Channel Coding

Prof. Dr.-Ing. Werner Henkel
Jacobs University Bremen
E-mail: w.henkel@jacobs-university.de
http://trsys.faculty.jacobs-university.de

2021

Contents

Literature

Lecture notes

Dorsch, B.: *Codierungstheorie*, TH Darmstadt.

Calderbank, A.R.: *Bandwidth Efficient Communication*, Princeton University, 1993.

Books on channel coding

Johnson, S. J., *Iterative Error Correction, Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*, Cambridge University Press, 2010, ISBN 978-0-521-87148-8.

Declercq, D., Fossorier, M., and Biglieri, E. (ed.), *Channel Coding, Theory, Algorithms, and Applications*, Elsevier Academic Press, 2014, ISBN 978-0-12-396499-1.

Lin, S., Ryan, W., *Channel Codes: Classical and Modern*, Cambridge University Press, 2009, ISBN 978-0521848688.

Lin, S., Costello, D.J., *Error Control Coding*, Pearson, 2004, ISBN 978-0130426727.

Richardson, T., Urbanke, R., *Modern Coding Theory*, Cambridge University Press, 2008, ISBN 978-0521852296.

Blahut, R.E., *Theory and Practice of Error Control Codes*, Addison Wesley, Reading, Massachusetts, 1983, ISBN 0-201-10102-5.

MacWilliams, F.J., Sloane, N.J.A., *The Theory of Error-Correcting Codes*, North Holland, Amsterdam, 5. Aufl. 1986, ISBN 0-444-85193-3.

Dholakia, A., *Introduction to Convolutional Codes with Applications*, Kluwer, Boston, 1994, ISBN 0-7923-9467-4.

Clark, G.C., Cain, J.B., *Error-Correction Coding for Digital Communications*, Plenum Press, New York and London, 3. Aufl. 1988, ISBN 0-306-40615-2.

Peterson, W.W., Weldon, E.J., *Error-Correcting Codes*, MIT Press, Cambridge, Massachusetts, London, England, 6. Aufl. 1981, ISBN 0-262-16039-0.

Heise, W., Quattrocchi, P., *Informations- und Codierungstheorie*, Springer, Berlin, 2. Aufl. 1989, ISBN 0-387-50537-7 und 3-540-50537-7.

Friedrichs, B., *Kanalcodierung. Grundlagen und Anwendungen in modernen Kommunikationssystemen*, Springer, Berlin, 1995, ISBN 3-540-59353-5.

Bossert, M., *Kanalcodierung*, Teubner, Stuttgart, 1992.

Michelson, A.M., Levesque, A.H., *Error-Control Techniques for Digital Communication*, Wiley, New York, 1985, ISBN 0-471-88074-4.

Blahut, R.E., *Algebraic Methods for Signal Processing and Communications Coding*, Springer, New York, Berlin, 1991, ISBN 0-387-97673-6 und 3-540-97673-6.

Mc Eliece, R.J., *Finite Fields for Computer Scientists and Engineers*, Kluwer, Boston, 1987, ISBN 0-89838-191-6.

Viterbi, A.J., Omura, J.K., *Principles of Digital Communication and Coding*, McGraw-Hill, Tokyo, 1979, ISBN 0-07-067516-3.

v. Ammon, U., Tröndle, K., *Mathematische Grundlagen der Codierung*, R. Oldenburg, München, Wien, 1974, ISBN 3-486-34681-4.

van Lint, J.H., *Introduction to Coding Theory*, Springer, New York, Heidelberg, Berlin, 1982, ISBN 0-387-11284-7 or 3-540-11284-7.

Berlekamp, E.R., *Algebraic Coding Theory*, McGraw-Hill, St. Louis, San Francisco, Toronto, London, Sydney, 1968.

Mc Eliece, R.J., *The Theory of Information and Coding*, Addison-Wesley, 1977.

Vanstone, S.A., van Oorschot, P.C., *An Introduction to Error Correcting Codes with Applications*,**Heegard, C., Wicker, S.B.,** *Turbo Coding*, Kluwer, Boston, Dordrecht, London, 1999, ISBN 0-7923-8378-8. Kluwer, Boston, 1989, ISBN 0-7923-9017-2.

Anderson, J.B., Mohan, S., *Source and Channel Coding*, Kluwer, Norwell, Massachusetts, 1991, ISBN 0-7923-9210-8.

Duske, J., Jürgensen, H., *Codierungstheorie*, BI, Mannheim, Wien, Zürich, 1977, ISBN 3-411-01527-6.

Pretzel, O., *Error-Correcting Codes and Finite Fields*, Oxford University Press, Oxford, 1992, ISBN 0-19-859678-2.

Berlekamp, E. (Editor), *Key Papers in the Development of Coding Theory*, IEEE Press, Piscataway, N.J., 1974, ISBN 0-87942-032-4.

Kaderali, F., *Digitale Kommunikationstechnik I*, Vieweg, Braunschweig, 1991, ISBN 3-528-04710-0.

Heegard, C., Wicker, S.B., *Turbo Coding*, Kluwer, Boston, Dordrecht, London, 1999, ISBN 0-7923-8378-8.

Vucetic, B., Yuan, J., *Turbo Codes: Principles and Applications*, Kluwer, 2000, ISBN 0-7923-7868-7.

Books on information theory

Cover, T.M., Thomas, J.A., *Elements of Information Theory*, Wiley, New York, 2006, ISBN 978-0471241959, 978-8126541942.

Csiszár, I., Körner, J., *Information Theory, Coding Theorems for Discrete Memoryless Systems*, Akadémiai Kiadó, Budapest 2011, ISBN 978-0521196819.

MacKay, D.J.C., *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press 2003, http://www.inference.org.uk/mackay/itila/book.html

Gallager, R.G., *Information Theory and Reliable Communication*, John Wiley & Sons, New

York, 1968, ISBN W-471-29048-3.

Blahut, R.E., *Principles and Practice of Information Theory*, Addison-Wesley, Reading, Massachusetts, 1987, ISBN 0-201-10709-0.

Slepian, D. (Editor), *Key Papers in the Development of Information Theory*, IEEE Press, Piscataway, N.J., 1973, ISBN 0-87942-027-8.

Gray, R.M., *Entropy and Information Theory*, Springer, New York, Heidelberg, Berlin, 1990, ISBN 0-387-97371-0 or 3-540-97371-0.

Ash, R., *Information Theory*, Dover Books on Mathematics, 1991, ISBN 978-0486665214.

Books on coded modulation

Conway, J.H., Sloane, N.J.A., *Sphere Packings, Lattices and Groups*, Springer, New York, Berlin, 1988, ISBN 0-387-96617-X und 3-540-96617-X.

Jamali, S.H., Le-Ngoc, T., *Coded-Modulation Techniques for Fading Channels*, Kluwer, Boston, 1994, ISBN 0-7923-9421-6.

Biglieri, E., Divsalar, D., McLane, P.J., Simon, M.K., *Introduction to Trellis-Coded Modulation with Applications*, MacMillan, New York, 1991, ISBN 0-02-946542-7.

Biglieri, E., Luise, M. (Editors), *Coded Modulation and Bandwidth-Efficient Transmission*, Elsevier, Amsterdam, 1991, ISBN 0-444-89202-8.

Huber, J., *Trelliscodierung*, Springer, Berlin, 1992, ISBN 3-540-55792-X.

Books on transmission, signal processing, and source coding (including some channel coding)

Lee, E.A., Messerschmitt, D.G., *Digital Communication*, Kluwer, Boston, 1988, ISBN 0-89838-274-2.

Proakis, J.G. and Salehi, M., *Digital Communications*, McGraw-Hill, 5th ed., 2007, ISBN 978-0072957167.

Gitlin, R.D., Hayes, J.F., Weinstein, S.B., *Data Communications Principles*, Plenum Press, New York, London, 1992, ISBN 0-306-43777-5.

Sklar, B. and Harris, F.J., *Digital Communications: Fundamentals and Applications*, Prentice Hall, Upper Saddle River, NJ, 2020, ISBN 0134588568.

Ziemer, R.E., Peterson, R.L., *Introduction to Digital Communication*, MacMillan, New York, 1992, ISBN 0-02-946431-5.

Blahut, R.E., *Digital Transmission of Information*, Addison-Wesley, Reading, Massachusetts, 1990, ISBN 0-201-06880-X.

Blahut, R.E., *Fast Algorithms for Digital Signal Processing*, Addison-Wesley, Reading, Massachusetts, 1984, ISBN 0-201-10155-6.

Benedetto, S., Biglieri, E., Castellani, V., *Digital Transmission Theory*, Prentice Hall, Englewood Cliffs, N.J., ISBN 0-13-215062-X.

Carlson, A.B., *Communication Systems*, McGraw-Hill, New York, 2. Aufl., 1987, ISBN 0-07-009960-X.

Spilker, J.J., *Digital Communications by Satellite*, Prentice Hall, Englewood Cliffs, N.J., 1977, ISBN 0-13-214155-8.

Salomon, D., *Data Compression, The Complete Reference*, 4th ed., Springer, 2007, ISBN 978-1-84628-603-2.

Gray, R.M., *Source Coding Theory*, Kluwer, Norwell, Massachusetts, 1990, ISBN 0-7923-9048-2.

(The list is by no means complete and the order should not imply any quality ranking.)

Chapter 1

Channels and channel models, metrics, linear block codes primer

This first chapter provides an introduction into the topic, starting from a distinction from neighboring fields. Some typical channels will be described, metrics will be defined as distance measures, and based on simple linear block codes, the basic block-coding definitions will be compiled. Following this chapter, however, block codes will first be put aside and convolutional codes will be treated, instead. A comprehensive treatment will then follow after an introduction into the necessary algebraic framework in Chapter 7. Convolutional codes can be explained without the tools of discrete algebra and thus, only for didactic reasons, these codes are treated first.

1.1 Distinction between Channel Coding and related topics

Very often, the term 'coding' is used with a different meaning. A clear definition and distinction to other topics is therefore put at the beginning of this channel coding text. We distinguish the following four areas that have either similar names or some other relation:

1. Line coding
2. Source coding
3. Channel coding
4. Cryptology

Line coding has been in practical use for a very long time, almost from the beginning of digital communications over AC-coupled copper pairs. Line coding provides the required spectral shaping, especially in the form of a spectral null at DC. A manifold of block- and sequence-oriented procedures have been developed. This will not be the focus of this work. Nevertheless, the corresponding metric will be defined in Section 1.4, and in Chapter 6.3, line coding will serve as

a possible application of Trellis-Shaping. A comprehensive treatment of line codes can, *e.g.*, be found in [1].

Source coding is a means of reduction of redundancy or irrelevance in a signal or in data. On one hand, such measures comprise methods that do not modify the contained information, such as the well-known Huffman algorithm [2] and the variants of the Lempel-Ziv-Algorithms [3–5], which are widely used under names like 'compress', 'zip', or 'gzip' and are part of almost every computer software installation.[1] On the other hand, there are procedures that take the subjective properties of the recipient into account. Irrelevant signal components are eliminated, which is not regarded as disturbance or at least is considered to be tolerable. Audio (speech, music) and video coding algorithms are widely used in modern transmission systems. [7] provides a collection of audio and video-coding algorithms. More recent books are [8–10] for audio and [11, 12] for video coding.
For a information-theoretic treatment of source coding, the reader is referred to [13].

Channel coding provides error protection by means of a suitable addition of redundancy to the signal or data. At first sight, this is conterproductive to Source Coding. Source coding eliminates less structured redundancy, whereas channel coding adds redundancy in such a structured way that it allows for error detection and correction. The different algorithms for channel coding will be treated in this book. We divide into block- and sequence-based methods,*i.e.* , block and convolutional codes.

All different kinds of coding need not be seen as separate and independent. Actually, there are tight relations between them. Often, it proves to be more efficient, to combine source and channel coding or to provide spectral shaping, *i.e.*, line coding, by means of special channel coding.

Cryptology appears to be somewhat aside from signal and date transmission. Nevertheless, there exist strong links between cryptology and channel coding. Both disciplines make use of the same mathematical bases, the discrete algebra, and, they even use the same algorithms. However, the aim of cryptology is not related to the reduction or introduction of redundancy. Cryptology is devoted to the secure transmission of information, avoiding unauthorized access of third parties, and the authentification. With crypto algorithms in place, there exist additional requirements for source and channel coding to ensure that signal redundancy does not allow for easier decryption. An introduction is given by [14–16].

In this book, cryptographic methods will not be treated and will be considered as part of the source and destination.

1.2 Components of a transmission system

Figure 1.1 shows the succession of source coding, channel coding, and modulation in a transmit path. This should not lead to the impression of separate and independent components. A common design of source and channel coding and of channel coding and modulation (coded modulation) is indeed usually the better approach. These aspects will also be discussed in this book. Especially, the chapters on convolutional codes and the chapters on block codes will be concluded with the corresponding coded modulation schemes.

[1] As more recent work, a paper by Willems et al. [6] may be mentioned.

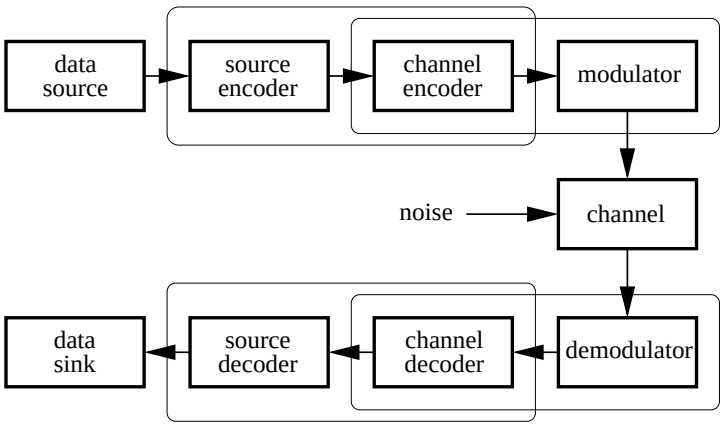


Figure 1.1: The transmission path

1.3 Channels and their modelling

Before we proceed to channel coding itself, this section provides some basic knowledge of typical exemplary channels.

1.3.1 AWGN (Additive White Gaussian Noise)

This is the most commonly used channel idealization. It is assumed that additive noise samples are the output of a random process that has a Gaussian density and whose samples are statistically independent.

The Gaussian density is known to be

$$p(r|v) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(r-v)^2}{2\sigma_n^2}} \quad . \tag{1.1}$$

σ_n can be written as $\sigma_n = 1/\sqrt{2E_s/N_0}$ when normalizing the signal energy. For 2-PSK, this would mean ± 1 instead of $\pm\sqrt{E_s}$. N_0 denotes the one-sided power spectral density. The derivation of this density and the corresponding standard deviation at the output of a 'matched' filter can be found in almost every standard book on transmission systems and will therefore not be handled here. Instead we refer to, *e.g.*, Gitlin et al. [17], Blahut [18], and Lee, Messerschmitt [19].

1.3.2 BEC (binary erasure channel)

The binary erasure channel models errors at known positions. It is suitable, e.g., to describe packet loss in data networks. It is also used a lot in proofs for LDPC codes, since they become more tractable and often, results appear transferable to other channels. Figure 1.2 shows the BEC, where Δ denotes the erasure.

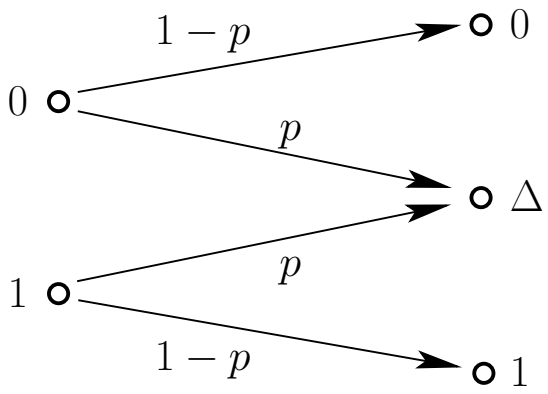


Figure 1.2: Binary erasure channel

1.3.3 BSC (binary symmetrical channel)

The binary symmetric channel as given in Fig. 1.3 is the typical channel model of binary transmission without any analog information, *i.e.*, at the output of a binary quantizer.

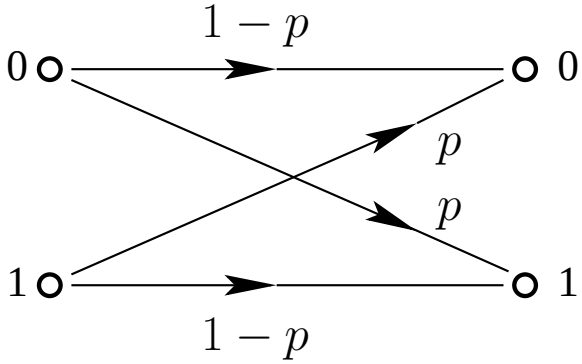


Figure 1.3: Binary symmetric channel

In case of an underlying Gaussian channel (AWGN), a source with a binary signal alphabet $\{+\sqrt{E_s}, -\sqrt{E_s}\}$ (normalized $\{+1, -1\}$), and a hard quantization with a threshold in the middle, we obtain using the densities

$$P(r|\pm 1) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(r\mp 1)^2}{2\sigma_n^2}}$$

the error probability p of the corresponding binary symmetric channel as

$$\begin{aligned} p &= \int_0^\infty p(y|-1)dy = \int_0^\infty \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(y+1)^2}{2\sigma_n^2}} dy = \frac{1}{\sqrt{\pi}} \int_{\frac{1}{\sqrt{2\sigma_n^2}}}^\infty e^{-t^2} dt \\ &= \frac{1}{2}\mathrm{erfc}\left(\frac{1}{\sqrt{2}\sigma_n}\right) = Q\left(\frac{1}{\sigma_n}\right) \\ &= \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}}\right) = Q\left(\sqrt{\frac{2E_s}{N_0}}\right) \end{aligned} \tag{1.2}$$

with

$$\begin{aligned} \mathrm{erf}(x) &= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \ , & (1.3) \\ \mathrm{erfc}(x) &= 1 - \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \ , & (1.4) \\ Q(x) &= \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt = \frac{1}{2} \cdot \mathrm{erfc}(x/\sqrt{2}) \ . & (1.5) \end{aligned}$$

When transmitting a word of length N over the binary symmetric channel, the probability of e erroneous bits at arbitrary positions is

$$p(e) = \binom{N}{e} p^e (1-p)^{N-e} \ . \tag{1.6}$$

In case a code would be able to correct t errors, the probability of error-free correction would be

$$p_{corr} = \sum_{e \le t} \binom{N}{e} p^e (1-p)^{N-e} \ . \tag{1.7}$$

Correspondingly,

$$p_{nocorr} = 1 - p_{corr} = \sum_{e > t} \binom{N}{e} p^e (1-p)^{N-e} \ . \tag{1.8}$$

is the probability of not being able to correct, *i.e.*, the block-error probability .

