as a root of $h(x^n)$ with the same multiplicity r.

Exercise [139, Ch. 18 §2] Let C_1 be the cyclic $(3,2,2,2)$-code of 5.3.24 and let C_2 be the cyclic $(7,3,4,2)$-code S_3 of 4.2.7. Compute the parameters, generator and check polynomial and the idempotent generator of $C_1 \otimes C_2$. E.5.3.8

5.4 More Details on Compact Discs 5.4

We already know that an audio compact disc contains 16-bit audio which is sampled at a rate of 44 100 Hz. Each sampling is divided into two 8-bit vectors, and each of these vectors is considered as one byte and also as an element of $\mathbb{F}_{2^8}$. Thus, each sampling process yields 2 bytes of audio information. Since both the left and the right channel are sampled separately, each sampling produces two bytes of audio information for the left and two bytes of audio information for the right channel. Hence, every second we collect $2 \cdot 2 \cdot 44\,100 = 176\,400$ bytes of audio information. This gives 10 584 000 bytes per minute and 635 040 000 bytes or 5 080 320 000 bits per hour. As we will see the total capacity required for storing this information on a compact disc is approximately three times as big. If not otherwise specified we excerpt or "quote" from the third chapter of [164]. "Storing audio information places great demands on a digital medium. [...] Error correction, synchronization and modulation are required for successful storage." "The compact disc was developed in order to meet" different user demands as "random access, small size, convenience to use, robustness, low cost, and ease of replication." The specifications for the compact disc system "were jointly developed by Philips and Sony and are defined in [...] the *Red Book*." (The *Red Book* is the 1980 document which provides the specifications for the standard compact disc (CD) developed by Sony and Philips. According to legend, the document was in a binder with red covers, originating the tradition for subsequent adaptations of CD specifications to be referred to as variously colored books. The *Red Book* describes the compact discs physical specifications, such as the tracks, sector and block layout, coding, and sampling. Sony and Philips referred to the discs as CD-DA (digital audio), defined as a content medium for audio data digitized at 44 100 samples per second and in a range of 65 536 possible values cf. [166]). "It is also contained in the IEC standard *Compact Disc Digital Audio* [95]." A compact disc allows us to store at least 74 minutes of stereo high fidelity audio. The disc must be made of transparent material with a refraction index of 1.55. "The optical system that reads the data from the disc uses a laser beam with a 780 nanometer wavelength." All the information of a compact disc is stored in the area between radius 23 mm and 58.5 mm. A lead-in and lead-out area cover the innermost and outermost part of this area. They do not con-

tain any audio information. The audio data is stored between radius 25 mm and 58 mm. In general, the information of a compact disc is stored in a track in form of "a continuous spiral running from the inner circumference to the outer." "Viewed from the readout surface the disc rotates counter-clockwise." The distance between successive tracks is 1.6 micrometers. "There are 22 188 revolutions across the discs surface." The rotational speed of a compact disc varies on the position of the pickup. "The disc rotates at a speed of 500 rotations per minute when the pickup is reading the inner circumference, and as the pickup moves out, the rotational speed decreases to 200 rotations per minute. Thus, a *constant linear velocity*, CLV, is maintained." Depending on the disc, this velocity can vary between 1.2 and 1.4 meters per second. The CD-player automatically regulates the disc rotational speed to maintain a constant bit rate of 4.3218 Mb/sec (cf. 5.4.9). That kind of track is also called a CLV servo system; i.e. the player constantly reads synchronization words from the data and adjusts the speed accordingly. Audio data is stored in a frame format on the disc. Among other information, each frame contains exactly 24 audio bytes. Consequently, there are exactly 7350 frames per second. Further details about frames will be presented later (cf. 5.4.9). First we analyze the error detection and correction process used for compact discs. The analog to digital converter produces sequences of audio data in PCM format. This way we obtain finite sequences $(L_{i,A})_{0\le i\le N}$, $(L_{i,B})_{0\le i\le N}$, $(R_{i,A})_{0\le i\le N}$, and $(R_{i,B})_{0\le i\le N}$ of bytes representing the first (A) or the second (B) byte of the left (L) or right (R) channel at the i-th sampling for $0 \le i \le N$, where N is the number of the last sampling. With L_i or R_i we denote the i-th sampling of the left or right channel, i.e. the pair $(L_{i,A}, L_{i,B})$ or $(R_{i,A}, R_{i,B})$, respectively. The analog to digital converter produces two sequences $(L_i)_{0\le i\le N}$ and $(R_i)_{0\le i\le N}$. Now we want to describe the CIRC-process in more details. Even though this process is usually illustrated with diagrams, we try to use common mathematical notation.

5.4.1 **CIRC encoding** In *step A* this data is scrambled into a series of vectors containing 24 bytes. The vectors obtained from the first samplings are the rows of the following array:

$$\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & L_1 & L_3 & L_5 & R_1 & R_3 & R_5 \\
0 & 0 & 0 & 0 & 0 & 0 & L_7 & L_9 & L_{11} & R_7 & R_9 & R_{11} \\
L_0 & L_2 & L_4 & R_0 & R_2 & R_4 & L_{13} & L_{15} & L_{17} & R_{13} & R_{15} & R_{17} \\
L_6 & L_8 & L_{10} & R_6 & R_8 & R_{10} & L_{19} & L_{21} & L_{23} & R_{19} & R_{21} & R_{23} \\
\dots & & & & & & & & & & & \dots
\end{array}$$

In general, for $n \ge 0$ the n-th vector is built as

$$\begin{array}{cccccc}
L_{6(n-2)} & L_{6(n-2)+2} & L_{6(n-2)+4} & R_{6(n-2)} & R_{6(n-2)+2} & R_{6(n-2)+4} \\
L_{6n+1} & L_{6n+3} & L_{6n+5} & R_{6n+1} & R_{6n+3} & R_{6n+5}.
\end{array}$$