1.3.4 Gilbert-Elliott model of a channel with memory

The Gilbert-Elliott model shown in Fig. 1.4 is a Markovian model of first order and is specified by transition probabilities between two states that are usually denoted as 'good' or 'bad'. It is a suitable simplifying model for a mobile communications channel with shadowing and extreme changes between very good channel conditions with a low bit error probability p_{bG} and a very bad channel state with a bit error probability of p_{bB} near 0.5 hat. The states thus represent different bit error probabilities.

Markovian models are described by means of their transition matrix:

$$\begin{pmatrix} p_G(t_0+T) \\ p_B(t_0+T) \end{pmatrix} = \begin{pmatrix} 1-q_1 & q_2 \\ q_1 & 1-q_2 \end{pmatrix} \begin{pmatrix} p_G(t_0) \\ p_B(t_0) \end{pmatrix} \ , \tag{1.9}$$

with the probabilities $p_G(t)$ and $p_B(t)$ that the channel is in state G and B, respectively, at time t.

Such $n \times n$-transition matrices fulfil $\sum_{i=1}^{n} q_{ij} = 1$, where q_{ij} are the transition probabilities.

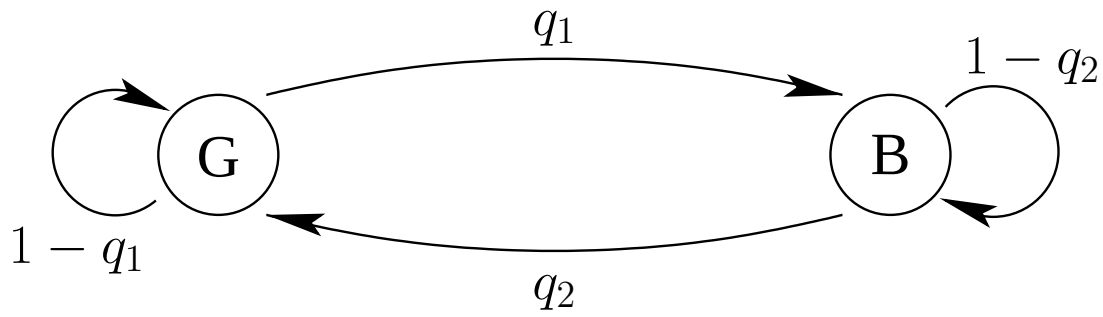


Figure 1.4: Gilbert-Elliott model

The average burst length, *i.e.*, the average time to remain in state B is

$$\sum_{i=0}^{\infty} 1 \cdot (1-q_2)^i = \frac{1}{1-(1-q_2)} = 1/q_2 \; . \tag{1.10}$$

The following averaging provides an alternative derivation.

$$\sum_{i=0}^{\infty} (i+1) \cdot q_2 \cdot (1-q_2)^i = 1/q_2 \tag{1.11}$$

Correspondingly, the average gap length, *i.e.*, the average burst separation is $1/q_1$.

The average bit error probability is given by

$$p = \frac{\frac{1}{q_2} p_{bB} + \frac{1}{q_1} p_{bG}}{\frac{1}{q_2} + \frac{1}{q_1}} = \frac{q_1 p_{bB} + q_2 p_{bG}}{q_1 + q_2} \tag{1.12}$$

1.3.5 Fading channels

After having described a simplifying model of a channel with memory that may be suited as a simple model of a fading channel, we will now shortly derive the distribution of so-called small-scale fading. These are the statistics that result from a manifold of scattered components and leads to faster variations. Shadowing and changes that depend on larger objects or the landscape are denoted as large-scale fading. It follows a log-normal density and will not be described in here. A more detailed treatment of the mobile channel can be found in [20–22].

In case of many scatter paths that are combined at the receiver, we obtain the received signal

$$\begin{aligned} r(t) &= \sum_i \alpha_i(t) \cos(\omega_c t - \theta_i(t)) \\ &= a_I(t) \cos(\omega_c t) + a_Q(t) \sin(\omega_c t) \; . \end{aligned} \tag{1.13}$$

with

$$
\begin{aligned}
a_I(t) &= \sum_i \alpha_i(t)\cos(\theta_i(t)) \\
a_Q(t) &= \sum_i \alpha_i(t)\sin(\theta_i(t)) \qquad \theta_i(t) = \omega_c \cdot \tau_i(t)
\end{aligned}
$$

Assuming $\alpha_i(t)$ to be iid (independent and identically distributed) and $\theta_i(t)$ to be equally distributed within the interval $[0, 2\pi]$, it follows from the central limit theorem that $a_I(t)$ and $a_Q(t)$ will be statistically independent zero mean Gaussian random variables with variance σ_a^2.

The probability density of the fading amplitude (envelope)

$$
a = \sqrt{a_I^2 + a_Q^2} \tag{1.14}
$$

results in

$$
p_F(a) = \frac{a}{\sigma_a^2} e^{-\frac{a^2}{2\sigma_a^2}} \; . \tag{1.15}
$$

In the following, a short proof is provided.

Proof: The proof is based on the transformation rule

$$
p(y = f(x)) = \sum_{x:x=f^{-1}(y)} p(x) \; / \; \left| \frac{df(x)}{dx} \right| \tag{1.16}
$$

and the fact that the sum of two random variables is described by the convolution of their densities.

Squaring of the I and Q components $y = f(x) = x^2 \longrightarrow df/dx = 2x = 2\sqrt{y}$ yields

$$
\frac{1}{\sqrt{2\pi\sigma_a^2}} \cdot 2 \cdot \frac{1}{2\sqrt{y}} \cdot e^{-\frac{y}{2\sigma_a^2}} \; , \quad y \geq 0 \tag{1.17}
$$

The convolution of the densities results in

$$
\int_0^y \frac{1}{2\pi\sigma_a^2} \frac{1}{\sqrt{t}} e^{-\frac{t}{2\sigma_a^2}} \cdot \frac{1}{\sqrt{y-t}} e^{-\frac{y-t}{2\sigma_a^2}} dt = \frac{1}{2\pi\sigma_a^2} e^{-\frac{y}{2\sigma_a^2}} \int_0^y \frac{1}{\sqrt{t(y-t)}} dt =
$$

$$
= \frac{1}{2\pi\sigma_a^2} e^{-\frac{y}{2\sigma_a^2}} \underbrace{[-\arcsin(1 - 2t/y)]_{t=0}^{t=y}}_{=\pi} =
$$

$$
= \frac{1}{2\sigma_a} e^{-\frac{y}{2\sigma_a^2}} \; , \quad y \geq 0 \; . \tag{1.18}
$$

A further application of the transformation rule with $a = f(y) = \sqrt{y} \longrightarrow df/dy = 1/(2a)$ yields the **Rayleigh** density.

$$p_F(a) = \frac{a}{\sigma_a^2} e^{-\frac{a^2}{2\sigma_a^2}} \ . \tag{1.19}$$

The phase of the fading $\Phi = \arctan \frac{a_Q}{a_I}$ is equally distributed within the interval $[0, 2\pi]$.

□

If there should be a strong direct path (line of sight, typical for maritime mobile communication), the so-called **Rice Fading** is obtained. The fading amplitude follows to be

$$a = \sqrt{(A + a_I)^2 + a_Q^2} \ . \tag{1.20}$$

It can be shown that

$$p_F(a) = 2a(1 + K)e^{(K+a^2(1+K))} I_0(2a\sqrt{K(K+1)}) \quad \text{with} \quad I_0(x) = \sum_{k=0}^{\infty} \left(\frac{x^k}{2^k k!} \right)^2 \ . \tag{1.21}$$

I_0 is the modified Bessel function of first kind and order zero and the Rice factor $K = \frac{A}{2\sigma_a^2}$ describes the share of the direct path. The phase will not be equally distributed any more.

1.3.6 Some other channels

To mention just a few other channel conditions, there are many non-white noise environments like the one due to NEXT and FEXT(Near-End Crosstalk and Far-End Crosstalk) between wire pairs in twisted-pair cables. Due to common clocks, such disturbances may even have cyclostationary properties.

Impulse noise caused by relays, electrical engines, lightning, etc., are a counterpart to Fading. In Fading the signal amplitude suddenly drops, whereas with impulse noise, the noise amplitude suddenly becomes very high. Both disturbances may lead to error bursts.

Jamming denotes an intentional impairment by short-time or narrow-band disturbers.

Furthermore, there are disturbances due to quantization and nonlinearities.

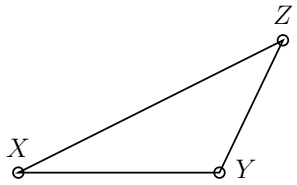
1.4 Metrics

In order to take different channel properties into account, corresponding distance measures have been defined that are called *metrics*, which is not meant in a mathematical strict sense. Later, we will discuss the choice of a suitable metric for special applications. In this section, we will just present the most important ones and intuitively, their possible application should become evident.

The mathematical metric definition would be

Definition 1.1 *Elements $x, y, z, \ldots$ from a set M, denoted as a metric space, a non-negative number $D[x, y]$ is associated with that must have the following properties*

1. $D[x, x] = 0$,
2. $D[x, y] = D[y, x] \neq 0$, *if* $x \neq y$ *(commutativity),*
3. $D[x, z] \leq D[x, y] + D[y, z]$ *(triangular inequality).*



$D[x, y]$ *is denoted as* distance.

For our purposes, the fulfillment of the triangular inequality is usually not important. We thus still call a distance measure a metric, even if the triangular inequality is not fulfilled.

1.4.1 Euclidean distance

The Euclidean distance fulfills all the axioms of a metric and is thus a real metric in strict mathematical sense. The squared Euclidean distance is contained in the Gaussian density. The probability of a certain received value is then dependent on the (squared) Euclidean distance from the transmitted symbol.

Definition 1.2 *The* **Euclidean distance** d_E *between two vectors* $\mathbf{a}$ *and* $\mathbf{b}$ *of length n with components $a_i, b_i \in \mathbb{R}$ is given by*

$$d_E^2 = \sum_{i=0}^{i=n} (a_i - b_i)^2 \tag{1.22}$$

Complex components a_i and b_i can be regarded as twice as long vectors with real components.

1.4.2 Hamming distance

In case of bursty noise, where only the number of disturbed symbols is of interest, or if a (binary) hard quantization is carried out at the receiver, the Hamming distance will be the suitable distance measure.

Definition 1.3 *The* **Hamming distance** d_H *between two vectors* $\mathbf{a}$ *and* $\mathbf{b}$ *of length n with components a_i and b_i that may be elements of an arbitrary number field, are given as the number of different components.*

(A somewhat more formal description would be the cardinality $d_H = |M|$ of the set $M = \{j | a_j \neq b_j\}$.)

Definition 1.4 *The* **Hamming weight** w_H *of a vector is the number of components different from zero.*

(More formal: $w_H = |M|$, with $M = \{j | a_j \neq 0\}$)

1.4.3 Lee distance

To understand the meaning of the Lee distance, one should think of signal points equally spaced on the unit circle, similar to a PSK. For this signal set, it can be useful to have a measure for the shortest distance between two points given in segments of the circle.

Definition 1.5 *Let a set M with cardinality $|M| = q$ be completely ordered with elements $a_0 < a_1 < a_2 < ... < a_{q-1}$. The Lee distance is then given as*

$$d_L^1(a_i, a_j) = \min\{|j - i|, |q + i - j|, |q + j - i|\} , \quad a_i, a_j \in M . \tag{1.23}$$

For vectors $\mathbf{w}$ and $\mathbf{v}$ of length n with components $w_k, v_k \in M$ the Lee metric is then given by

$$d_L^n(\mathbf{w}, \mathbf{v}) = \sum_{k=1}^{n} d_L^1(w_k, v_k) . \tag{1.24}$$

For illustration, one may determine the Lee distance of two points in Fig. 1.5.

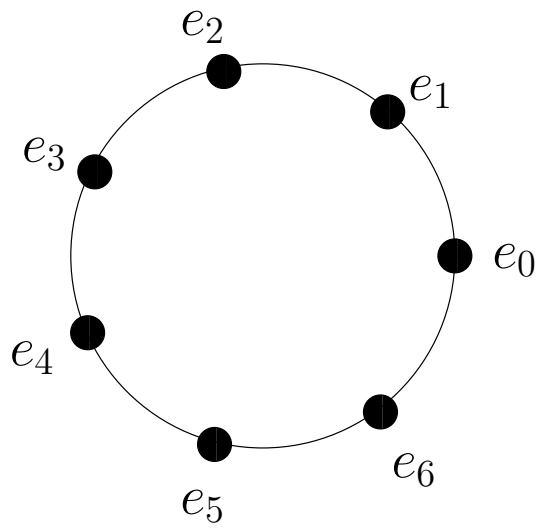


Figure 1.5: Illustration of the Lee distance

1.4.4 Product distance

The so-called product distance shows up in the computation of the pairwise error probability between two valid code sequences (for Fading channels).

Definition 1.6 *Let L be the length of an error event (burst) and let $\boldsymbol{a}$, b be two valid code sequences. The* **product distance** *is then defined as*

$$d_{\Pi}^2 = \prod_{\substack{k=1 \\ d_k \neq 0}}^{L} d_k^2(a_k, b_k) \tag{1.25}$$

The product distance thus is the L'th power of the geometric mean of the quadratic Euclidean distances of L' different sequence components.

1.4.5 Running Digital Sum (RDS)

This metric is applied in line coding and is thus a little aside from the other metrics listed before. Since line coding will be handled together with channel coding, it will nevertheless been mentioned here.

The discrete Fourier transform (DFT) is given by

$$F_k = \sum_{i=0}^{n-1} f_i e^{-j\frac{2\pi}{n}ik} \tag{1.26}$$

The power at DC, *i.e.*, at $f = 0$ is then given by

$$P(0) = \frac{1}{n^2} F_0^2 = \frac{1}{n^2} \left[\sum_{i=0}^{n-1} f_i \right]^2 \tag{1.27}$$

Definition 1.7 *The* **Running Digital Sum** *RDS(n) after n symbols is defined to be*

$$RDS(n) = \sum_{i=0}^{n-1} f_i \tag{1.28}$$

Theorem 1.1 *A spectral null at DC ($f = 0$) is obtained if the Running Digital Sum does not grow without limits.*

This follows directly from (1.27) and taking the limit $n \to \infty$: if $\mathrm{RDS} \leq S$

$$\lim_{n\to\infty} \frac{1}{n^2} \left[\sum_{i=0}^{n-1} f_i \right]^2 \leq \lim_{n\to\infty} \frac{1}{n^2} S^2 = 0 \tag{1.29}$$

The RDS however does not have an important property of a metric. It has a sign, *i.e.*, it is not non-negative. Therefore, the variance of the RDS may be used as a metric, instead. This will then be minimized.

Zeros at other positions of the power density spectrum can easily be obtained by applying the shift properties of the Fourier transform to the metric specification.

1.5 Basic definitions and some important block codes

Selten habt ihr mich verstanden,
Selten auch verstand ich euch,
Nur wenn wir im "**Code**" (Kot) uns fanden,
So verstanden wir uns gleich.

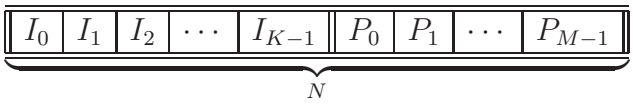
Heinrich Heine

Ye could not understand mine ire,
Nor I the tales that ye did tell,
But when we met within the "**Code**" (mire),
We knew each other very well.

1.5.1 First basic coding facts

Definition 1.8 *A codeword of a block code with length N consists of K information symbols $I_j \in \mathbb{F}$ and M parity symbols $P_j \in \mathbb{F}$, where $\mathbb{F}$ is an arbitrary number field. The ratio $R = K/N$ is the* **code rate**.

(Remark: Codes over groups or rings will not been described.)



An information block will thus be extended by a redundant part by applying a linear or even nonlinear operation. This should allow the correction (or just detection) of errors that may have occurred during transmission. We speak of a **systematic code** if the information is directly part of the codeword, without any modification.

The most important code parameter is its minimum Hamming distance.

Definition 1.9 *The* **minimum Hamming distance** *of a code $\mathcal{C}$ is given by*

$$d_{Hm} = \min_{\substack{\boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C} \\ i \neq j}} d_H(\boldsymbol{c}_i, \boldsymbol{c}_j) \; . \tag{1.30}$$

Practically, especially such codes are applied that have a linear relation between information and redundancy.

Definition 1.10 *A* **linear code** $\mathcal{C}$ *with codewords $\boldsymbol{c}_i \in \mathcal{C}$, whose components are elements of the field $\mathbb{F}$ have the property*

$$a \cdot \boldsymbol{c}_i + b \cdot \boldsymbol{c}_j \in \mathcal{C} \; , \quad a, b \in \mathbb{F} \tag{1.31}$$

The distance properties, *i.e.*, the distances to other codewords are thus independent from the actual reference codeword. This leads to an important property of linear codes:

Theorem 1.2 *The minimum Hamming distance d_{Hm} between the codewords of a linear Code $\mathcal{C}$ is equal to its* **minimum Hamming weight** *w_{Hm}.*

$$d_{Hm} = \min_{\mathbf{c} \neq 0} w_H(\mathbf{c}) = w_{Hm} \tag{1.32}$$

Proof:

$$d_{Hm} = \min_{\substack{\mathbf{c}_i, \mathbf{c}_j \in \mathcal{C} \\ i \neq j}} d_H(\mathbf{c}_i, \mathbf{c}_j) = \min_{\substack{\mathbf{c}_i, \mathbf{c}_j \in \mathcal{C} \\ i \neq j}} d_H(\mathbf{c}_i - \mathbf{c}_j, 0) = \min_{\substack{\mathbf{c} \in \mathcal{C} \\ \mathbf{c} \neq 0}} w_H(\mathbf{c}) \tag{1.33}$$

□

Theorem 1.3 *A code with the minimum distance d_{Hm} allows for the correction of $\lfloor (d_{Hm} - 1)/2 \rfloor$ errors. For even d_{Hm}, $(d_{Hm} - 2)/2$ errors can be corrected. Additionally, $d_{Hm}/2$ errors can be detected.*

For illustration, see Fig. 1.6.

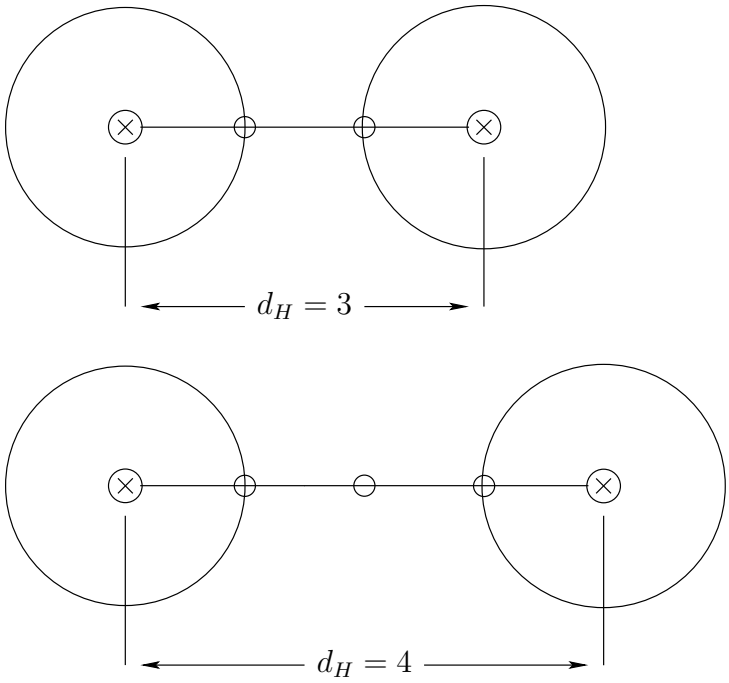


Figure 1.6: Illustration of the correction capabilities dependent on the minimum distances

Theorem 1.4 Singleton bound *[23]***:** *The minimum distance and the minimum weight of a linear (N, K) code is limited by*

$$d_{Hm} = w_{Hm} \leq 1 + N - K = 1 + M \; . \tag{1.34}$$

Codes are usually specified by the three parameters (N, K, d_{Hm}).

Proof: Let the minimum weight of a codeword be d_{Hm}.
We now consider a systematic codeword with just one information symbol different from zero and $N-K$ parity symbols (see Fig. 1.7). We obtain the maximum weight to be $1+N-K$. (qed.) □

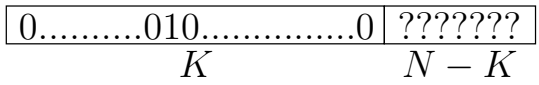


Figure 1.7: Illustration of the Singleton bound

The binary parity-check code is a simple example, where a mod-2 sum of all information bits is attached as an even parity bit. At the receiver, a single error (or an odd number of errors) can easily been detected by just adding up all received bits modulo 2.

1.5.2 Matrix description of the coding and the decoding of linear block codes

Since the relation between parity and information part of a codeword should be linear, it should be possible to check if a word belongs to the code by just applying M equations. To this end, a **parity-check matrix H** is defined.

Definition 1.11 *The parity-check matrix* $\mathbf{H}$ *is an* $(N-K) \times N$ *matrix that fulfills the following equation for all codewords* $\mathbf{c}$ *from a code* $\mathcal{C}$*.*

$$\mathbf{H} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{pmatrix} = \mathbf{H}\mathbf{c}^T = \mathbf{0} \, . \tag{1.35}$$

The parity-check matrix of the parity code would be $\mathbf{H} = (1, 1, 1, \ldots, 1)$.

Let us now consider a code that cannot just be used to detect an error, but also is able to correct a single error – the **Hamming code**. We are only discussing the Hamming code of length $N = 7$. A possible parity-check matrix would be

$$\mathbf{H} = \begin{pmatrix} 0001111 \\ 0110011 \\ 1010101 \end{pmatrix} \, . \tag{1.36}$$

In this formulation, the result of the parity-check equations, denoted as **syndrome**, has the nice property that the error position of a single error is obtained in binary notation.

Let, *e.g.*, $\mathbf{r} = (1, 0, 0, 1, 0, 0, 0)$ be a received word. We then obtain

$$\mathbf{s}^T = \mathbf{H}\mathbf{r}^T = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \text{mod } 2\,. \tag{1.37}$$

Note that inverting bit $5 = 101_2$ leads to a zero syndrome. If there is just one error, then the location is directly specified by the syndrome.

The parity-check equations can also be graphically illustrated, as shown in Fig. 1.8.

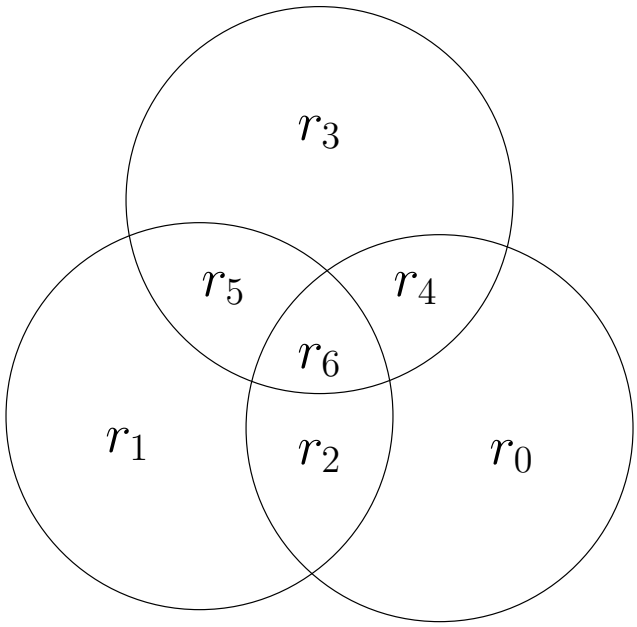


Figure 1.8: A graphical illustration of the parity-check equations of the Hamming code of length $N = 7$

Each circle stands for a parity-check equation. The error location can be seen from the overlap of non-zero parity-check equations.

We have seen that the syndrome follows from a linear combination of the columns of $\mathbf{H}$. The columns will be selected by the error locations. Superimposed valid codewords will be eliminated by the parity-check matrix. To be able to correct t errors, *i.e.*, to ensure a minimum Hamming distance of $d_{Hm} = 2t + 1$, a number of $d_{Hm} - 1 = 2t$ columns of the $\mathbf{H}$ matrix have to be linearly independent. With less linearly independent columns, it would be possible to move from one valid codeword to another with a smaller number of errors. This means, in other words, that a valid codeword with the minimum weight d_{Hm} leads to a linear combination of d_{Hm} columns of the parity-check matrix. This linear combination should result in the zero vector. This should, however, not be the case with a lower weight. Otherwise, this would also be a valid codeword and would mean that the code would have a lower minimum distance = minimum weight.

1.5.3 Equivalent representations of the parity-check matrix

It is important to note that the parity-check matrix can appear in a manifold of equivalent representations. Since valid codewords always lead to a zero syndrome in (1.35), an equivalent representation can be obtained by linearly combining the rows of the parity-check matrix. The **H** matrix of the Hamming code in (1.36) may, *e.g.*, be modified by the following linear combinations:

$$\begin{aligned} I + II &\longrightarrow I \\ I + III &\longrightarrow II \\ I + II + III &\longrightarrow III \end{aligned}$$

leading to the **systematic form**

$$\left(\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right) = (\,\boldsymbol{A}\,|\,\boldsymbol{I}\,) \tag{1.38}$$

A identity matrix $\boldsymbol{I}$ appears on the right-hand side. We obtain a coding rule that allows to derive the parity positions 4-6 from the information positions 0-3.

Furthermore, the columns of the **H** matrix may be exchanged. However, this comes with a permutation of the codeword positions. A **cyclic representation** can be obtained as follows:

$$\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array} \tag{1.39}$$

Definition 1.12 *A code $\mathcal{C}$ is called* **cyclic**, *if for $(c_0, c_1, \ldots, c_{N-1}) \in \mathcal{C}$ also follows $(c_1, c_2, \ldots, c_{N-1}, c_0) \in \mathcal{C}$.*

Up to now, we only discussed the Hamming code with parameters $(N = 7, K = 4, d_H = 3)$. The properties of binary Hamming codes of other lengths are gathered in the following table.

Properties of Hamming codes	
Length:	$N = 2^M - 1$
Number of parity positions:	M
Number of information pos. (dimension):	$K = 2^M - 1 - M$
Minimum Hamming distance:	$d_{Hm} = 3$

The parity-check equations relate parity-check and information symbols (bits). *e.g.*, we obtain from the parity-check equations with the **H**-Matrix from (1.38)

$$\begin{aligned} c_4 &= c_1 + c_2 + c_3 \\ c_5 &= c_0 + c_2 + c_3 \\ c_6 &= c_0 + c_1 + c_3 \end{aligned} \tag{1.40}$$

Let c_i, $i = 0..3$ be the information positions taken to be the unit vectors. We then obtain the four codewords

c_0	c_1	c_2	c_3	c_4	c_5	c_6
1	0	0	0	0	1	1
0	1	0	0	1	0	1
0	0	1	0	1	1	0
0	0	0	1	1	1	1

(1.41)

These are also the four linearly independent code vectors that span the vector subspace of the code.

Theorem 1.5 *The code $\mathcal{C}$ of length N and dimension K results from a linear combination of K linearly independent codewords. These linearly independent codewords constitute the so-called* **generator matrix**

Hence, codewords are obtained by the product

$$\mathbf{c} = \mathbf{iG} \tag{1.42}$$

In systematic representation, one can easily derive the corresponding **G**-Matrix for a given **H**-Matrix:

Theorem 1.6 *Let a parity-check matrix* $\mathbf{H}$ *be given in the form* $\mathbf{H} = [\mathbf{A}|\mathbf{I_{N-K}}]$. *Then the corresponding generator matrix* $\mathbf{G}$ *follows to be* $\mathbf{G} = [\mathbf{I_K}| - \mathbf{A^T}]$.

Proof: From $\mathbf{Hc^T} = \mathbf{0} \Longrightarrow$

$$[\boldsymbol{A}|\boldsymbol{I}_{N-K}] \begin{pmatrix} c_0 \\ \vdots \\ c_{N-1} \end{pmatrix} = 0$$

$$\begin{pmatrix} c_K \\ \vdots \\ c_{N-1} \end{pmatrix} = -\boldsymbol{A} \begin{pmatrix} c_0 \\ \vdots \\ c_{K-1} \end{pmatrix} = -\boldsymbol{A} \begin{pmatrix} i_0 \\ \vdots \\ i_{K-1} \end{pmatrix}$$

With

$$\begin{pmatrix} c_0 \\ \vdots \\ c_{K-1} \end{pmatrix} = \boldsymbol{I}_K \begin{pmatrix} i_0 \\ \vdots \\ i_{K-1} \end{pmatrix} ,$$

we obtain

$$\begin{pmatrix} c_0 \\ \vdots \\ c_{N-1} \end{pmatrix} = \left[\frac{\boldsymbol{I}_K}{-\boldsymbol{A}}\right] \begin{pmatrix} i_0 \\ \vdots \\ i_{K-1} \end{pmatrix}$$

Taking the transposes, we finally get

$$\mathbf{c} = \mathbf{iG} \quad \text{with} \quad \mathbf{G} = [\mathbf{I_K}| - \mathbf{A^T}] .$$

□

1.5.4 Equivalent representations of the generator matrix

Similar to the modifications of the parity-check matrix in Section 1.5.3, we observe: linear combinations of codewords are still valid codewords. This means that linear combinations of the rows of the generator matrix do not change the code. Permutations of the columns, however, do also mean a corresponding permutation of the bit positions in the codeword. This has to be taken into consideration for the parity-check matrix, as well.

1.5.5 Extending Codewords

One often used extension is the attachment of an additional parity position, thereby increasing the codeword length by one. If the original code had an odd minimum Hamming distance (weight), the minimum distance (weight) will be increase by one, as well.

For the parity equations one additional parity sum is required. The new parity-check matrix **H** will be

$$\mathbf{H_e} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ & & & & 0 \\ & & & & 0 \\ & \mathbf{H} & & & \vdots \\ & & & & 0 \end{pmatrix} \tag{1.43}$$

The extended Hamming code with the parameters (8,4,4) has an **H** matrix

$$\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{array} \tag{1.44}$$

As far as the generator matrix $\mathbf{G}$ is concerned, this just means the extension by one parity position, which results from the modulo-2 sum of all other positions.

$$\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array} \tag{1.45}$$

This extended Hamming code is also the first-order Reed-Muller code of length 2^3.

1.5.6 Reed-Muller codes

An early description of the repetition code ?

> But let your speech be, Yea, yea; Nay, nay:
> and whatsoever is more than these is of the evil one
>
> Matthew 5:37

An 'r-th order' Reed-Muller code RM(r, m) of length 2^m can be defined by a blocked generator matrix as follows

$$\mathbf{G} = \begin{pmatrix} \mathbf{G_0} \\ \mathbf{G_1} \\ \vdots \\ \mathbf{G_r} \end{pmatrix}, \tag{1.46}$$

where $\mathbf{G_0}$ is the all-ones vector of length 2^m (repetition code!). $\mathbf{G_1}$ is an $m \times 2^m$ matrix consisting of all binary m-tuples appearing only once. $\mathbf{G_l}$ $(2 \leq l \leq r)$ results from all different products of l rows of $\mathbf{G_1}$.

The number of information bits is thus

$$K = 1 + \binom{m}{1} + \cdots + \binom{m}{r}. \tag{1.47}$$

The minimum Hamming distance d_{Hm} is $d_{Hm} = 2^{m-r}$ (without proof).

As an example, we show the generator matrix of the 3rd order RM code of length $2^4 = 16$:

$$\mathbf{G}\left\{\begin{array}{lll} \mathbf{G}_0 = & \left[\begin{array}{c} 1111111111111111 \end{array}\right] & = [\mathbf{c}_0] \\ \mathbf{G}_1 = & \left[\begin{array}{c} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \end{array}\right] & = \left[\begin{array}{c} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \\ \mathbf{c}_4 \end{array}\right] \\ \mathbf{G}_2 = & \left[\begin{array}{c} 0000000000001111 \\ 0000000000110011 \\ 0000000001010101 \\ 0000001100000011 \\ 0000010100000101 \\ 0001000100010001 \end{array}\right] & = \left[\begin{array}{c} \mathbf{c}_1\mathbf{c}_2 \\ \mathbf{c}_1\mathbf{c}_3 \\ \mathbf{c}_1\mathbf{c}_4 \\ \mathbf{c}_2\mathbf{c}_3 \\ \mathbf{c}_2\mathbf{c}_4 \\ \mathbf{c}_3\mathbf{c}_4 \end{array}\right] \\ \mathbf{G}_3 = & \left[\begin{array}{c} 0000000000000011 \\ 0000000000000101 \\ 0000000000010001 \\ 0000000100000001 \end{array}\right] & = \left[\begin{array}{c} \mathbf{c}_1\mathbf{c}_2\mathbf{c}_3 \\ \mathbf{c}_1\mathbf{c}_2\mathbf{c}_4 \\ \mathbf{c}_1\mathbf{c}_3\mathbf{c}_4 \\ \mathbf{c}_2\mathbf{c}_3\mathbf{c}_4 \end{array}\right] \end{array}\right. . \tag{1.48}$$

Obviously,

$$\mathrm{RM}(0,m) \subset \mathrm{RM}(1,m) \subset \mathrm{RM}(2,m) \subset \mathrm{RM}(3,m) \; . \tag{1.49}$$

Search for the generator matrix of the extended Hamming code (16,11,4) inside the Reed-Muller $RM(3,4)$ generator matrix!

1.5.7 Perfect codes

Theorem 1.7 Sphere-packing or Hamming bound:

Let a code with parameters (N,K) have a correcting capability of t.