The first half of this vector contains the sampling values of even samples, the second half of odd samples. Of course, at the very beginning and at the very end empty fields must be filled with zero bytes. Using a $(28, 24)$ Reed–Solomon-code C_2 over $\mathbb{F}_{2^8}$, in *step B* this vector is encoded into the C_2-codeword

$$\begin{array}{cccc} L_{6(n-2),A} & L_{6(n-2),B} & L_{6(n-2)+2,A} & L_{6(n-2)+2,B} \\ L_{6(n-2)+4,A} & L_{6(n-2)+4,B} & R_{6(n-2),A} & R_{6(n-2),B} \\ R_{6(n-2)+2,A} & R_{6(n-2)+2,B} & R_{6(n-2)+4,A} & R_{6(n-2)+4,B} \\ Q_{n,0} & Q_{n,1} & Q_{n,2} & Q_{n,3} \\ L_{6n+1,A} & L_{6n+1,B} & L_{6n+3,A} & L_{6n+3,B} \\ L_{6n+5,A} & L_{6n+5,B} & R_{6n+1,A} & R_{6n+1,B} \\ R_{6n+3,A} & R_{6n+3,B} & R_{6n+5,A} & R_{6n+5,B}, \end{array} \tag{5.4.2}$$

where the four new symbols (they are usually denoted by the letter Q) are inserted in the middle of the vector, i.e. between the audio information of the even and odd samples. (In general, these codewords are just sequences of 28 bytes. To increase the readability they were arranged in form of an array.) The code C_2 is obtained by shortening a $(255, 251, 5, 2^8)$-Reed–Solomon-code, which is a shortened BCH-code and also an MDS-code (cf. Exercise 5.4.1).

In *step C* these C_2-codewords are interleaved with delay $d = 4$. This yields, according to 5.3.18, vectors of the form

$$\begin{array}{cccc} L_{6(n-2),A} & L_{6(n-2-d),B} & L_{6(n-2-2d)+2,A} & L_{6(n-2-3d)+2,B} \\ L_{6(n-2-4d)+4,A} & L_{6(n-2-5d)+4,B} & R_{6(n-2-6d),A} & R_{6(n-2-7d),B} \\ R_{6(n-2-8d)+2,A} & R_{6(n-2-9d)+2,B} & R_{6(n-2-10d)+4,A} & R_{6(n-2-11d)+4,B} \\ Q_{n-12d,0} & Q_{n-13d,1} & Q_{n-14d,2} & Q_{n-15d,3} \\ L_{6(n-16d)+1,A} & L_{6(n-17d)+1,B} & L_{6(n-18d)+3,A} & L_{6(n-19d)+3,B} \\ L_{6(n-20d)+5,A} & L_{6(n-21d)+5,B} & R_{6(n-22d)+1,A} & R_{6(n-23d)+1,B} \\ R_{6(n-24d)+3,A} & R_{6(n-25d)+3,B} & R_{6(n-26d)+5,A} & R_{6(n-27d)+5,B}. \end{array}$$

Another Reed–Solomon-code C_1, a $(32, 28)$-code over $\mathbb{F}_{2^8}$, is used *in step D* to encode these vectors as C_1-codewords. Again we have to attach 4 bytes, this time they are appended at the end of the vector. In the literature these bytes are usually indicated with the letter P. We obtain

$$\begin{array}{cccc} L_{6(n-2),A} & L_{6(n-2-d),B} & L_{6(n-2-2d)+2,A} & L_{6(n-2-3d)+2,B} \\ L_{6(n-2-4d)+4,A} & L_{6(n-2-5d)+4,B} & R_{6(n-2-6d),A} & R_{6(n-2-7d),B} \\ R_{6(n-2-8d)+2,A} & R_{6(n-2-9d)+2,B} & R_{6(n-2-10d)+4,A} & R_{6(n-2-11d)+4,B} \\ Q_{n-12d,0} & Q_{n-13d,1} & Q_{n-14d,2} & Q_{n-15d,3} \\ L_{6(n-16d)+1,A} & L_{6(n-17d)+1,B} & L_{6(n-18d)+3,A} & L_{6(n-19d)+3,B} \\ L_{6(n-20d)+5,A} & L_{6(n-21d)+5,B} & R_{6(n-22d)+1,A} & R_{6(n-23d)+1,B} \\ R_{6(n-24d)+3,A} & R_{6(n-25d)+3,B} & R_{6(n-26d)+5,A} & R_{6(n-27d)+5,B} \\ P_{n,0} & P_{n,1} & P_{n,2} & P_{n,3}. \end{array} \tag{5.4.3}$$

Also this code is obtained by shortening a $(255, 251, 5, 2^8)$-Reed–Solomon code.

Finally, in *step E* all bytes in odd positions, i.e. in position $1, 3 \ldots, 31$, of this codeword are combined with the bytes in even positions, i.e. in position $0, 2, \ldots, 30$, of the preceding C_1-codeword, and the 8 bytes representing the Q- and P-check symbols are inverted, i.e. in the representation of these bytes as binary vectors the binary values 1 and 0 are exchanged. (This is indicated by overlining the corresponding P and Q symbols.) There are only technical reasons for this inversion. It assists data readout during areas with muted audio program.