Binary case:

$$2^K \left(1 + \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{t}\right) \leq 2^N \tag{1.50}$$

Non-binary case over a field with q elements.

$$q^K \left(1 + \binom{N}{1}(q-1) + \binom{N}{2}(q-1)^2 + \cdots + \binom{N}{t}(q-1)^t\right) \leq q^N \tag{1.51}$$

This theorem [24] follows from simple counting of all possible vectors within an n-dimensional 'sphere' of 'radius' t according to Fig. 1.9. These are all possible vectors of weight less or equal to t. All vectors inside of all correction spheres follow from a multiplication with 2^K, the number of codewords. This number, of course, has always to be smaller than the number of all possible

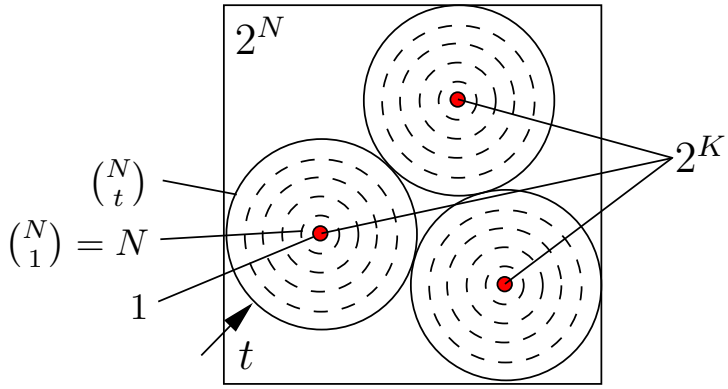


Figure 1.9: Illustration of the sphere-packing bound

vectors of length n. The non-binary case is an obvious extension, since there are $q-1$ possible error values.

Definition 1.13 *A* **perfect code** *fulfills the Hamming bound with equality,* i.e.*, no words are left outside of the decoding spheres.*

Only three different code classes do exist that are perfect:

1. binary repetition code with odd length N
2. Hamming codes
3. binary and ternary Golay code

The **repetition code**, as the name implies, has a generator matrix

$$\mathbf{G} = \underbrace{(1, 1, 1, \ldots, 1)}_{N} \tag{1.52}$$

and the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & & 0 \\ 1 & 0 & 0 & 1 & & 0 \\ \vdots & & & & \ddots & \\ 1 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \tag{1.53}$$

The repetition code is **dual** to the parity-check code, hence, the above G and H matrices of the repetition code are the same as the H and G matrices of the parity-check code, respectively.

Definition 1.14 *Let the generator and parity-check matrices of a code $\mathcal{C}$ be $\mathbf{G}$ and $\mathbf{H}$, respectively. Its dual code $\mathcal{C}^\perp$ has a generator matrix $\mathbf{G}^\perp = \mathbf{H}$ and a parity-check matrix $\mathbf{H}^\perp = \mathbf{G}$.*

We have

$$\begin{aligned} \mathbf{H}\,\mathbf{G}^T &= 0 \\ \mathbf{G}\,\mathbf{H}^T &= 0 \end{aligned} \tag{1.54}$$

Golay codes

The binary Golay code has the parameters (23,12,7), extended (24,12,8), and the ternary one has the parameters (11,6,5).

Theorem 1.8 *The extended binary Golay code is* **self-dual**, i.e., $\mathbf{G}_{24}^{\perp} = \mathbf{G}_{24}$.

Note that duality is not the same as orthogonality. There is no orthogonality in finite fields. In [25], Massey defines so-called linear codes with complementary duals to model orthogonality to some extent. To make the difference clear, consider the scalar product of two vectors $\mathbf{a}$ and $\mathbf{b}$. If $\mathbf{a} \cdot \mathbf{b}^H = 0$, one would conclude the two vectors to be orthogonal when the components are real or complex. Now consider multiplying tow exemplary vectors

$$\left[\begin{array}{ccccccc} 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right] \left[\begin{array}{c} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{array}\right] = 0 \quad \text{modulo 2}$$

The vectors are the same and hence not orthogonal. We obtain a zero result for all vectors of even weight. We clearly see that the notion of orthogonality is not suitable for finite fields, e.g., for the binary number field. This also means that projection to find a least-squares optimum over the reals or complex numbers does not have a counterpart over finite fields.

The self-dual generator matrix of the binary extended Golay code is shown in (1.55).

$$\mathbf{G_{24}} = \left(\begin{array}{c|cccccccccccc|c|cccccccccccc} 1 & 1 & & & & & & & & & & & & & 1 & 1 & & 1 & 1 & 1 & & & & 1 & \\ 1 & & 1 & & & & & & & & & & & & & 1 & 1 & & 1 & 1 & 1 & & & & 1 \\ 1 & & & 1 & & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 & & & \\ 1 & & & & 1 & & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 & & \\ 1 & & & & & 1 & & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 & \\ 1 & & & & & & 1 & & & & & & & & & & & 1 & & 1 & 1 & & 1 & 1 & 1 \\ 1 & & & & & & & 1 & & & & & & & 1 & & & & 1 & & 1 & 1 & & 1 & 1 \\ 1 & & & & & & & & 1 & & & & & & 1 & & & & 1 & & 1 & 1 & & 1 & 1 \\ 1 & & & & & & & & & 1 & & & & & 1 & 1 & & & & 1 & & 1 & 1 & & 1 \\ 1 & & & & & & & & & & 1 & & & & 1 & 1 & 1 & & & & 1 & & 1 & 1 & \\ 1 & & & & & & & & & & & 1 & & & & 1 & 1 & 1 & & & & 1 & & 1 & 1 \\ 1 & & & & & & & & & & & & 1 & & 1 & & 1 & 1 & 1 & & & & 1 & & 1 \\ \hline & & & & & & & & & & & & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right) \tag{1.55}$$

1.5.8 Decoding with the standard array

The standard array is a table of all q^N words with components from a field with q elements. The first row consists of the codewords, starting from the all-zero word. Along the columns, error vectors will be added to the codewords, until all possible q^N vectors are tabulated. If one would stop at the error-correcting capability t (**designed distance**) as the maximum weight of error vectors, a non-complete error correction will be represented by the table. Non-tabulated words would not be assigned to codeword and would thus stay uncorrected. The error vectors that are added to the zero codeword are called **coset leaders**, the corresponding row is the **coset**. Cosets have the same distance properties as the original code, but they are nonlinear (no zero codeword). The columns down to the error pattern with weight t are denoted as **decoding spheres**.

Standard array:

			decoding sphere		
	$\mathbf{0}$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\cdots$	$\mathbf{c}_{q^k}$
	$\mathbf{0}+\mathbf{e}_1$	$\mathbf{c}_2+\mathbf{e}_1$	$\mathbf{c}_3+\mathbf{e}_1$	$\cdots$	$\mathbf{c}_{q^k}+\mathbf{e}_1$
coset	$\mathbf{0}+\mathbf{e}_2$	$\mathbf{c}_2+\mathbf{e}_2$	$\mathbf{c}_3+\mathbf{e}_2$	$\cdots$	$\mathbf{c}_{q^k}+\mathbf{e}_2$
	$\vdots$				$\vdots$
	$\mathbf{0}+\mathbf{e}_j$	$\mathbf{c}_2+\mathbf{e}_j$	$\mathbf{c}_3+\mathbf{e}_j$	$\cdots$	$\mathbf{c}_{q^k}+\mathbf{e}_j$
	$\mathbf{0}+\mathbf{e}_{j+1}$	$\mathbf{c}_2+\mathbf{e}_{j+1}$	$\mathbf{c}_3+e_{j+1}$	$\cdots$	$\mathbf{c}_{q^k}+\mathbf{e}_{j+1}$
	$\vdots$				$\vdots$
	$\mathbf{0}+\mathbf{e}_l$	$\mathbf{c}_2+\mathbf{e}_l$	$\mathbf{c}_3+\mathbf{e}_l$	$\cdots$	$\mathbf{c}_{q^k}+\mathbf{e}_l$
	coset leader				

$$j = \sum_{i=1}^{t} \binom{N}{i} (q-1)^i$$

Chapter 8

Reed-Solomon codes

Reed-Solomon codes are one of the most important codes in applications. They are applied in numerous modern transmission and storage system. One may just think of the CD player, ADSL, optical and wireless transmission. These block codes are not bit-, but symbol-oriented, a symbol consisting of some bits (over $GF(2^m)$, *e.g.*, a byte). They are thus very suited for the correction of bursts. For practical applications, the existence of powerful en- and decoding algorithms are essential. RS codes fulfil the Singleton bound with equality. This property is also named *MDS (maximum distance separable)* . It is also advantageous that the weight distribution can be computed easily. This simplifies error probability approximations. RS codes are based on the discrete Fourier transform. Before we will introduce it over finite fields, in the following section, we will illustrate encoding (and decoding) as a polynomial interpolation.

8.1 Encoding as a polynomial interpolation

A polynomial $a(x) = a_0 + a_1 x + \cdots + a_{K-1}x^{K-1}$ of degree $K-1$ can be formulated dependent on given samples by means of so-called 'continuous' Kronecker delta functions $\delta_j(x)$. $a(x)$ is then given by the Lagrange interpolation

$$a(x) = \sum_{j=0}^{K-1} \delta_j(x) \cdot a(x_j) \tag{8.1}$$

$$\begin{aligned} \delta_j(x) &= \frac{\prod\limits_{\substack{k=0,\ldots,K-1 \\ k\neq j}} (x - x_k)}{\prod\limits_{\substack{k=0,\ldots,K-1 \\ k\neq j}} (x_j - x_k)} \\ &= \frac{\prod_{k=0}^{K-1}(x - x_k)}{(x - x_j) \cdot \left[\frac{d}{dx}\prod_{k=0}^{K-1}(x - x_k)\right]_{x=x_j}} \end{aligned} \tag{8.2}$$

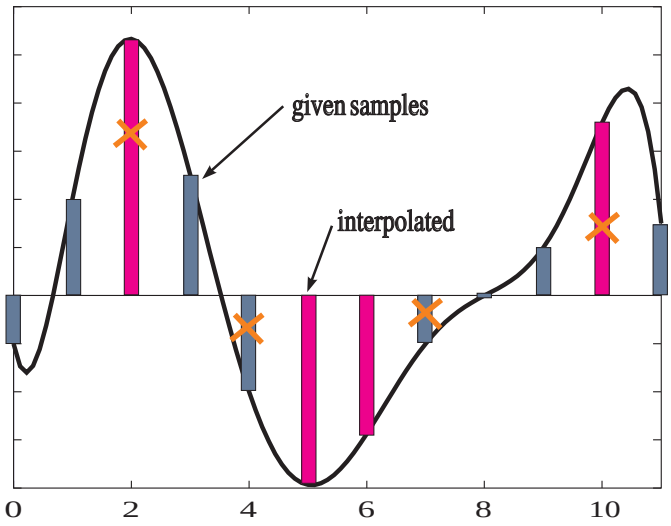


Figure 8.1: Encoding as a polynomial interpolation

For the delta function, we have

$$\delta_j(x) = \begin{cases} 1 & \text{if } x = x_j \\ 0 & \text{if } x = x_k\,,\ k = 0, \ldots, K-1\,,\ x \neq x_j \end{cases} \tag{8.3}$$

With the delta functions, it is straight forward to compute further $N-K$ samples as a function of the K given samples assuming that the underlying polynomial is of limited degree $K-1$. Figure 8.1 shows an interpolating polynomial of degree 7 with 8 predefined samples. Four further samples have been computed. These four samples are redundant, since the interpolating polynomial is already uniquely defined by 8 samples. Adding these four samples can be considered as an encoding. One may think of the 12 samples as a transmitted codeword. Let us imagine that four errors have occurred at known positions. Since the polynomial is uniquely defined by any arbitrary 8 samples, with 8 error-free samples, all 12 samples can again be recomputed. This procedure is called erasure decoding, the correction of errors at known positions.

In the following section, we will introduce the link between the polynomial interpolation and the DFT. To illustrate this already here, let us think of time-domain codeword samples to be located around the unit circle as shown in Fig. 8.2. For drawing purposes, we are additionally forcing the time domain to be real by requiring DFT-domain components to be conjugate symmetric. This, the reader might later recognize as a complex BCH code after having read Chapter 13. In the figure, red bars stand for information symbols in time domain (original domain), while green ones represent redundant ones.

8.2 Reed-Solomon codes and the discrete Fourier transform

In a finite field, only a finite number of samples can be used. This results from the powers of an *element of order* N. Let us, *e.g.*, consider all the elements of an extension field $GF(P^m)$. Then there are $P^m - 1$ elements that can be generated as powers of the primitive element. This means

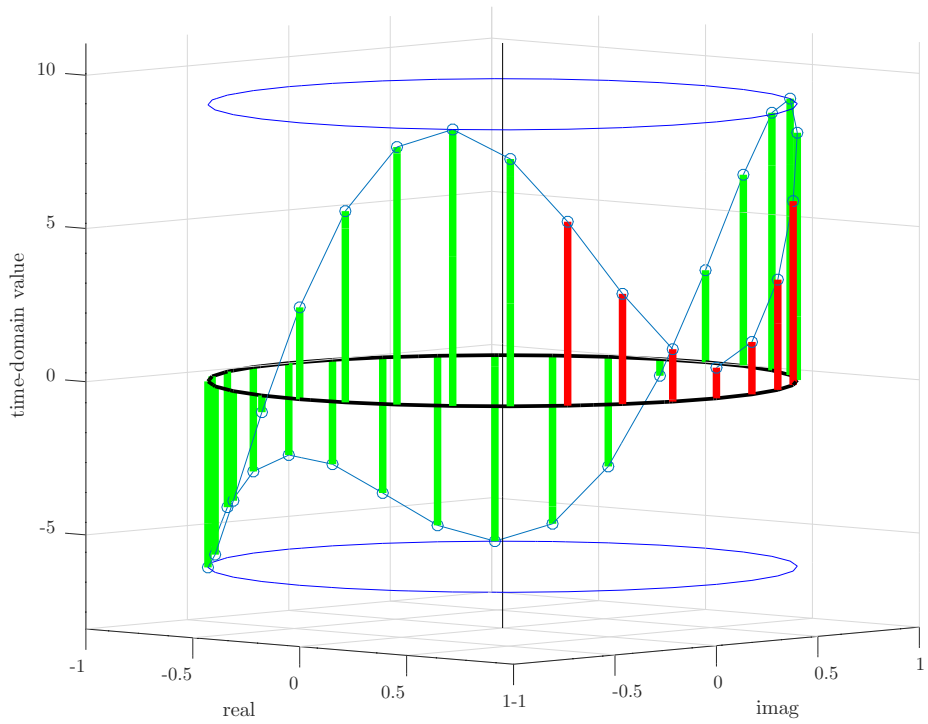


Figure 8.2: Analog BCH code (RS code with conjugacy constraints)

that the possible code length would be $N = P^m - 1$. The values at these positions again result from a polynomial of degree $K-1$. In an extension field, however, there are also elements of order N with $N|P^m - 1$.

From the limited degree of the polynomial $C(x)$, one can immediately make statements regarding the minimum distance = minimum weight of the resulting codes. At first, we recall the so-called **fundamental theorem of algebra**.

Theorem 8.1 *A polynomial* $C(x) = C_0 + C_1 x + C_2 x^2 + \ldots C_{K-1} x^{K-1}$ *of degree* $K-1$ *has at most* $K-1$ *different roots* x_j*. Hence, the polynomial can be factorized as* $C(x) = C_{K-1} \cdot (x - x_1) \cdot (x - x_2) \cdots (x - x_{K-1}) = C_{K-1} \cdot \prod_{j=1}^{K-1} (x - x_j)$.

With this, we can immediately provide a statement on the minimum weight and with it on the minimum distance of a code defined by a degree-limited polynomial.

Theorem 8.2 *Let* $C(x)$ *be a polynomial of degree* $K - 1 = N - M - 1$ *with arbitrary coefficients from a field* $\mathbb{F}$*. If we compute the values of the polynomial at* N *different positions* $x = x_j$, $x_j \in \mathbb{F}$, $j = 0, \ldots, N-1$*, the vector of* N *samples* $(C(x_0), C(x_1), \ldots, C(x_{N-1}))$ *has minimum weight* $w_{Hm} = M + 1 = N - K + 1$*. Since the sum of two such vectors is equivalent to the sum of the corresponding polynomial coefficients, the sum of vectors fulfills the degree limitation, too. Thus, it is a linear code. The minimum distance is equal to the minimum weight.*

Proof: The number of zeros cannot exceed $K - 1 = N - M - 1$. Hence, at least, $N - (K-1) = M + 1$ positions have to be non-zero. □

This theorem also specifies an important property that is named as follows:

Definition 8.1 *A* **Maximum Distance Separable (MDS)** *code fulfills the Singleton bound ($d_{Hm} \leq M + 1 = N - K + 1$) with equality.*

According to the previous theorem this is the case. We will now formally define Reed-Solomon codes. (This definition will later be generalized.)

Definition 8.2 *A* **Reed-Solomon (RS) code** *of length N and minimum Hamming distance d_{Hm} is the set of vectors, whose components are values of a polynomial $C(x)$ of degree $\leq K - 1 = N - d_{Hm}$ at the positions z^j, with z being an element of order N from an arbitrary number field. We are, of course, especially interested in $z \in GF(P^m)$, $z^N = 1$, $z^j \neq 1$ for $0 < j < N$.*

$$c = (c_0, \ldots, c_{N-1}) \ , \quad c_j = C(x = z^j) \tag{8.4}$$

Example 8.1 *$GF(P = 7)$, primitive element: $z = 5$, order $N = P - 1 = 6$*
We specify an RS code of length $N = 6$ with correction capability of $t = 2$ symbols. From $d_{Hm} \geq 2 \cdot t + 1$ follows the maximum degree of the polynomial $K - 1 = N - d_{Hm} = 6 - 5 = 1$. The degree-limited polynomial exists only of two components C_0, C_1. Since every component can assume $P = 7$ values, the code consists of $7 \cdot 7 = 49$ codewords.