In conclusion, we obtain a sequence of vectors

$$\begin{array}{cccc}
L_{6(n-3),A} & L_{6(n-2-d),B} & L_{6(n-3-2d)+2,A} & L_{6(n-2-3d)+2,B} \\
L_{6(n-3-4d)+4,A} & L_{6(n-2-5d)+4,B} & R_{6(n-3-6d),A} & R_{6(n-2-7d),B} \\
R_{\overline{6(n-3-8d)+2},A} & R_{\overline{6(n-2-9d)+2},B} & R_{\overline{6(n-3-10d)+4},A} & R_{\overline{6(n-2-11d)+4},B} \\
\overline{Q_{n-1-12d,0}} & \overline{Q_{n-13d,1}} & \overline{Q_{n-1-14d,2}} & \overline{Q_{n-15d,3}} \\
L_{6(n-1-16d)+1,A} & L_{6(n-17d)+1,B} & L_{6(n-1-18d)+3,A} & L_{6(n-19d)+3,B} \\
L_{6(n-1-20d)+5,A} & L_{6(n-21d)+5,B} & R_{6(n-1-22d)+1,A} & R_{6(n-23d)+1,B} \\
R_{\overline{6(n-1-24d)+3},A} & R_{\overline{6(n-25d)+3},B} & R_{\overline{6(n-1-26d)+5},A} & R_{\overline{6(n-27d)+5},B} \\
\overline{P_{n-1,0}} & \overline{P_{n,1}} & \overline{P_{n-1,2}} & \overline{P_{n,3}}.
\end{array}$$

This completes the description of the encoding process for error-detection and error-correction. For technical reasons, the resulting sequence of bytes is encoded once again before the data is written onto the disc. This final encoding, called EFM, ensures that the stored information satisfies certain standards used for binary data written on optical discs. For further details see 5.4.7 and 5.4.9.

Let v be the row vector of 5.4.2. In step B the four parity bytes $Q_{n,0}, \ldots, Q_{n,3}$ are determined by the equation $v \cdot \Delta_2^\top = 0$, where the check matrix Δ_2 of C_2 is given by

$$\Delta_2 := \begin{pmatrix} 1 & 1 & \ldots & 1 & 1 & 1 \\ \alpha^{27} & \alpha^{26} & \ldots & \alpha^2 & \alpha & 1 \\ \alpha^{54} & \alpha^{52} & \ldots & \alpha^4 & \alpha^2 & 1 \\ \alpha^{81} & \alpha^{78} & \ldots & \alpha^6 & \alpha^3 & 1 \end{pmatrix}$$

for α a root of the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1 \in \mathbb{F}_{2^8}[x]$. Let w be the row vector of 5.4.3. In step D the four parity bytes $P_{n,0}, \ldots, P_{n,3}$ are determined by the equation $w \cdot \Delta_1^\top = 0$, where the check matrix Δ_1 of C_1 is given by

$$\Delta_1 := \begin{pmatrix} 1 & 1 & \ldots & 1 & 1 & 1 \\ \alpha^{31} & \alpha^{30} & \ldots & \alpha^2 & \alpha & 1 \\ \alpha^{62} & \alpha^{60} & \ldots & \alpha^4 & \alpha^2 & 1 \\ \alpha^{93} & \alpha^{90} & \ldots & \alpha^6 & \alpha^3 & 1 \end{pmatrix}$$

for the same α as above.

CIRC decoding The standard for compact discs does not explicitly describe a CIRC decoding strategy. Different CD-players use different strategies, so the quality of error correcting performance varies from player to player. Here we describe one possible decoding strategy. **5.4.4**

The decoder obtains vectors containing 32 bytes, 24 of them contain audio information, the other 8 are check symbols added by C_1 and C_2. Odd numbered symbols are delayed by one vector and the parity symbols are inverted in order to reverse *step E* of the encoding process. The code C_1 has minimum distance $d_1 = 5$. It is used in order to correct a single error and to detect two or three errors in a codeword. If it detects exactly one error, then the wrong byte will be replaced by the corrected one. If it detects more than one error, then all 28 information symbols of the corresponding C_1-codeword are marked as erasures. Thus, C_1 is designed to correct short random errors and to detect longer burst errors.

How large is the probability that C_1 does not detect an error? (Cf. [104].) Assume that c was sent and $c + e$ was received. The error vector e is not detected if and only if $c + e$ belongs to a ball of radius 1 around a C_1-codeword c' different from c. This probability is

$$\frac{|C_1 \setminus \{c\}| \cdot |\{v \in \mathbb{F}_q^n \mid \operatorname{dist}(v,c) \leq 1\}|}{|\mathbb{F}_q^n|}$$

for $q = 2^8$ and $n = 32$. This is

$$\frac{(q^{28}-1)(1+32(q-1))}{q^{32}} \leq \frac{32q-31}{q^4} \approx 2^{-19}.$$

The interleaving of two consecutive C_1-codewords in *step E* allows one to break short burst errors.

Then, in order to reverse *step D* and *step C*, the last four symbols of each vector output by C_1, which are the check symbols, are deleted. Deinterleaving collects the 28 symbols representing C_2-codewords. The code C_2 is first of all used for correcting erasures. Since its minimum distance $d_2 = 5$, according to 5.2.2, it is possible to correct up to 4 erasures per codeword. If it is not possible to correct all erasures with C_2, then the erased symbols are passed to an interpolation process.

In addition, C_2 can be used to correct a single error and to detect symbols miscorrected by C_1. If miscorrected symbols are found, then all 24 audio symbols of the corresponding C_2-codeword are marked as erased and passed to the interpolation process.

Thus, C_2 is designed for the correction of burst errors and short random errors which were not corrected or miscorrected by C_1.

In worst case situations when the error is so massive that even interpolation fails, the audio signal is usually muted. In general, the brief silence is preferable to the burst of digital noise usually heard as a click.

What is the maximal size of a burst error which can still be corrected by CIRC? The interleaving with delay $d = 4$ in *step C* causes that each C_2-codeword is spread over 28 different C_1-codewords distributed among $27 \cdot 4 + 1 = 109$ consecutive C_1-codewords. This fact allows one to break long burst errors. Even if C_1 marked the symbols of 16 consecutive C_1-codewords as erasures, the original information can be reconstructed if no further errors have occurred in this data area. After deinterleaving, these $16 \cdot 28 = 448$ erasure marks are distributed over 124 different C_2-codewords, where at most 4 erasures occur in each of these codewords. Consequently, these erasures can be corrected. As mentioned above, there are exactly 24 audio symbols contained in a frame. These are the 24 audio bytes contained in a C_2-codeword. Since there are exactly 7350 frames per second which cover a track of 1200 mm, the CIRC decoding allows one to correct physical track errors of about 2.6 mm length. Thus, that kind of maximum length correctable error contains 384 audio bytes. The erased symbols are contained in 124 C_2-codewords, which are responsible for approximately 16.7 milliseconds of music.

The raw error bit rate of a CD is around 10^{-5} to 10^{-6}. This means that there is one wrong bit every 10^5 to 10^6 bits. Considering that an audio compact disc has an output of more than 4 million bits per second (cf. 5.4.9), the need for error correction is obvious. With error correction, perhaps 200 errors per second will be completely corrected. According to [164], the error rate after CIRC is between 10^{-10} and 10^{-11}. Nevertheless, the quality of error-correction varies from player to player, depending on the chosen CIRC decoding strategy.

5.4.5 **Interpolation** If it is impossible for the decoder to reconstruct a C_2-codeword, then the CD-player tries to interpolate the missing audio bytes from neighboring ones in case they are reliable. Because of the high correlation between music samples, an uncorrected error can be made virtually inaudible by synthesizing new data from surrounding data. Various interpolation schemes are used with different performance levels. In its simplest form the previous value is simply repeated. In first order interpolation the erased audio bytes are replaced by the mean value of the previous and the subsequent byte.

The interpolation process can be applied to determine the missing values even if the audio bytes of two consecutive C_2-codewords are marked as erasures, based on the scrambling in *step A*. Assume that the n-th and $(n+1)$-th C_2-codeword are erased, then the neighboring bytes of even samples occur in

the $(n-3)$-th, $(n-2)$-th, and $(n-1)$-th codeword and the neighboring bytes of odd samples occur in the $(n+2)$-th, $(n+3)$-th, and $(n+4)$-th codeword.

What is the maximal size of a burst error which can still be reconstructed with interpolation? Even if C_1 marked the symbols of 48 consecutive C_1-codewords as erasures, the audio information can still be reconstructed by interpolation. After deinterleaving the received information we obtain a sequence of vectors $(y^{(n)})_n$ with $y^{(n)} = (y_{28n}, y_{28n+1}, \ldots, y_{28n+27})$. These $48 \cdot 28 = 1344$ erasure marks are distributed over 124 different vectors $y^{(n)}$. Assume that $y^{(m)}$ is the first vector in this sequence whose last component belongs to these erased rows. Hence, in the interleaving array the column the top entry of which has the index $m + 108$ is the first erased column. Now we have to check that it is still possible to reconstruct all the audio information with error correction and interpolation. The vectors $y^{(m)}, \ldots y^{(m+3)}$ contain exactly one erasure which occurs in the last position. The vectors $y^{(m+4)}, \ldots y^{(m+7)}$ contain exactly two erasures which occur in the last two positions. The vectors $y^{(m+8)}, \ldots y^{(m+11)}$ contain exactly three erasures which occur in the last three positions. The vectors $y^{(m+12)}, \ldots y^{(m+15)}$ contain exactly four erasures which occur in the last four positions. Consequently, in all these vectors so far it is possible to fill the erased positions by correcting erasures with C_2.

For the following vectors we analyze how many erased bytes they contain, in which position they occur, and in which vectors and positions the neighboring audio bytes occur. Finally we will see that it is always possible to approximate the missing values by first order interpolation. The first column gives the index n of the vector $y^{(n)}$, the second shows the number of erased bytes in this vector, the third contains the position of the erased bytes, the next column contains the labels n' of the vectors $y^{(n')}$ which contain the audio bytes necessary for interpolation, and finally the last column contains the positions where these neighboring audio bytes occur. Careful investigation of this table proves that it is possible to reconstruct the missing information by interpolation.

n	erasures	pos.	n'	pos.
$m+16, \ldots, m+19$	5	$23-27$	$m+18, \ldots, m+22$	$6-11$
$m+20, \ldots, m+23$	6	$22-27$	$m+22, \ldots, m+26$	$6-11$
$m+24, \ldots, m+27$	7	$21-27$	$m+26, \ldots, m+30$	$0-11$
$m+28, \ldots, m+31$	8	$20-27$	$m+30, \ldots, m+34$	$0-11$
$m+32, \ldots, m+35$	9	$19-27$	$m+34, \ldots, m+38$	$0-11$
$m+36, \ldots, m+39$	10	$18-27$	$m+38, \ldots, m+42$	$0-11$
$m+40, \ldots, m+43$	11	$17-27$	$m+42, \ldots, m+46$	$0-11$
$m+44, \ldots, m+47$	12	$16-27$	$m+46, \ldots, m+50$	$0-11$
$m+48, \ldots, m+51$	12	$15-26$	$m+50, \ldots, m+54$	$0-11$
$m+52, \ldots, m+55$	12	$14-25$	$m+54, \ldots, m+58$	$0-11$

$m+56,\dots,m+59$	12	$13-24$	$m+58,\dots,m+62$	$0-11$
$m+60,\dots,m+63$	12	$12-23$	$m+62,\dots,m+66$	$0-9$
$m+64,\dots,m+67$	12	$11-22$	$m+66,\dots,m+70$	$0-9$
			$m+62,\dots,m+65$	$24-27$
$m+68,\dots,m+71$	12	$10-21$	$m+70,\dots,m+74$	$0-5$
			$m+66,\dots,m+69$	$24-27$
$m+72,\dots,m+75$	12	$9-20$	$m+74,\dots,m+78$	$0-5$
			$m+70,\dots,m+73$	$22-27$
$m+76,\dots,m+79$	12	$8-19$	$m+78,\dots,m+81$	$0-5$
			$m+74,\dots,m+77$	$22-27$
$m+80,\dots,m+83$	12	$7-18$	$m+82,\dots,m+85$	$0-5$
			$m+77,\dots,m+81$	$22-27$

The remaining cases can be studied in a similar way. For instance, the study of the four columns $m+80,\dots,m+83$ corresponds to the situation of $m+72,\dots,m+75$.