Let the information be localized in the coefficients of the polynomial. With $C(x) = C_0 + C_1 x = 3 + 1 \cdot x$, we obtain

$$\begin{aligned} c_0 &= C(x = z^0 = 1) = 3 + 1 \cdot 1 = 4 \\ c_1 &= C(x = z^1 = 5) = 3 + 1 \cdot 5 = 1 \\ c_2 &= C(x = z^2 = 4) = 3 + 1 \cdot 4 = 0 \\ &\vdots \\ c &= (c_0, \ldots, c_5) = (4, 1, 0, 2, 5, 6) \end{aligned}$$

The definition of RS codes by means of a polynomial at positions z^j with z being an element of order N reminds us to the well-known discrete Fourier transform that we will write in a maybe somewhat unusual form. Let us for a moment assume that the considered polynomial $C(x)$ has full degree $N - 1$. Then we would have

$$c_j = C(x = z^j) = \sum_{k=0}^{N-1} C_k z^{jk} \tag{8.5}$$

This is nothing else than the IDFT from the DFT domain into the original domain. In the following, we again define forward and backward transforms of the DFT. We write the inverse transform first, since we have already introduced it. We could also have defined the first relation as the 'forward' transform; however, we would like to have a relation to time and frequency domain.

Definition 8.3 *Let $\mathcal{C} = (C_0, C_1, \ldots, C_{N-1})$ be the coefficient vector of a polynomial $C(x) = C_0 + C_1 x + \cdots + C_{N-1}x^{N-1}$ and let z be an element of order N (*i.e.*, $z^N = 1$, but $z^j \neq 1$ for $0 < j < N$). The* **discrete Fourier transform (DFT)** *and its inverse transform (IDFT) are defined to be*

IDFT:

$$c_j = C(x = z^j) = \sum_{k=0}^{N-1} C_k z^{jk} \ , \quad j = 0, 1, \ldots, N-1 \ , \tag{8.6}$$

DFT:

$$C_k = N^{-1} \cdot c(x = z^{-k}) = N^{-1} \cdot \sum_{j=0}^{N-1} c_j z^{-jk} \ , \quad k = 0, 1, \ldots, N-1 \ , \tag{8.7}$$

where we again have written the vector $\mathbf{c} = (c_0, c_1, \ldots, c_{N-1})$ as a polynomial $c(x) = c_0 + c_1 x + \cdots + c_{N-1}x^{N-1}$.

Note the alternative description of vectors as polynomials which is often used in coding theory.

We will provide the proof that the DFT transform is uniquely invertible.

Proof:

We start from (8.7):

$$\begin{aligned} N^{-1} \cdot c(x = z^{-k}) &= N^{-1} \cdot \sum_{j=0}^{N-1} c_j z^{-jk} \\ &= N^{-1} \cdot \sum_{j=0}^{N-1} \left(\sum_{l=0}^{N-1} C_l z^{jl} \right) z^{-jk} \\ &= N^{-1} \cdot \sum_{l=0}^{N-1} C_l \sum_{j=0}^{N-1} z^{j(l-k)} \\ &= N^{-1} \cdot \sum_{l=0}^{N-1} C_l \sum_{j=0}^{N-1} \gamma^j \ , \quad \gamma = z^{l-k} \end{aligned}$$

The geometric series $\sum_{j=0}^{N-1} \gamma^j$ is equal to

$$\sum_{j=0}^{N-1} \gamma^j = \begin{cases} N & \text{for } \gamma = z^{l-k} = 1, \ i.e., \ l = k \\ \frac{\gamma^N - 1}{\gamma - 1} = 0 & \text{for } \gamma = z^{l-k} \neq 1, \ i.e., \ l \neq k, \ \text{but } \gamma^N = z^{N(l-k)} = 1 \end{cases}$$

Hence, in the transformation, only $N^{-1} \cdot c(x = z^{-k}) = N^{-1} \cdot C_k \cdot N$ is left, as required. □

In a prime field $GF(P)$, we have for $N = P - 1$: $N^{-1} = N = P - 1$, since $(P-1)(P-1) = P^2 - 2P + 1 = 1 \bmod P$. In an extension field of the binary numbers $GF(2^m)$, we even have $1/N = 1/(2^m - 1) = 1$.

In the following, we will recall some important properties of the discrete Fourier transform; especially, we are interested in the convolution and shift properties. Hereto, we note that in the polynomials we replace x by a power of an element of order N, *i.e.*, x has the property $x^N = 1$. This allows to compute mod $(x^N - 1)$, *i.e.*, x^N can be replaced by 1.

First, we will apply this property when multiplying two polynomials. Since we compute mod $(x^N - 1)$, the result will not have a degree higher than $N - 1$.

$$(a_0 + a_1 x + \cdots + a_{N-1} x^{N-1}) \cdot (b_0 + b_1 x + \cdots + b_{N-1} x^{N-1}) = c_0 + c_1 x + \cdots + c_{N-1} x^{N-1} \quad (8.8)$$

$$\begin{aligned} c_0 &= b_0 \cdot a_0 + b_1 \cdot a_{N-1} + b_2 \cdot a_{N-2} + \cdots + b_{N-1} \cdot a_1 \\ c_1 &= b_0 \cdot a_1 + b_1 \cdot a_0 + b_2 \cdot a_{N-1} + \cdots + b_{N-1} \cdot a_2 \\ &\vdots \\ c_j &= \sum_{l=0}^{N-1} b_l \cdot a_{j-l \bmod N} \end{aligned}$$

This is nothing else than the *cyclic convolution* of the coefficients of the vectors $\mathbf{a} = (a_0, a_1, \ldots, a_{N-1})$ and $\mathbf{b} = (b_0, b_1, \ldots, b_{N-1})$ that we are used to indicate with a '$\star$'.

$$\mathbf{a} \star \mathbf{b} \equiv a(x) \cdot b(x) \bmod (x^N - 1) \quad (8.9)$$

The following table (Eq. (8.10)) collects the convolution and shift properties of the DFT.

Original domain	DFT domain
$c_j = a_j \cdot b_j$	$C(x) = A(x) \cdot B(x)\,,$ $C_k = \sum_{l=0}^{N-1} A_l \cdot B_{k-l \bmod N}$
$c(x) = N^{-1} a(x) \cdot b(x)\,,$ $c_j = N^{-1} \cdot \sum_{l=0}^{N-1} a_l \cdot b_{j-l \bmod N}$	$A_k \cdot B_k$
$c_{j \bmod N} = a_{j-l \bmod N}\,,$ $c(x) = a(x) \cdot x^l \bmod (x^N - 1)$	$C_k = A_k \cdot z^{-k \cdot l}$
$c_j = a_j \cdot z^{j \cdot l}$	$C_{k \bmod N} = A_{k-l \bmod N}\,,$ $C(x) = A(x) \cdot x^l \bmod (x^N - 1)$

(8.10)

We shortly proof the two convolution theorems.

Proof:

1^{st} convolution theorem:

$$c_j = C(x = z^j) = A(x = z^j) \cdot B(x = z^j) = a_j \cdot b_j$$

2^{nd} convolution theorem:

$$C_k = N^{-1} \cdot c(x = z^{-k}) = N^{-1} \cdot a(x = z^{-k}) \cdot N^{-1} \cdot b(x = z^{-k}) = A_k \cdot B_k$$

□

From the shift theorem in the DFT domain, we conclude an important property of RS codes allowing us to extend their definition. The shift theorem tells us that zeros stay unchanged and no new ones will be added when cyclically shifting the transform domain, *i.e.*, the minimum distance is preserved.

Definition 8.4 *(extended version) A* **Reed-Solomon (RS) code** *of length N and minimum Hamming distance d_{Hm} is a set of vectors, whose components are the values of a polynomial $C(x) = x^j \cdot C'(x)$ of degree$\{C'(x)\} \leq K - 1 = N - d_{Hm}$ at positions z^j, with z being an element of order N from an arbitrary number field.*

$$c = (c_0, \ldots, c_{N-1})\,, \quad c_j = C(x = z^j) \tag{8.11}$$

Before we will proceed to means of encoding, we will discuss some properties that will then be required.

First, we study the case when a product of two polynomials $a(x) \cdot b(x) = 0 \bmod (x^N - 1) = (\ldots) \cdot (x^N - 1)$. Since the product of two polynomials corresponds to the convolution of their coefficients, we obtain under the additional assumption that $b(x)$ is monic (degree$\{b(x)\} = K < N - 1$, $b_K = 1$)

$$0 = \sum_{l=0}^{N-1} b_l \cdot a_{j-l \bmod N} = 1 \cdot a_{j-K \bmod N} + \sum_{l=0}^{K-1} b_l \cdot a_{j-l \bmod N} \tag{8.12}$$

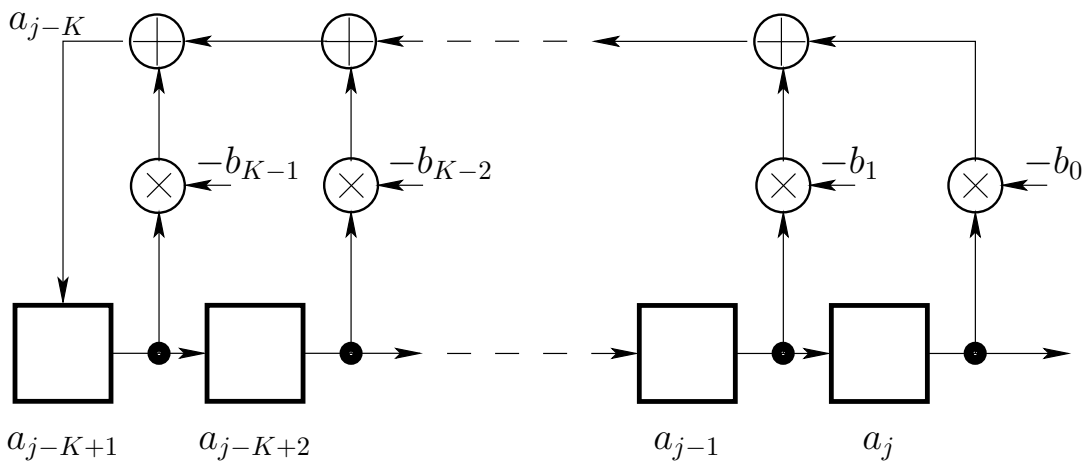


Figure 8.3: Shift register of the convolution in Eq. (8.12)

This recursion can be realized by the shift register in Fig. 8.3.

Furthermore, we will require the fact that zeros in the original or transform domain will correspond to linear factors of the polynomial description in the transform or original domain, respectively.

$$c_j = 0 \iff C(x) = (\dots) \cdot (x - z^{+j}) \tag{8.13}$$

$$C_{N-j} = 0 \iff c(x) = (\dots) \cdot (x - z^{+j}) \tag{8.14}$$

All possible linear factors together yield

$$x^N - 1 = \prod_{j=0}^{N-1} (x - z^j) \tag{8.15}$$

This follows from the fact that $(x^N - 1)$ can at most have N roots. Furthermore, since z is an element of order N, all N different powers z^j, $j = 0, 1, \dots, N-1$ are roots of $x^N - 1$ ($(z^j)^N = (z^N)^j = 1$).

8.3 Encoding

We have described RS codes by a degree-limited polynomial, *i.e.*, requiring $M = 2t$ subsequent zeros from index K to $N-1$ of the DFT vector (not mentioning cylic shifts). How will we now choose the relation between K information symbols and the codewords? In principal, there are two possibilities. On one hand, the information can, of course, directly be chosen to be the K DFT-domain samples. An IDFT yields the code vector. On the other hand, the K samples can be specified in time domain and a polynomial interpolation would yield the other samples taking into account the limited degree of the polynomial. We thus distinguish between methods in DFT and in the original (time) domain. Usually, one of four different methods is applied. In the sequel, we will describe these four options.

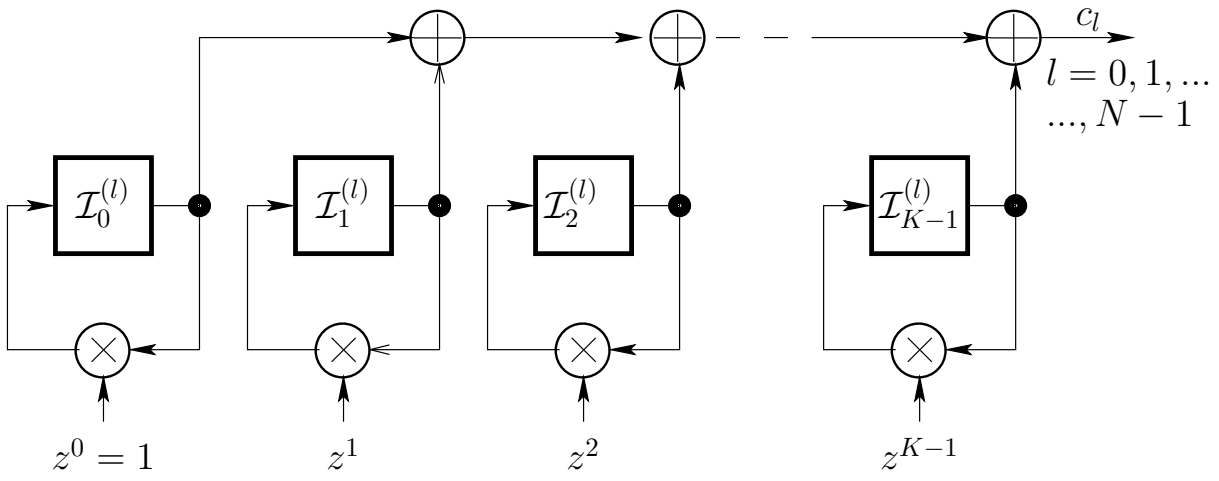


Figure 8.4: IDFT shift register

8.3.1 Encoding in DFT domain

As has been mentioned, the information can just be put into $K = N - 2t$ subsequent positions in DFT domain. We first write the IDFT as a matrix operation.

$$\mathbf{c} = \mathbf{C} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & z^1 & z^2 & z^3 & \\ 1 & z^2 & z^4 & z^6 & \\ 1 & z^3 & z^6 & z^9 & \\ \vdots & & & & \ddots \end{pmatrix} \tag{8.16}$$

Let the information be $\mathbf{C} = (I_0, I_1, \ldots, I_{K-1}, \underbrace{0, \ldots, 0}_{2t = d_{H\,m} - 1})$. This yields the set of linear equations

$$\mathbf{c} = (I_0, I_1, \ldots, I_{K-1}) \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & z^1 & z^2 & z^3 & & \\ 1 & z^2 & z^4 & z^6 & & \\ \vdots & & & & & \\ 1 & z^{K-1} & z^{2(K-1)} & z^{3(K-1)} & \ldots & z^{(N-1)(K-1)} \end{pmatrix} . \tag{8.17}$$

In order to realize this as a shift register, we rewrite the equations, following the so-called Horner scheme, as

$$\begin{array}{llllll} c_0 = C(z^0 = 1) & = I_0 & +I_1 & +I_2 & +\cdots+ I_{K-1} & \\ c_1 = C(z^1) & = I_0 & +I_1 \cdot z^1 & +I_2 \cdot z^2 & +\cdots+ I_{K-1} \cdot z^{K-1} & \\ c_2 = C(z^2) & = I_0 & +(I_1 \cdot z^1) \cdot z^1 & +(I_2 \cdot z^2) \cdot z^2 & +\cdots+ (I_{K-1} \cdot z^{K-1}) \cdot z^{K-1} & . \\ \vdots & & & & & \end{array} \tag{8.18}$$

For the term of a sum denoted as $\mathcal{I}_j^{(l)}$, we obtain the recursion $\mathcal{I}_j^{(l+1)} = \mathcal{I}_j^{(l)} \cdot z^j$ with $\mathcal{I}_j^{(0)} = I_j$. This representation leads to the shift register shown in Fig. 8.4.

As an example, we study the code (6,2,5) over GF(7) with the primitive element $z = 5$ ($z^0 = 1$,

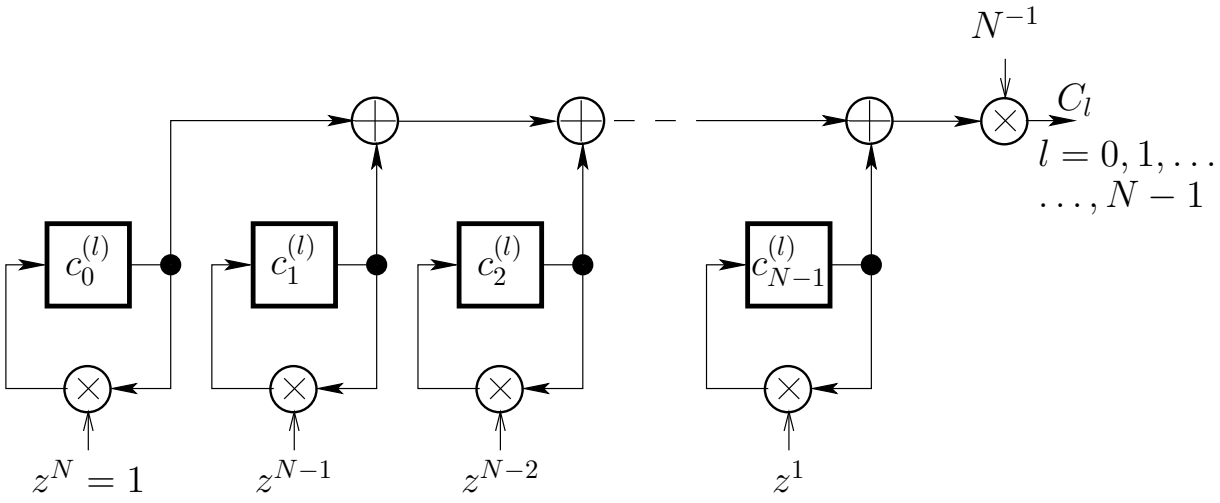


Figure 8.5: DFT shift register

$z^1 = 5$). With $(I_0, I_1) = (3, 1)$, we subsequently obtain

	$\cdot 1$	$\cdot 5$	
l	$\mathcal{I}_0^l$	$\mathcal{I}_1^l$	$\longrightarrow c_l = \sum_j \mathcal{I}_j^l$
0	3	1	$\longrightarrow 4$
1	3	5	$\longrightarrow 1$
2	3	4	$\longrightarrow 0$
3	3	6	$\longrightarrow 2$
4	3	2	$\longrightarrow 5$
5	3	3	$\longrightarrow 6$

$$\mathbf{c} = (4, 1, 0, 2, 5, 6)$$

Since the DFT differs from the IDFT only in the factor N^{-1} and the sign of the powers of the element of order N, the same matrix description applies also to $C_j = N^{-1} \cdot c(x = z^{N-j})$, leading to the DFT shift register in Fig. 8.5.