Consequently, all these erasures can be filled by interpolation. Even if 48 C_1-codewords are erased, the last byte of the first codeword preceding this erased block and the first byte of the codeword following this block are erroneous, it is possible to reconstruct approximations of the erased bytes by interpolation. These 48 erased C_1-codewords contain the audio information of 48 frames. Each second the CD-player reads 7350 frames which cover 1200 mm of the track, whence interpolation is able to deal with burst errors of length up to 7.8 mm.

Finally we discuss how data is stored on a CD (cf. [164, pages 51ff]).

5.4.6 Pits and lands "A transparent plastic substrate forms most of a discs 1.2 mm thickness. Data is physically contained in *pits* which are impressed along its top surface and are covered with a very thin metal layer. Another thin plastic layer protects the metallized pit surface on top of which the identifying label is printed. A laser beam is used to read the data. It is applied from below and passes through the transparent substrate, is reflected at the metallized pit surface, and passes back." The beam of size 800 micrometers at the discs surface is focused to 1.7 micrometers on the metallized pit surface. Pits are very small, approximately 0.5 micrometers wide and 0.11 to 0.13 micrometers high. As we will see, they are of varying length. Thus, "the laser beam is focused to a point about three times larger than the pit width."

"When viewed from the lasers perspective, the pits appear as bumps." The areas between pits are called *lands*. Data are read from the compact disc by measuring the reflected light. Almost 90% of the laser beam are reflected by a land. Caused by the height of the pits, the refraction index of the transparent

material, the wavelength of the laser beam, and the fact that the laser beam hits also land located around pits, almost no light is reflected from a pit. The remaining reflected light is used as a tracking signal. "The transition from pit to land or from land to pit, i.e. the change of the intensity of the reflected light, is considered as a binary one. When the laser beam hits areas within a land or within a pit it is interpreted as a sequence of zeros." For technical reasons pits and lands may not be too short or too long. To be more precise, a land or pit must have the length of at least 2 and at most 10 zeros. In other words, when reading binary data from a compact disc or writing it onto a compact disc, between two consecutive binary ones there must be a sequence of at least 2 and at most 10 zeros. Obviously, "binary data obtained from sampling and encoding does not satisfy these requirements. Thus, data provided by the CIRC encoder still must be changed before it can be written onto a compact disc." This is done by the channel encoder.

EFM The channel encoder uses *eight-to-fourteen modulation*, EFM, in order to change a byte, which is a binary vector of length 8, into a binary vector of length 14 which satisfies the requirements on the number of zeros between two consecutive binary ones (cf. [164, pages 77ff]). This process is usually done by table-lookup. Here is a small part of this table: **5.4.7**

00000000	01001000100000
00000001	10000100000000
00000010	10010000100000
00000011	10001000100000
00000100	01000100000000
00000101	00000100010000
00000110	00010000100000
00000111	00100100000000
⋮	⋮
11111000	01001000010010
11111001	10000000010010
11111010	10010000010010
11111011	10001000010010
11111100	01000000010010
11111101	00001000010010
11111110	00010000010010
11111111	00100000010010

It is easy to see (cf. Exercise 5.4.3) that 14 is the least length of vectors needed, in order to represent all 2^8 bytes by different vectors satisfying the requirements on the number of zeros between two consecutive binary ones. Actually there are 267 vectors with these properties. Two of them which are not used by EFM are used for subcode synchronization words (cf. 5.4.10).

5.4.8 **Merging bits** Now it can still happen that the concatenation of two vectors produced by EFM still does not satisfy the condition on the number of zeros between two consecutive binary ones. For this reason three *merging bits* are inserted between two vectors of length 14 (cf. [164, page 80]). Actually, there are only four possible choices for these bits, namely

$$000 \quad 100 \quad 010 \quad 001.$$

Two merging bits 00 are necessary to prevent consecutive binary ones. The third merging bit is added so that the average *digital sum* value is close to zero.

A bit pattern can also be interpreted as a rectangular wave. It admits values ± 1. A binary one causes a change of the sign. For instance, the vector 00000100010000 can be considered as

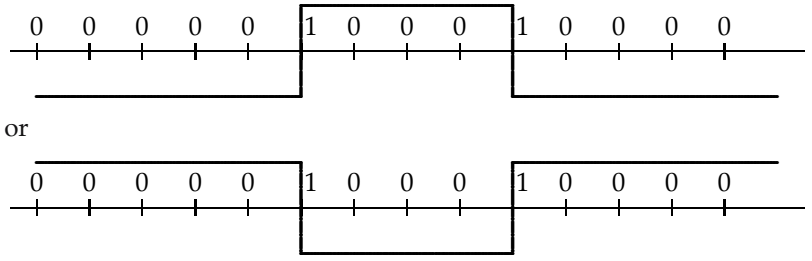
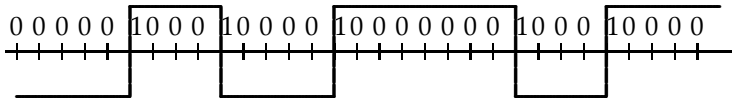


Fig. 5.8 Rectangular wave form of a bit pattern

The digital sum is determined by assigning $+1$ to the positive and -1 to the negative amplitude and summing these values for each bit. This is the integral of the rectangular wave form of the bit pattern when the distance between two bits is considered to be equal to 1.

Concatenating the above binary vector with itself, we may only use the merging bits containing a binary one, since otherwise there would be too many consecutive zeros. The three possibilities 100, 010, and 001 yield the following rectangular wave forms:



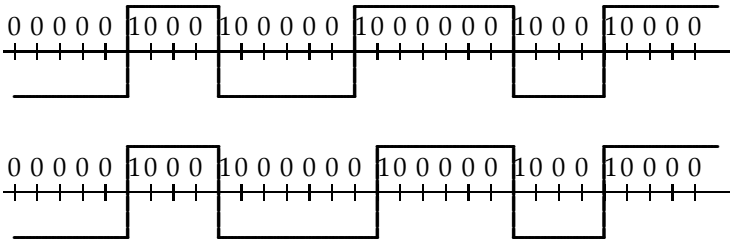


Fig. 5.9 Concatenation with merging bits

They have the digital sum values 3, 1, and -1. Depending on the digital sum value we started from, we insert those merging bits so that the final digital sum value is close to zero.