As a check, one may compute the DFT vector $(C_0, C_1, \ldots, C_{N-1})$ from the time-domain vector $\mathbf{c} = (4, 1, 0, 2, 5, 6)$.

	$\cdot 1$	$\cdot 3$	$\cdot 2$	$\cdot 6$	$\cdot 4$	$\cdot 5$	
l	c_0^l	c_1^l	c_2^l	c_3^l	c_4^l	c_5^l	$\longrightarrow C_l = N^{-1} \cdot \sum_j c_j^{(l)}$
0							$\longrightarrow 3$
1							$\longrightarrow 1$
2							$\longrightarrow 0$
3							$\longrightarrow 0$
4							$\longrightarrow 0$
5							$\longrightarrow 0$

Since the encoding in DFT domain will not lead to a systematic codeword in time domain, *i.e.*, the information will not be visible in the codeword, this encoding procedure will usually not be selected. The same applies for the non-systematic encoding directly in time domain, which will be described in the next section. However, this will be the basis for the next two encoding methods.

8.3.2 Non-systematic encoding with the generator polynomial

The degree limitation of the polynomial $C(x)$ (degree$\{C(x)\} \leq K-1 = N-d_{Hm}$) can be described by linear factors in time domain, since

$$C_{N-j} = N^{-1} \cdot c(x = z^{-(N-j)}) = N^{-1} \cdot c(x = z^j) = 0 \ , j = 1, ..., M \ . \tag{8.19}$$

$c(x)$ has to be a multiple of

$$g(x) = \prod_{j:C_{N-j}=0} (x - z^j) = g_0 + g_1 \cdot x + \cdots + 1 \cdot x^{2t} \ . \tag{8.20}$$

In the special case of zeros at the upper end, which we will mostly consider, this means

$$g(x) = \prod_{j=1}^{2t} (x - z^j) \ . \tag{8.21}$$

$g(x)$ is called **generator polynomial** due to its relation to the generator matrix. How the generator matrix is actually derived from the generator polynomial will be described later.
A codeword in polynomial representation $c(x)$ thus follows from the product

$$c(x) = \underbrace{i(x)}_{\text{degree: } K-1=N-2t-1} \cdot \underbrace{g(x)}_{\text{degree: } M=2t} \ . \tag{8.22}$$

The K coefficients of $i(x) = i_0 + i_1 \cdot x + \cdots + i_{K-1} \cdot x^{K-1}$ can be used as information. Clearly, $C_{N-j} = N^{-1} \cdot c(x) = N^{-1} \cdot i(x) \cdot g(x) = 0$ for the roots of $g(x)$, $x = z^{-(N-j)}$, $j = 1, ..., M$. The roots of the generator polynomial are hence in line with the zeros in DFT domain. The code, *i.e.*, the set of codewords is the same as with encoding in DFT domain, having obeyed the degree limitation of the polynomial $C(x)$. However, the relation between information and codeword differs. Now, especially, the information is neither contained in the original, nor in the DFT domain. It is a non-systematic encoding.

In our example, we had four uppermost zeros in the DFT domain, which means

$$\begin{aligned} g(x) &= (x - z^1)(x - z^2)(x - z^3)(x - z^4) \\ &= (x - 5)(x - 4)(x - 6)(x - 2) \\ &= (x + 2)(x + 3)(x + 1)(x + 5) \\ &= 2 + 5x + 6x^2 + 4x^3 + x^4 \ . \end{aligned}$$

With the same information $i(x) = 3 + 1 \cdot x$, we obtain the codeword

$$c(x) = i(x) \cdot g(x) = 6 + 3x + 2x^2 + 4x^3 + x^5 \Longleftrightarrow \mathbf{c} = (6, 3, 2, 4, 0, 1) \ ,$$

which is not the same as the one that resulted from the DFT-domain encoding. We also determine the $i(x)$ that corresponds to the codeword that we obtained from the DFT-domain encoding.

$$i(x) = \frac{c(x)}{g(x)} =$$

$$
\begin{array}{rrrrrrl}
(\quad 6x^5 & +5x^4 & +2x^3 & +0x^2 & +1x & +4 &) : (x^4+4x^3+6x^2+5x+2) = 6x+2 \\
-6x^5 & -3x^4 & -1x^3 & -2x^2 & -5x & & \\
\hline
 & 2x^4 & +1x^3 & +5x^2 & +3x & +4 & \\
 & -2x^4 & -1x^3 & -5x^2 & -3x & -4 & \\
\hline
 & 0 & & & & &
\end{array}
$$

In this division, the K coefficients of $i(x)$ do only depend on the K highest coefficients of $c(x)$. The remaining $M = N - K$ coefficients of $c(x)$ have an influence on the remainder of the division, which has to be zero.

8.3.3 Systematic encoding with the generator polynomial

From the previous section, we already know that the code polynomial can be described as a multiple of the generator polynomial.

$$c(x) = u(x) \cdot g(x) \ .$$

We now require to preserve the higher coefficients of the code polynomial $c(x)$ as a systematic part of the encoding. Therefore, we write

$$
\begin{aligned}
c(x) &= c_{N-1}x^{N-1} + \cdots + c_{N-K}x^{N-K} + c_{N-K-1}x^{N-K-1} + \cdots + c_0 \\
&= i_{K-1}x^{N-1} + \cdots + i_0 x^{N-K} + c_{N-K-1}x^{N-K-1} + \cdots + c_0 \qquad (8.23) \\
&= u(x) \cdot g(x) \ . \qquad (8.24)
\end{aligned}
$$

The part of the code polynomial that does not carry information will now be moved to the right side.

$$
\begin{aligned}
c_{N-1}x^{N-1} + \cdots + c_{N-K}x^{N-K} &= u(x) \cdot g(x) - c_{N-K-1}x^{N-K-1} \cdots - c_1 x - c_0 \\
&= u(x) \cdot g(x) + r(x) \qquad (8.25)
\end{aligned}
$$

This means that the remaining part of the codeword $-r(x)$ is equal to the remainder of the division of $c_{N-1}x^{N-1} + \cdots + c_{N-K}x^{N-K}$ by $g(x)$.

Example 8.2 *In the example that we used to illustrate the DFT-domain encoding, we had the correspondence*

$$\mathbf{c} = (4, 1, 0, 2, 5, 6) \qquad \mathbf{C} = (3, 1, 0, 0, 0, 0)$$

Let us use the two rightmost positions in the original domain as information, i.e., *$c_5x^5 + c_4x^4 = 6x^5 + 5x^4$. With the generator polynomial $g(x) = x^4 + 4x^3 + 6x^2 + 5x + 2$, the remainder of the division is computed as follows.*

$$
\begin{array}{rrrrrrl}
(\quad 6x^5 & +5x^4 &) : (x^4+4x^3+6x^2+5x+2) = 6x+2 & & & & \\
-6x^5 & -3x^4 & -1x^3 & -2x^2 & -5x & & \\
\hline
 & 2x^4 & +6x^3 & +5x^2 & +2x & & \\
 & -2x^4 & -1x^3 & -5x^2 & -3x & -4 & \\
\hline
 & & 5x^3 & +0x^2 & +6x & +3 & = r(x)
\end{array}
$$

The parity positions of the codeword are obtained from the remainder $r(x)$ as $c_0 + c_1x + c_2x^2 + c_3x^3 = -r(x) = -(3 + 6x + 0x^2 + 5x^3) = 4 + 1x^1 + 0x^2 + 2x^3$.

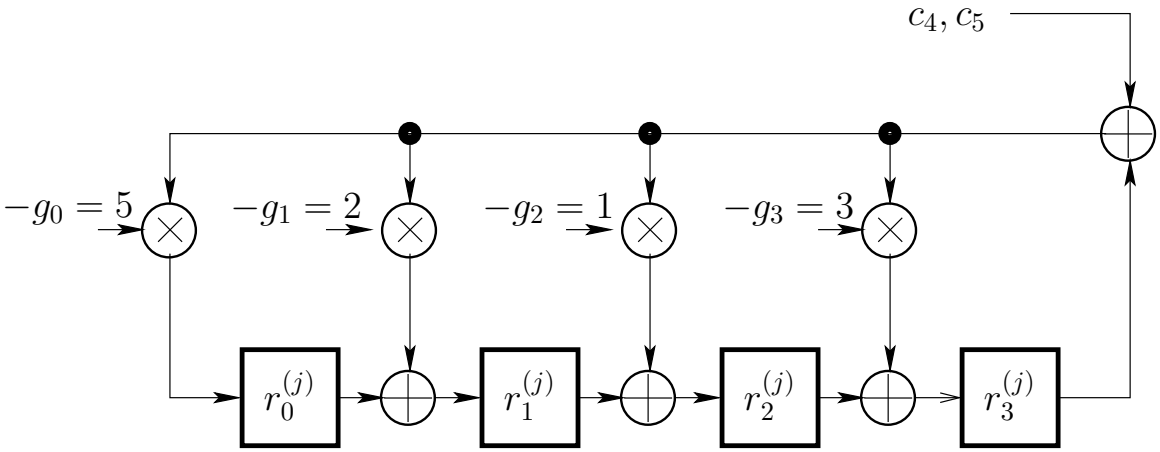


Figure 8.6: Systematic encoder circuit based on the generator polynomial

When we determine the remainder, we actually compute modulo the polynomial $g(x)$. To come to a shift register structure, we formulate the division as an iterative computation mod $g(x)$. This yields a division circuit that we already know from Section 7.2.5. We write the part of $c(x)$ that carries the information as

$$\begin{aligned} c_{N-1}x^{N-1} + c_{N-2}x^{N-2} + \cdots + c_{N-K}x^{N-K} = \\ c_{N-1}x^{N-1} + c_{N-2}x^{N-2} + \cdots + c_M x^M = \\ (\ldots((0x + c_{N-1}x^M)x + c_{N-2}x^M)x + \ldots)x + c_M x^M \ . \end{aligned} \tag{8.26}$$

When starting in the inner brackets with the computation mod $g(x)$, the iteration for the remainder $r(x)$ will become

$$r^{(j)}(x) = x \cdot r^{(j-1)}(x) + c_{N-j}x^M \bmod g(x) \ . \tag{8.27}$$

Figure 8.6 shows the shift register for the example.

The remainders that show up during the iterations are listed in the following table.

j	$r_0^{(j)}$	$r_1^{(j)}$	$r_2^{(j)}$	$r_3^{(j)}$	u_j
0	0	0	0	0	0
1	2	5	6	4	6
2	3	6	0	5	2

The result of the division is not too interesting. It is nevertheless computed and the coefficients will appear behind the (upper right) addition and are also listed in the rightmost column of the table. This column yields $u(x) = u_0x + u_1 = 6x + 2$.

Noting that the parity positions result from the negative remainder, the shift register can be extended such that the complete codeword will be put out. Such an extension is shown in Fig. 8.7.

8.3.4 Systematic encoding with the parity-check polynomial

In (8.12) we provided a recursion formula and subsequently a shift register structure for the case when a product of two polynomials is a multiple of $(x^N - 1)$, *i.e.*, $a(x) \cdot b(x) = 0 \bmod (x^N - 1) = (\ldots) \cdot (x^N - 1)$.

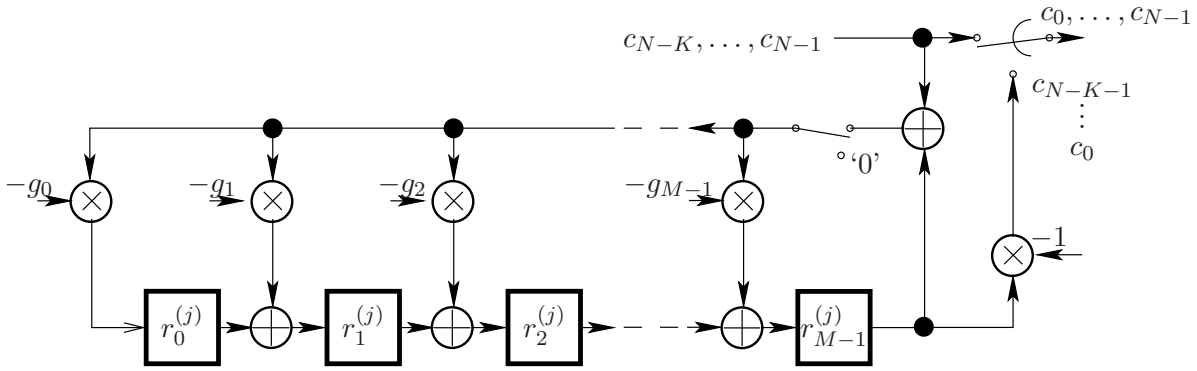


Figure 8.7: Extended systematic encoder circuit based on the generator polynomial

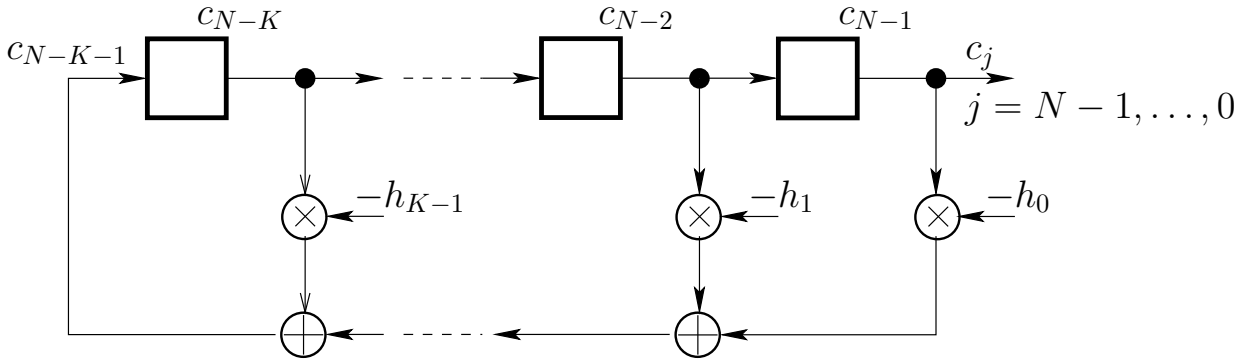


Figure 8.8: Systematic encoding shift register based on the parity-check polynomial

Let $a(x)$ be the code polynomial $c(x) = u(x) \cdot g(x)$, a multiple of $g(x)$. If we define a **parity-check polynomial**

$$h(x) := (x^N - 1)/g(x) = h_0 + h_1 x + \cdots + 1x^K \tag{8.28}$$

(degree $h(x) = N -$ degree $g(x) = N - M = K$),

we obtain

$$c(x) \cdot h(x) = 0 \bmod (x^N - 1) = (\ldots) \cdot (x^N - 1) \; . \tag{8.29}$$

Figure 8.8 shows the shift-register structure.

Example 8.3 *In our example, we had* $N = 6$ *and* $M = 4$ *with* $g(x) = (x - z^1)(x - z^2)(x - z^3)(x - z^4)$. $h(x)$ *follows to be*

$$h(x) = (x^6 - 1)/g(x) = (x - z^5)(x - z^0) = (x - 3)(x - 1) = 3 + 3x + x^2 \; .$$

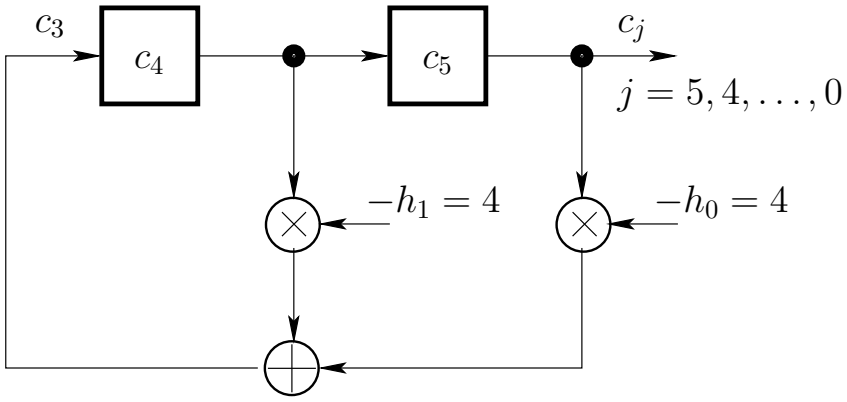


Figure 8.9: Systematic encoder using the parity-check polynomial of Example 8.3

The corresponding encoder circuit is shown in Fig. 8.9.