Frames As was already mentioned, the information on a compact disc is organized in form of frames (cf. [164, pages 82ff]). A frame consist of exactly 588 channel bits. Among other information it contains the audio information of one C_1-codeword. **5.4.9**

Each frame starts with a 24-bit synchronization pattern which is uniquely distinguishable from all other possible data patterns. It is given by

$$100000000001000000000010,$$

and it is used to maintain a constant data readout rate. The rate of synchronization patterns influences the rotational speed of the disc. After three merging bits a 14-bit subcode and another three merging bits are added. Then each byte of a C_1-codeword encoded by EFM to a 14 bit sequence and another 3 merging bits are added. This means that each byte of this codeword is enlarged to 14 bits by EFM and between two sequences produced by EFM three merging bits are inserted.

Thus a frame is of the following form, where "Sync." indicates the synchronization patterns, "M." suitable merging bits, "Subc." the subcode, and w_i the EFM-encoding of the i-th byte of the C_1-codeword. In the second row the number of used channel bits is indicated.

	Sync.	M.	Subc.	M.	w_0	M.	...	w_{31}	M.
bits	24	3	14	3	14	3	$30 \cdot (14+3)$	14	3

In summary, we have 24 synchronization bits, one 14-bit subcode, $32 \cdot 14$ bits produced by EFM from one C_1-codeword, and $34 \cdot 3$ merging bits in each frame.

Since there are 7350 frames per second, 4 321 800 channel bits are read or written per second. This gives 259 308 000 channel bits (32 413 500 channel bytes) per minute and 15 558 480 000 bits (1 944 810 000 channel bytes) per hour.

5.4.10 **Subcode** The 14-bit *subcode* is produced by EFM from an 8-bit subcode. These 8 bits are usually referred to as P, Q, R, S, T, U, V, and W. On an audio CD only the P and Q bits are used. They should not be mixed with the P and Q symbols of the CIRC. The subcode holds information about the different tracks on a compact disc, display information, digital copy permission, and control information for different functions of the CD-player. (For more details see [164, pages 90ff].)

Each frame contains 8 subcode bits. In order to make better use of these bits and store useful information with them, the subcode bits of 98 frames are collected to form a *subcode block*. Each frame contributes one P-bit, Q-bit and so forth to the subcode block. Thus the 8 subcode bits are used as 8 different channels.

Since there are 7350 frames per second and the subcode of 98 frames is collected to one block, there are 75 subcode blocks per second.

In general, each subcode channel contains synchronization words, instructions, commands, data, and even some parity check symbols. Each subcode block starts with two synchronization words. These are two patterns not used by the EFM. In other words, the 14-bit representation of the subcode of the first two frames in a subcode block can uniquely be detected. The two synchronization words are given by

$$00100000000001 \quad \text{and} \quad 00000000010010.$$

Hence the first two bits of each channel are reserved for synchronization. After the first two bits of the Q-channel reserved for synchronization, there follow 4 control bits, 4 address bits describing 3 different modes, then 72 bits of data, and finally 16 bits for cyclic redundancy checking. This CRC is done in the following way. Consider the vector of the $4 + 4 + 72 = 80$ (control-, address-, data-) bits as a polynomial f of degree at most 79 over $\mathbb{F}_2$. By the division algorithm determine $q, r \in \mathbb{F}_2[x]$ so that

$$f(x) = \big(x^{16} + x^{12} + x^5 + 1\big)q(x) + r(x),$$

with $\deg r < 16$, and consider the coefficients of r (or as it is actually done, the inverted binary values of them) as the 16 check symbols. When the receiver obtains a Q-channel, the first two bits are stripped, the next 80 bits are read, the division algorithm is applied and the remainder is compared with the final 16 parity check bits. If they coincide the receiver assumes that the data was correct.

The P-channel designates the starting and stopping of tracks.

In the next section we describe some differences between CD-DA and CD-ROM.

Exercises

Exercise Let C be a Reed–Solomon-code of length $n = p^r - 1$, dimension $k = p^r - d$ and minimum distance d. For $1 \leq s < k$ prove the following facts about the shortened Reed–Solomon-code $C(s)$ of length $n - s$, which is obtained from C by taking all codewords which have zeros in the last s positions and deleting the last s positions: **E.5.4.1**

1. If g is the generator polynomial of C, then
$$C(s) = \{fg \mid f \in \mathbb{F}_{p^r}[x],\ \deg f < k - s\}.$$
2. $C(s)$ is an $(n - s, k - s, d)$-code, thus it is an MDS-code.

Exercise Show that when the symbols of 17 consecutive C_1-codewords are erased, CIRC fails to correct this error. Show that when the symbols of 49 consecutive C_1-codewords are erased, the interpolation process fails to correct this error. **E.5.4.2**

Exercise Let $a(n)$ be the number of binary vectors of length n such that between two consecutive binary ones there are at least 2 and at most 10 zeros. Show that $n = 14$ is the least integer such that $a(n) \geq 2^8$. Prove that $a(14) = 267$. **E.5.4.3**

5.5 More Details on CD-ROM 5.5

CD-ROM and diverse other disc formats are thoroughly described in chapter 6 of [164]. Here we provide the reader with a short summary. "CD-ROM is the logical extension of the compact disc format towards the much broader application of information storage in general." The compact disc is used "as a read-only memory system" which can contain "any kind of program material." It is a "cost-effective way of distributing large amounts of information, especially information not requiring frequent updating," for instance, databases and mass storage for computer related applications.

Although the CD-ROM looks identically to an audio compact disc it uses a modified data format. "A CD-ROM identifies itself as differing from an audio compact disc through the Q-subcode channel."