Note: An encoder on the basis of the generator polynomial in the previous section requires a shift register of length M, whereas the one using the parity-check polynomial needs a register of length K. Depending on the coderate, one or the other method will be preferred. When the number of parity positions is required to be variable, the circuit using the parity-check polynomial should be chosen. RS codes allow to adaptively change the number of parity positions, *i.e.*, one can adaptively transmit more and more parities and always preserve the optimum error-correction properties ($t = \lfloor M/2 \rfloor$).

8.3.5 Generator and parity-check matrices from generator and parity-check polynomials

We have currently described cyclic codes by polynomials. At the beginning, however, we specified linear block codes by their generator and parity-check matrices. We will now provide the connection between both representations. For simplicity, we only do this for the non-systematic encoding with the generator polynomial in Section 8.3.2 described by Eq. (8.22),

$$c(x) = i(x) \cdot g(x) \; .$$

We again arrange the coefficients of the polynomials as vectors in the form

$$\begin{aligned} c(x) &\iff \mathbf{c} = (c_0, c_1, \ldots, c_{N-1}) \\ i(x) &\iff \mathbf{i} = (i_0, i_1, \ldots, i_{K-1}) \; . \end{aligned}$$

The multiplication of two polynomials $i(x) \cdot g(x)$ corresponds to the convolution of their coefficients.

$$c_l = \sum_{j=0}^{K-1} i_j g_{l-j} \tag{8.30}$$

This can be rephrased by a set of linear equations with a Toeplitz matrix.

$$(c_0, c_1, \ldots, c_{N-1}) = (i_0, i_1, \ldots, i_{K-1}) \cdot \mathbf{G}$$

with

$$\mathbf{G} = \begin{pmatrix} g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{N-K} & 0 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{N-K} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{N-K} & 0 & \cdot & \cdot & 0 \\ \vdots & & & & & & & & & & & & & & \vdots \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{N-K} \end{pmatrix} \tag{8.31}$$

In order to determine a corresponding parity-check matrix, we consider the product

$$c(x) \cdot h(x) = i(x) \cdot g(x) \cdot h(x) = i(x)(x^N - 1) = -i(x) + x^N i(x) \; . \tag{8.32}$$

Since the maximum degree of $i(x)$ is $K-1$, the coefficients of the powers K to $N-1$ will be zero. Since the multiplication of polynomials means the convolution of their coefficients, we obtain

$$\sum_{j=0}^{K} h_j c_{l-j} = 0 \quad \text{for } l = K, \ldots, N-1 \; , \tag{8.33}$$

expanding it,

$$\begin{aligned} h_0 c_K + h_1 c_{K-1} + h_2 c_{K-2} + \cdots + h_K c_0 &= 0 \\ h_0 c_{K+1} + h_1 c_K + h_2 c_{K-1} + \cdots + h_K c_1 &= 0 \\ &\vdots \\ h_0 c_{N-1} + h_1 c_{N-2} + h_2 c_{N-3} + \cdots + h_K c_{N-K-1} &= 0 \; . \end{aligned}$$

We obtain the parity-check equations

$$\underbrace{(0, 0, \ldots, 0)}_{N-K} = (c_0, c_1, \ldots, c_{N-1}) \cdot \mathbf{H}^T$$

with

$$\mathbf{H} = \begin{pmatrix} h_K & h_{K-1} & h_{K-2} & \cdot & \cdot & \cdot & \cdot & \cdot & h_0 & 0 & 0 & 0 & \cdot & \cdot 0 & \\ 0 & h_K & h_{K-1} & h_{K-2} & \cdot & \cdot & \cdot & \cdot & \cdot & h_0 & 0 & 0 & \cdot & \cdot 0 & \\ 0 & 0 & h_K & h_{K-1} & h_{K-2} & \cdot & \cdot & \cdot & \cdot & \cdot & h_0 & 0 & \cdot & \cdot & 0 \\ \vdots & & & & & & & & & & & & & & \vdots \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & h_K & h_{K-1} & h_{K-2} & \cdot & \cdot & \cdot & \cdot & \cdot & h_0 \end{pmatrix} . \tag{8.34}$$

This $\mathbf{H}$-matrix is again the generator matrix $\mathbf{G}^\perp$ of the dual code with the generator polynomial $g^\perp(x) = x^K h(x^{-1})$, the reciprocal parity-check polynomial.

Taking advantage of the Euclidean division algorithm treated later, systematic generator and parity-check matrices can be derived, too. We do not provide this derivation.

Chapter 9

The decoding as an interpolation problem and the derivation of the key equation for error localization

The decoding becomes very simple when the error positions are already given. This case is known as **erasure decoding**. Erased symbols are regenerated from the rest of the codeword. For RS codes, a possible procedure has already been described. The non-erased components determine the code polynomial of degree $\leq K-1$ and replacing the erased components is then again a polynomial interpolation and is thus the same as the encoding. More problematic seems to find the location of errors. Hereto, first an error-indicator, called **syndrome** is required. The syndrome definition will be treated in the next section. The decoding, *i.e.*, the error correction thus consists of three steps:

1. Computation of the syndrome,
2. Finding the error locations,
3. Determining the error values at the error locations and correction of the received word.

9.1 The syndrome as an error indicator

The syndrome is usually defined as follows:

Definition 9.1 *Terms computed from the received word* $\mathbf{r}$ *that are only dependent on the error vector* $\mathbf{f}$ *but not of the transmitted codeword* $\mathbf{c}$, i.e.*, of the submitted information.*

One possibility to define a syndrome would, *e.g.*, be by means of the $\mathbf{H}$-matrix

$$\mathbf{s} = \mathbf{r}\,\mathbf{H}^T = (\mathbf{c}+\mathbf{f})\,\mathbf{H}^T = \mathbf{f}\,\mathbf{H}^T\ . \tag{9.1}$$

This syndrome vector could, *e.g.*, be used to address the entries of a table of error pattern ordered by their syndromes (list decoding).

In the case of RS codes, however, it appears to be more suitable to define the M consecutive positions of the DFT vector as a syndrome that were required to be zero for all valid codewords.

$$\{\mathbf{c}|c_j = C(x = z^j), \text{ degree } C(x) \leq K - 1 = N - M - 1 = N - 2t - 1 = N - d_{Hm}\} \; . \tag{9.2}$$

Then, the syndrome $\mathbf{S}$ follows from

$$\mathbf{r} = \mathbf{c} + \mathbf{f} \;\circ\!\!-\!\!\bullet\; \mathbf{R} = \mathbf{C} + \mathbf{F} = (C_0 + F_0, \ldots, C_{K-1} + F_{K-1}, \underbrace{F_{N-2t}, \ldots, F_{N-1}}_{\mathbf{S}}) \; . \tag{9.3}$$

In polynomial representation this reads

$$R(x) = \underbrace{R_0 + R_1 x + \cdots + R_{K-1} x^{K-1}}_{R^\star(x) \text{ (unknown part of } F(x))} + \underbrace{R_{N-2t} x^{N-2t} + \cdots + R_{N-1} x^{N-1}}_{S(x) \cdot x^{N-2t} \text{ (known part of } F(x))} \tag{9.4}$$

Since the correcting capability of RS codes does not change when cyclically shifting the zeros in the DFT vector, the syndrome can be located at every positions that are cyclically consecutive, *i.e.*, also at the lower end.

$$\mathbf{r} = \mathbf{c} + \mathbf{f} \;\circ\!\!-\!\!\bullet\; \mathbf{R} = \mathbf{C} + \mathbf{F} = (\underbrace{F_0, \ldots, F_{2t-1}}_{\mathbf{S}}, C_{2t} + F_{2t}, \ldots, C_{N-1} + F_{N-1}) \tag{9.5}$$

In polynomial representation this reads

$$R(x) = \underbrace{R_0 + \cdots + R_{2t-1} x^{2t-1}}_{S(x) \text{ (known part of } F(x))} + \underbrace{R_{2t} x^{2t} + \cdots + R_{N-1} x^{N-1}}_{R^\star(x) \cdot x^{2t} \text{ (unknown part of } F(x))} \tag{9.6}$$

Note that the shift property of the DFT says that a cyclic shift in DFT domain corresponds to a multiplication with a factor in time domain, *i.e.*, the error positions stay the same, whatever syndrome position was selected. Nevertheless, the error values are changed!

Example 9.1 *We again consider our example over $GF(7)$ with $N = 6$ and $t = 2$:*

$$\mathbf{c} = (4, 1, 0, 2, 5, 6) \;\circ\!\!-\!\!\bullet\; (3, 1, 0, 0, 0, 0)$$

Let the received vector be

$$\mathbf{r} = (4, 1, 0, 4, 5, 5)$$

We have the correspondence

$$\mathbf{r} = (4, 1, 0, 4, 5, 5) \;\circ\!\!-\!\!\bullet\; (2, 1, \underbrace{2, 1, 0, 5}_{\mathbf{S}})$$

This syndrome follows from the error vector $\mathbf{f} = (0, 0, 0, 2, 0, 6)$.

9.2 Prony's curve-fitting method as a decoding algorithm

In Prony's approach [61], given samples are interpolated by a linear combination of exponential functions. The method is known for centuries (1795), but has been reinvented in the coding theory under the name *Gorenstein-Zierler* algorithm. Wolf [62] realized the equivalence in 1967.

Prony's method uses the following interpolation function:

$$y(x) \;=\; A_0 e^{a_0 x} \;+\; A_1 e^{a_1 x} \;+\cdots+\; A_{n-1} e^{a_{n-1} x}\;. \tag{9.7}$$

Abbreviating $\mu_j^x := e^{a_j x}$, this reads

$$y(x) \;=\; A_0 \mu_0^x \;+\; A_1 \mu_1^x \;+\cdots+\; A_{n-1} \mu_{n-1}^x\,. \tag{9.8}$$

In case of equidistant samples $x = 0, 1, \ldots, N_S - 1$, we obtain the set of equations

$$\begin{aligned}
A_0 \;+\; A_1 \;+\cdots+\; A_{e-1} &= y_0 \\
A_0\mu_0^1 \;+\; A_1\mu_1^1 \;+\cdots+\; A_{e-1}\mu_{e-1}^1 &= y_1 \\
A_0\mu_0^2 \;+\; A_1\mu_1^2 \;+\cdots+\; A_{e-1}\mu_{e-1}^2 &= y_2 \\
&\;\;\vdots \\
A_0\mu_0^e \;+\; A_1\mu_1^e \;+\cdots+\; A_{e-1}\mu_{e-1}^e &= y_e \\
&\;\;\vdots \\
A_0\mu_0^{N_S-1} \;+\; A_1\mu_1^{N_S-1} \;+\cdots+\; A_{e-1}\mu_{e-1}^{N_S-1} &= y_{N_S-1}\;.
\end{aligned} \tag{9.9}$$

With respect to the unknowns $A_0, \ldots, A_{e-1}$, the set of linear equations has a Vandermonde coefficient matrix, as we know it from the DFT/IDFT. Would all the μ_j be known, we would just have to solve the Vandermonde system for the A_j. With respect to the μ_j, however, the set of equation is nonlinear. The solution will be enabled by defining a polynomial

$$\mu^e + \alpha_1\mu^{e-1} + \alpha_2\mu^{e-2} + \cdots + \alpha_{e-1}\mu + \alpha_e \;=\; 0 \tag{9.10}$$

with the roots μ_j. Shortly, we will recognize that this polynomial is equivalent to the so-called *error-locator polynomial* in the decoding of RS codes, whose roots determine the error locations.

If we multiply the first equation from (9.9) with α_e, the second with α_{e-1}, ..., and the $(e+1)$st with 1 and sum up all these equations, we obtain the first equation of the following set of linear equations.

$$\begin{aligned}
y_{e-1}\alpha_1 + y_{e-2}\alpha_2 + \cdots + y_0\alpha_e &= -y_e \\
y_e\alpha_1 + y_{e-1}\alpha_2 + \cdots + y_1\alpha_e &= -y_{e+1} \\
&\;\;\vdots \\
y_{N_S-2}\alpha_1 + y_{N_S-3}\alpha_2 + \cdots + y_{N_S-n-1}\alpha_e &= -y_{N_S-1}
\end{aligned} \tag{9.11}$$

The further equations are obtained by successively starting from the second equation, third, and so forth. This set of equations can directly be solved for $N_S = 2e$ and has a **Toeplitz**[1] structure.

[1] A **Hankel** matrix is an upside-down Toeplitz matrix.

(9.11) can be written as

$$(1, \alpha_1, \alpha_2, \ldots, \alpha_e) \begin{pmatrix} y_e & y_{e+1} & y_{e+2} & & y_{N_S-1} \\ y_{e-1} & y_e & y_{e+1} & \ddots & \\ y_{e-2} & y_{e-1} & y_e & \ddots & \\ & \ddots & \ddots & & \\ y_0 & & & & y_{N_S-e-1} \end{pmatrix} = (0, \ldots, 0) \; . \tag{9.12}$$

For such banded Toeplitz matrices exist efficient solution methods from which we will describe the most important two, the Berlekamp-Massey and the Euclidean division algorithm. In the area of signal processing (*e.g.*, linear prediction, LPC), other algorithms have been introduced, namely the Levinson-Durbin and the Trench algorithms. The Levinson-Durbin algorithm, *e.g.*, cannot be used in the decoding due to problems with singular submatrices. This is actually also known to be a problem in signal processing when submatrices become ill-conditioned. The Berlekamp-Massey and Euclidean algorithms, however, can easily be modified to be used in signal-processing applications.

The set of equations can be solved for $N_S = 2e$. Then the roots are to be determined and finally, the coefficients A_i are derived from the Vandermonde system.

The relation to the decoding problem can be seen at once when writing down the equations of the syndrome components dependent on the error positions and error values. For simplicity, we assume a syndrome at the lower end. We obtain

$$\begin{aligned} f_{l_0} + f_{l_1} + \cdots + f_{l_{e-1}} &= S_0 \\ f_{l_0}(z^{-l_0})^1 + f_{l_1}(z^{-l_1})^1 + \cdots + f_{l_{e-1}}(z^{-l_{e-1}})^1 &= S_1 \\ f_{l_0}(z^{-l_0})^2 + f_{l_1}(z^{-l_1})^2 + \cdots + f_{l_{e-1}}(z^{-l_{e-1}})^2 &= S_2 \\ &\vdots \\ f_{l_0}(z^{-l_0})^{2t-1} + f_{l_1}(z^{-l_1})^{2t-1} + \cdots + f_{l_{e-1}}(z^{-l_{e-1}})^{2t-1} &= S_{2t-1} \; . \end{aligned} \tag{9.13}$$

We observe the correspondences

$$\mu_j \equiv z^{-l_j} \qquad y_j \equiv S_j \; .$$

For $e \leq t$, the set of equations can be uniquely solved. From the solutions $\mu_j = z^{-l_j}$ of the polynomial (9.10), the usage of the polynomial as an error-locator polynomial becomes obvious, since l_j are the error locations.

9.3 Derivation of the key equation

Slightly different from the previous section, the error-locator polynomial is usually defined as

Definition 9.2 *The error-locator polynomial* $\Gamma(x)$ *is defined as*

$$\Gamma(x) = \prod_{j \in \mathbb{F}} (x - z^j) \, ,$$

where $\mathbb{F}$ *is the set of indices of the error positions,* i.e., $(\mathbb{F} = \{j | f_j \neq 0\})$.

Likewise, the non-error locator polynomial is defined as

$$\overline{\Gamma}(x) \;=\; \prod_{j \in \overline{\mathbb{F}}} (x \;-\; z^j)\,,$$

where $\overline{\mathbb{F}}$ *is the set of indices of the error-free positions,* i.e., $(\overline{\mathbb{F}} = \{j | f_j = 0\})$.

Due to $x^N - 1 = \prod_{j=0}^{N-1}(x - z^j)$, we have

$$\Gamma(x) \cdot \overline{\Gamma}(x) = (x^N - 1)\,. \tag{9.14}$$

Since $F(x)$ has zeros at the non-error positions $(j \in \overline{\mathbb{F}} :\; F(z^j) = 0)$, we also have

$$F(x) = T(x) \cdot \overline{\Gamma}(x) = T(x) \cdot \prod_{j \in \overline{\mathbb{F}}} (x \;-\; z^j)\,. \tag{9.15}$$

$F(x)$ is also a multiple of $\overline{\Gamma}(x)$. The polynomial $T(x)$, the so-called *error-evaluator polynomial* will later be described when we will deal with the error values.

From the two previous equations, we obtain a relation, called the **key equation**:

$$\Gamma(x) \cdot F(x) \;=\; \Gamma(x) \cdot T(x) \cdot \overline{\Gamma}(x) \;=\; T(x) \cdot (x^N - 1) \;=\; 0 \mod (x^N - 1)\,. \tag{9.16}$$

The product of the polynomials $\Gamma(x)$ and $F(x)$ corresponds to the cyclic convolution of the coefficient vectors. Let e be the number of errors, we obtain

$$\sum_{l=0}^{N-1} \Gamma_l \cdot F_{j-l \bmod N} \;=\; \sum_{l=0}^{e} \Gamma_l \cdot F_{j-l \bmod N} \;=\; 0 \quad j = 0, \ldots, N-1\,. \tag{9.17}$$

The frequency components due to errors can only be directly read at the positions of the **syndrome S**. The decoding problem can now be formulated such that we have to find the error-locator polynomial with the least number of linear factors (roots) $(x - z^j)$, *i.e.*, of lowest degree that fulfills the key equation (9.16) or (9.17). This is equivalent to searching for an error polynomial $F(x)$, from which we only know the syndrome positions, with as many linear factors $(x - z^j)$ as possible. The underlying assumption is that a smaller number of errors are more probable than a larger number of errors. This is usually the case (see, *e.g.*, the binary symmetric channel). Furthermore, we assume that all codewords appear with the same probability and the error vector does not depend on the transmitted codeword.