The CD-ROM standard as specified in the *Yellow Book*, in the ISO/IEC standard *Information technology – Data interchange on read-only* 120 *mm optical data disks (CD-ROM)* [96], or also in [48], "does not link CD-ROM to a specific

application." Unlike the audio CD standard, "it does not define the type of information that is stored" on a CD-ROM. "Furthermore, the layout of the information on the disc is not defined, and it does not indicate where and how to store the directory, how to identify the beginning or end of a file, or how to open a file." On an audio compact disc sampled digital audio is stored. Since usually neighboring samples are quite similar, it is possible to apply interpolation in order to reconstruct audio information which could not be properly decoded by CIRC. For obvious reasons, when storing arbitrary information on a CD-ROM the method of interpolation cannot be applied. Hence, the error correction and error detection of a CD-ROM must provide a higher data integrity than on an audio compact disc.

The smallest data area of an audio compact disc is a frame containing 24 bytes of audio information. A frame is too short for numerical applications and there is no way of addressing frames. Similarly as with the subcode channels, 98 frames are combined to form a *sector* of a CD-ROM. (See also [164, pages 215ff].) A sector is the basic data unit of a CD-ROM. In general a frame contains $98 \cdot 24 = 2352$ bytes of information. Since in an audio CD there are exactly 7350 frames per second, we have $7350/98 = 75$ sectors per second on a CD-ROM.

5.5.1 Sector and sector modes

The first 12 bytes of each sector are used as a synchronization word. The next 4 bytes form a header field containing three address bytes and one mode byte. The *address bytes* indicate the minute (usually from 1 to 74), the second within this minute (from 0 to 59) and the sector within this second (from 1 to 75). For example, the three values $45 - 20 - 12$ indicate the 12-th sector in the 20-th second of the 45-th minute. This information is also found in the Q-subcode channel, but it speeds up and provides greater accuracy for searching. The *mode byte* indicates one of three different modes available for CD-ROM sectors. (See also [164, Fig. 6.2].)

Mode 0 just contains null data. Thus, after the synchronization word and the header it just contains 2336 zero-bytes.

	Header				
Sync.	Address			Mode 0	null data
	Min.	Sec.	Block		
12	1	1	1	1	2336

Fig. 5.10 CD-ROM sector mode 0

Mode 1 specifies, as described in Fig. 5.11, that 2048 of the remaining bytes are devoted to user data and the final 288 bytes are reserved for error detection EDC and error correction ECC. The error detection code is a CRC-code with

Sync.	Header			Mode 1	user data	Auxiliary data			
	Address					EDC	Space	ECC	
	Min.	Sec.	Block					*P*-parity	*Q*-parity
12	1	1	1	1	2048	4	8	172	104

Fig. 5.11 CD-ROM sector mode 1

respect to the polynomial

$$g(x) := \left(x^{16} + x^{15} + x^2 + 1\right)\left(x^{16} + x^2 + x + 1\right) \in \mathbb{F}_2[x].$$

The sequence of synchronization-, header-, and user data is considered as a binary polynomial. Dividing this polynomial by g, the division algorithm yields a remainder of degree less than 32, the coefficients of which are stored in the 4 EDC-bytes. Immediately after these bytes a sequence of 8 zero-bytes is appended as a space between the EDC and ECC bytes. The error correction encoding of a sector is carried out by a Reed–Solomon Product-like Code, RSPC. For more details see 5.5.2.

In **Mode 2** all remaining 2336 bytes are available for user data.

Sync.	Header			Mode 2	user data
	Address				
	Min.	Sec.	Block		
12	1	1	1	1	2336

Fig. 5.12 CD-ROM sector mode 1

The additional EDC and ECC of Mode 1 ensure a level of data integrity essential for storing arbitrary information. In Mode 1 each sector contains exactly 2 kB of information. Because of extended error correction Mode 1 has the most number of applications. The error rate is improved over that of an audio CD. Theoretically there will be one uncorrectable bit in every 10^{16} or 10^{17} bits. In Mode 2 it is possible to store more data on a CD-ROM, however with lower data integrity than in Mode 1. Therefore, it is mainly used for "gracefully degrading data such as video and audio."

RSPC encoding We assume that the bytes of a sector in Mode 1 are labeled as $B_0, \ldots, B_{2351}$. The header-bytes, user data, EDC-bytes, and space bytes are **5.5.2**

input to the RSPC encoder. These input bytes and the P-parity and Q-parity bytes make a total of 2340 bytes per sector. They are ordered in 1170 words W_n, $n \in 1170$. Each word consists of exactly 2 bytes, in more details

$$W_n = \big(B_{2n+12}, B_{2n+13}\big), \qquad n \in 1170.$$

Then we form two arrays of bytes, namely

$$(B_{2n+12})_{n\in 1170} \quad \text{and} \quad (B_{2n+13})_{n\in 1170}.$$

These two arrays are encoded separately. Let V_n, $n \in 1170$, denote the elements of such an array, i.e. either $V_n = B_{2n+12}$ or $V_n = B_{2n+13}$. The bytes V_n must be inserted into the diagram of Fig. 5.13 so that the first row contains from left to right the elements $V_0, V_1, \ldots, V_{42}$, the next one $V_{43}, V_{44}, \ldots, V_{85}$ and so on.

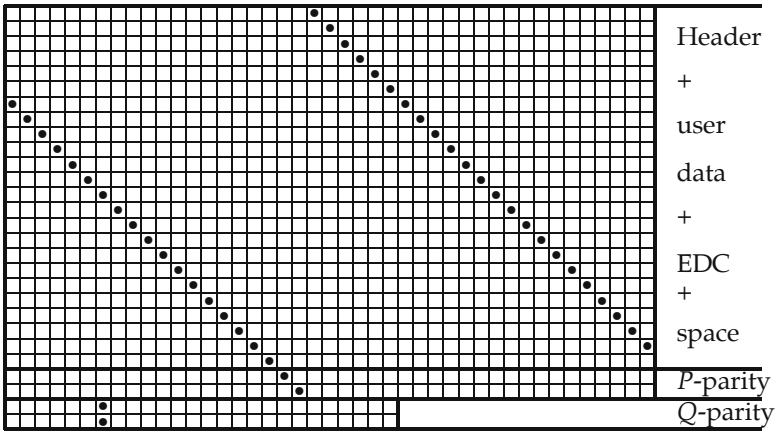


Fig. 5.13 RSPC encoding

The P-parity symbols are determined by a $(26, 24)$-Reed–Solomon-code over $\mathbb{F}_{2^8}$. It takes the first 24 entries of each column

$$(V_r, V_{r+43}, V_{r+2\cdot 43}, \ldots, V_{r+23\cdot 43}), \qquad r \in 44,$$

as an input and computes the symbols $V_{r+24\cdot 43}$ and $V_{r+25\cdot 43}$. A check matrix of this code is given by

$$\begin{pmatrix} 1 & 1 & \ldots & 1 & 1 & 1 \\ \alpha^{25} & \alpha^{24} & \ldots & \alpha^2 & \alpha & 1 \end{pmatrix}$$

for α a root of the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1 \in \mathbb{F}_{2^8}[x]$.

In order to determine the Q-parity symbols, a $(45, 43)$-Reed–Solomon-code over $\mathbb{F}_{2^8}$ is applied. Its codewords are diagonals as indicated by the black circles in the above diagram (cf. [68]). The i-th codeword, $i \in 26$, starts in the

leftmost column of the i-th row of this array. The 45 symbols occurring in the first 26 rows are the information symbols. The parity check symbols of this code are input into the last two rows of this array. The Q-parity check symbols of the i-th codeword occur in the i-th column of this part of the array. A check matrix of this code is given by

$$\begin{pmatrix} 1 & 1 & \dots & 1 & 1 & 1 \\ \alpha^{44} & \alpha^{43} & \dots & \alpha^2 & \alpha & 1 \end{pmatrix}$$

for the same α as above. Actually, these two codes are obtained by shortening a $(255, 253, 3, 2^8)$-Reed–Solomon code (cf. Exercise 5.4.1).

5.5.3

Encoding and decoding After having determined the bytes filled into a sector (null data in Mode 0, user data, EDC and ECC in Mode 1, and just user data in Mode 2), the bytes $B_{12}, \dots, B_{2351}$ of a sector are *scrambled*. For more details see [48] and [96]. "A regular bit pattern fed into the EFM encoder can cause large values of the digital sum value in case the merging bits cannot reduce this value. The scrambler reduces this risk by converting the input bit stream with a shift register in a prescribed way." The bytes of a *scrambled sector* are mapped onto a series of consecutive frames. Each frame consists of exactly 24 bytes. However, the starting point of a sector is not necessarily the starting point of the frame. The byte B_0 of a sector can be insert as the $4n$-th byte of a frame, $0 \leq n \leq 5$. Consecutive bytes of the sector are placed in consecutive bytes of a frame. After the byte B_{2351} the byte B_0 of the next sector is inserted. A frame is, therefore, of the form

$$\begin{array}{cccccc} B_{4r} & B_{4r+1} & B_{4r+2} & B_{4r+3} & B_{4r+4} & B_{4r+5} \\ B_{4r+6} & B_{4r+7} & B_{4r+8} & B_{4r+9} & B_{4r+10} & B_{4r+11} \\ B_{4r+12} & B_{4r+13} & B_{4r+14} & B_{4r+15} & B_{4r+16} & B_{4r+17} \\ B_{4r+18} & B_{4r+19} & B_{4r+20} & B_{4r+21} & B_{4r+22} & B_{4r+23}. \end{array}$$

Next the byte order of each even-odd numbered pair of bytes in the frame is reversed. We obtain

$$\begin{array}{cccccc} B_{4r+1} & B_{4r} & B_{4r+3} & B_{4r+2} & B_{4r+5} & B_{4r+4} \\ B_{4r+7} & B_{4r+6} & B_{4r+9} & B_{4r+8} & B_{4r+11} & B_{4r+10} \\ B_{4r+13} & B_{4r+12} & B_{4r+15} & B_{4r+14} & B_{4r+17} & B_{4r+16} \\ B_{4r+19} & B_{4r+18} & B_{4r+21} & B_{4r+20} & B_{4r+23} & B_{4r+22} \end{array}$$

a so called *F1-frame*. Each F1-frame is then encoded by a conventional CIRC encoder (cf. 5.4.1). This yields an *F2-frame*, containing exactly 32 bytes. Adding one additional subcode byte to each F2-frame yields an *F3-frame*. Similarly as in an audio compact disc, there are eight different subcode bits referred to as P, Q, R, S, T, U, V, and W (cf. 5.4.10). The information stored in the subcode

of 98 consecutive frames is collected to 8 subcode channels. A group of 98 F3-frames is also known as a *section*. Because of the delays during the CIRC encoding, sections have nothing to do with sectors. Finally, after sending the 33 bytes of an F3-frame to EFM (cf. 5.4.7) the data are written onto the CD.

Usually the error rate of a compact disc after CIRC decoding is approximately 10^{-10} to 10^{-11} errors per bit. The CIRC decoder also delivers information about bytes that could not be correctly decoded. Since we know the exact position of these bytes, they are erasures. The two Reed–Solomon-codes used in RSPC have minimum distance $d = 3$. Hence, each of them can be used to correct 1 error or 2 erasures per codeword. Combining both CIRC and RSPC decreases the bit error rate of a CD-ROM so that it will be between 10^{-16} and 10^{-17} (cf. [164]).

There exist various extension of the CD-ROM standard, for instance Compact Disc-Interactive, CD-I, described in the *Green Book*, Video compact discs defined in the *White Book*, or recordable discs, the standard of which can be found in the *Orange Book*. For more details see [164].