In the following, we write the error-locator polynomial in a slightly different form in which the first coefficient is equal to one, *i.e.*,

$$\Gamma(x)|_{neu} = \frac{1}{\prod_{j \in \mathbb{F}} z^j} \cdot \Gamma(x) = 1 + \Gamma_1 x + \Gamma_2 x^2 + \cdots + \Gamma_e x^e\,. \tag{9.18}$$

When starting the decoding, the actual number of errors is unknown. As we said, the natural assumption is that error pattern with lower weight are more probable. Thus, if the actual error

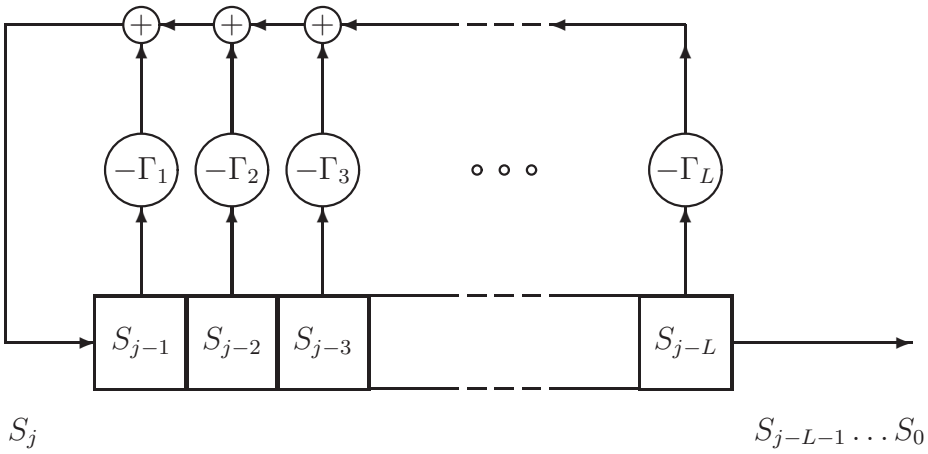


Figure 9.1: Shift-register representation of the key equation

number e is not known, the best we can do is to successively increase the degree of the error-locator polynomial and check, if Eq. (9.17) can be fulfilled. Hence, we work with an error-locator polynomial

$$\Gamma(x) = 1 + \Gamma_1 x + \Gamma_2 x^2 + \cdots + \Gamma_L x^L \tag{9.19}$$

of degree L and check

$$\sum_{l=0}^{L} \Gamma_l \cdot S_{j-l \bmod N} = 0 \ , \quad S_{j-l \bmod N} = F_{j-l \bmod N} \ , \quad j_{S_0} + L \le j \le j_{S_{M-1}} \mod N$$

$$\iff \quad S_j = -\sum_{l=1}^{L} \Gamma_l \cdot S_{j-l \bmod N} \ , \quad j_{S_0} + L \le j \le j_{S_{M-1}} \mod N \tag{9.20}$$

As long as $e \le t = \lfloor M/2 \rfloor$, a unique solution to Eq. (9.20) can be found. This solution can be represented by the linear shift register in Fig. 9.1.

Chapter 10

The Berlekamp-Massey algorithm for solving the key equation

This Toeplitz algorithm was first described in Berlekamp's book [63] and further illustrated as shift-register synthesis by Massey [64]. In here, we will discuss Massey's description in parallel to a matrix-based illustration.

10.1 The key operations of the algorithm

We are searching for a shift-register of shortest length according to Fig. 9.1, *i.e.*, an error-locator polynomial of lowest degree L that fulfills the key equation (9.20). To this end, in the BMA we successively increase the length of the shift-register starting from $L = 0$ until L will be equal to the real error number e (assuming no decoding error).

A change in the coefficients Γ_j or additionally, a length change, will take place, if Eq. (9.20) is not fulfilled or in other words, if the *discrepancy*

$$d_n \;=\; S_n \;+\; \sum_{j=1}^{L_n} \Gamma_j S_{n-j} \tag{10.1}$$

is not equal to zero.

The new shift-register coefficients result from a previous one by adding an earlier coefficient set *before the last length change*. These coefficients will be shifted so far that they again meet the same syndrome positions that they once processed when leading to the discrepancy d_m. Accordingly, the shifted coefficient set yields the same discrepancy d_m that lead to the earlier length change. The recursion formula of the BMA for updating the coefficients of the error-locator polynomial,

$$\Gamma_{n+1}(x) \;=\; \Gamma_n(x) \;-\; \frac{d_n}{d_m} x^{n-m} \Gamma_m(x)\;, \tag{10.2}$$

$$\begin{aligned} n-m \;&:\; \text{shift},\\ d_n \;&:\; \text{current discrepancy},\\ d_m \;&:\; \text{discrepancy before the last length change}, \end{aligned}$$

forces the discrepancy to zero.

A length change is only required, if $d_n \neq 0 \wedge 2L_n \leq n$. $d_n \neq 0 \wedge 2L_n \leq n$ ist.

A further illustration is given in Fig. 10.1, where the coefficient sets are shifted along the syndrome vector. There we see that $n - m$ leading zeros (right side) are added to the earlier coefficient set when the two coefficient sets are linearly combined at the position n to cancel out the current discrepancy. The prepending of the leading zeros is described by the multiplication by x^{n-m}.

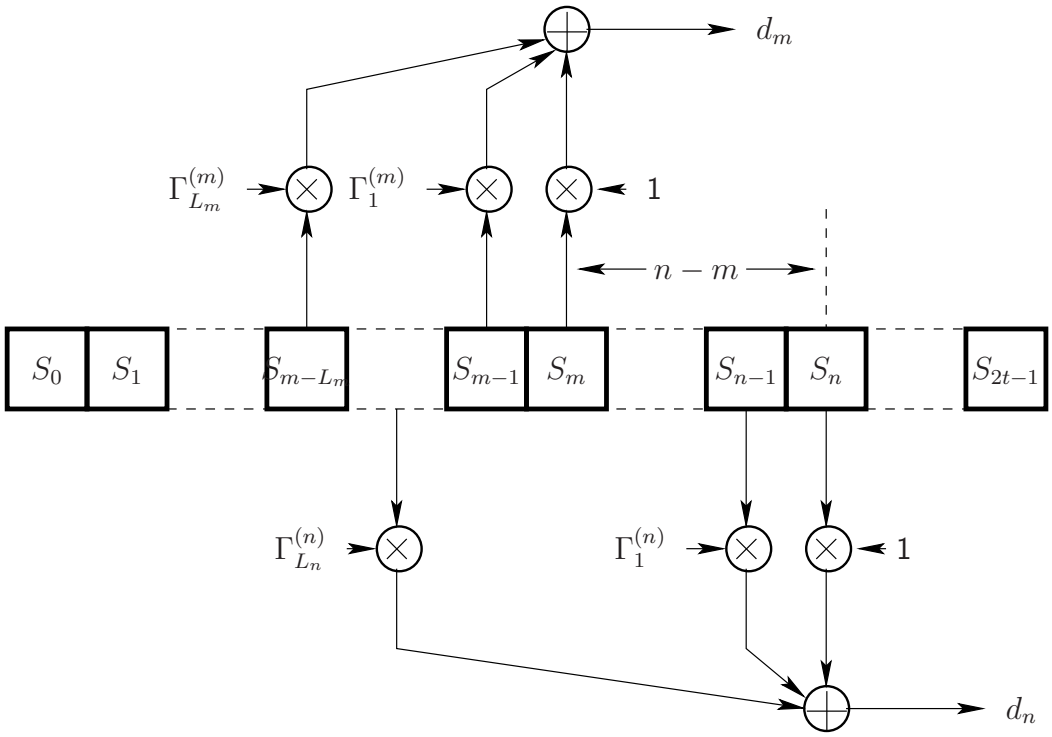


Figure 10.1: Illustration of the recursion formula of the Berlekamp-Massey algorithm

The principal structure of the BMA will become more obvious when we look at the iteratively solved set of linear equations [65]. This leads to an alternative description to the more common shift-register synthesis that was once provided by Massey [64].

The key equation was

$$S_j \;=\; -\sum_{l=1}^{e} \Gamma_l S_{j-l} \;, \tag{10.3}$$

which corresponds to the set of linear equations

$$(1, \Gamma_1, \Gamma_2, \ldots, \Gamma_e) \begin{pmatrix} S_e & S_{e+1} & \cdot & \cdot & S_{2e} & \cdot & S_{M-1} \\ S_{e-1} & S_e & \ddots & & \cdot & & \cdot \\ \cdot & S_{e-1} & \ddots & \ddots & \cdot & & \cdot \\ \cdot & & \ddots & \ddots & S_{e+1} & & \cdot \\ S_0 & \cdot & \cdot & S_{e-1} & S_e & \cdot & S_{M-1-e} \end{pmatrix} = (0, \ldots, 0) \tag{10.4}$$

with a Toeplitz coefficient matrix.

In the following, we describe the BMA as a logical consequence of the matrix structure and the requirement to find the error-locator polynomial of lowest possible degree. Note that this

requirement results from the assumption that the probability of error pattern decreases with their weight.

The algorithm starts from $L = 0$ using the smallest possible sub-matrix

$$(1) \cdot (S_0) \;=\; (S_0) \;=:\; (d_0^I)\,. \tag{10.5}$$

The first discrepancy d_0^I is thus equal to S_0. Even if the discrepancy would be zero, it would be necessary to check the next bigger sub-system.

Since we are searching for the error-locator polynomial of lowest degree, it is natural to extend the vector $\mathbf{\Gamma}$ by append a zero to the right to investigate the next bigger sub-system. This does not yet increase the degree of the corresponding error-locator polynomial. We obtain the set of equations

$$(1,0) \begin{pmatrix} S_1 & S_2 \\ S_0 & S_1 \end{pmatrix} \;=\; (d_0^{II}, d_1^{II})\,. \tag{10.6}$$

We are looking for a solution leading to a zero right side. If d_0^{II} and d_1^{II} are not zero, the first position can be forced to zero, if also d_0^I was unequal zero.

If we use a vector $(0,1)$ together with a 2×2 matrix instead of (1) in Eq. (10.5), *i.e.*, prepending a zero on the left, the first component on the right side will again be the earlier discrepancy $d_0^I = S_0$.

By linearly combining the two left and right zero-extended vectors, it is now possible to force the first component on the right side to zero.

$$[(1,0) \;-\; \frac{d_0^{II}}{d_0^I}(0,1)] \cdot \begin{pmatrix} S_1 & S_2 \\ S_0 & S_1 \end{pmatrix} \;=\; (0, d_1^{III}) \tag{10.7}$$

In case $d_1^{III} \neq 0$, the sub-matrix is non-singular ($\det \mathbf{S}_1 \neq 0$) and a length change together with using the next bigger sub-matrix is required. However, even if d_1^{III} should be zero, the next bigger sub-matrix would have to be checked if it is singular, too.

At this point, we like to mention that a length change in the sense of the usual BMA description [64] is performed, when the discrepancy was zero and the next operation extends the solution vector with non-zero components.

After the first steps of the algorithm and thus its initialization have been presented, we will now point out the main operations that rely on the Toeplitz structure.

It has already become visible that the extension of the solution vector with zeros on the left or right sides, together with considering the corresponding bigger Toeplitz sub-matrix represent the kernel of the algorithm. In the following we study the zero-extensions, *i.e.*, its impact on the right side in some more detail.

Appending zeros on the right:

Extending the vector $\mathbf{\Gamma}$ by appending zeros on the right leads to a left shift of the right side by one position and two new components will show up on the right side.

$$(1, \Gamma_1, \Gamma_2, \ldots, \Gamma_l) \cdot \begin{pmatrix} S_l & & S_{2l} \\ & \ddots & \\ S_0 & & S_l \end{pmatrix} = (\rho_0, \rho_1, \rho_2, \ldots, \rho_l) \tag{10.8}$$

$$(1, \Gamma_1, \Gamma_2, \ldots, \Gamma_l, 0) \cdot \begin{pmatrix} S_{l+1} & & S_{2(l+1)} \\ & \ddots & \\ S_0 & & S_{l+1} \end{pmatrix} = (\rho_1, \rho_2, \ldots, \rho_l, \rho_{l+1}, \rho_{l+2}) \tag{10.9}$$

Prepending zeros on the left:

Extending the vector $\mathbf{\Gamma}$ by prepending zeros on the left preserves the right side and just adds one new component at the rightmost location.

$$(1, \Gamma_1, \Gamma_2, \ldots, \Gamma_l) \cdot \begin{pmatrix} S_l & & S_{2l} \\ & \ddots & \\ S_0 & & S_l \end{pmatrix} = (\rho_0, \rho_1, \rho_2, \ldots, \rho_l) \tag{10.10}$$

$$(0, 1, \Gamma_1, \Gamma_2, \ldots, \Gamma_l) \cdot \begin{pmatrix} S_{l+1} & & S_{2(l+1)} \\ & \ddots & \\ S_0 & & S_{l+1} \end{pmatrix} = (\rho_0, \rho_1, \rho_2, \ldots, \rho_l, \rho_{l+1}) \tag{10.11}$$

We use the property that an earlier solution vector (*before the last length change*) leads to zeros on the right side down from a certain component on. A prepending of zeros to the left preserves this right side except for new components on the right. Let the resulting vector be $\mathbf{\Gamma_v}$.

The current solution vector $\mathbf{\Gamma}$ will now be appended with zeros on the right end, shifting the right side of the set of equations to the left, until the first non-zero component exactly matches the position of the first non-zero component when using $\mathbf{\Gamma_v}$.

Both set of equations will then have the structure

$$(0, \ldots, 0, 1, \Gamma_{v1}, \Gamma_{v2}, \ldots) \cdot (\,\cdot\,) = (\underbrace{0, \ldots, 0}, d_{vj}, d_{vj+1}, d_{vj+2}, \ldots) \tag{10.12}$$

$$(1, \Gamma_1, \Gamma_2, \ldots, 0, \ldots, 0) \cdot (\,\cdot\,) = (\overbrace{0, \ldots, 0}, d_j, d_{j+1}, d_{j+2}, \ldots) \tag{10.13}$$

With a linear combination of both solution vectors, the first non-zero component d_j is forced to zero. The new solution vector will then be

$$\mathbf{\Gamma} := (1, \Gamma_1, \Gamma_2, \ldots, 0, \ldots, 0) - \underbrace{\frac{d_j}{d_{vj}}}_{=d_n/d_m} (0, \ldots, 0, 1, \Gamma_{v1}, \Gamma_{v2}, \ldots) \tag{10.14}$$

This is equivalent to the recursion (10.2).

The first non-zero component on the right side of the set of equations *before the last length change* acts like a marking that needs to be reached by the current solution vector by rightsided appending of zeros.

In the following, we illustrate the algorithm with two examples. The first one shows the basic procedure more clearly, whereas Example 10.2 treats the special case of singular sub-matrices. There, we will observe that it may even be necessary to extend the syndrome matrix by unknown components to proceed in the algorithm. These will however not be used in the following steps.

Both examples use an odd number of syndrome components, which may first appear to be unusual, since the correcting capability of an RS code is $t = \lfloor M/2 \rfloor$. When looking more carefully, one will realize that the last syndrome component is actually not required. It can only be used for an additional check, if all errors have been found.

From the example, we also observe that a length change in the sense of the BMA will only occur, when $d_n \neq 0 \wedge 2L_n \leq n$.

The last step of the second example shows the final result together with the original 4×4 matrix. The BMA solution, however, was already found in the previous step. This last step was only provided for illustration purposes. Note that it does not result from just eliminating the last row (and last column) of the 5×5 matrix which would correspond to the zero in the solution vector.

Example 10.1

GF(7), primitive element: 5, syndrome length: 5

Error vector: $(1,0,0,0,1,0)$
Syndrome: $(5,2,4,5,2)$

$$(1,\Gamma_1,\Gamma_2)\begin{pmatrix}4&5&2\\2&4&5\\5&2&4\end{pmatrix}\stackrel{!}{=}(0,0,0)$$

$$(1)\cdot(5)=(5)$$

$$(1,0)\begin{pmatrix}2&4\\5&2\end{pmatrix}=(2,4)$$

$$(1,0)-\tfrac{2}{5}(0,1)=(1,1)$$

$$(1,1)\begin{pmatrix}2&4\\5&2\end{pmatrix}=(0,6)$$

$$(1,1,0)\begin{pmatrix}4&5&2\\2&4&5\\5&2&4\end{pmatrix}=(6,2,0)$$

$$(1,1,0)-\tfrac{6}{5}(0,0,1)=(1,1,3)$$

$$(1,1,3)\begin{pmatrix}4&5&2\\2&4&5\\5&2&4\end{pmatrix}=(0,1,5)$$

$$(1,1,3)-\tfrac{1}{6}(0,1,1)=(1,2,4)$$

$$(1,2,4)\begin{pmatrix}4&5&2\\2&4&5\\5&2&4\end{pmatrix}=(0,0,0)$$

$$\Gamma(x)=1+2x+4x^2$$

$$\begin{aligned}
\Gamma(5^0) &= 1+2\cdot1+4\cdot1=0 \longleftarrow\\
\Gamma(5^1) &= 1+2\cdot5+4\cdot4=6\\
\Gamma(5^2) &= 1+2\cdot4+4\cdot2=3\\
\Gamma(5^3) &= 1+2\cdot6+4\cdot1=3\\
\Gamma(5^4) &= 1+2\cdot2+4\cdot4=0 \longleftarrow\\
\Gamma(5^5) &= 1+2\cdot3+4\cdot2=1
\end{aligned}$$

Error positions: 0 *and* 4

Example 10.2

GF(11), primitive element: 6, syndrome length: 7

Error vector: $(1,0,1,0,4,0,0,0,0,0)$
Syndrome: $(5,8,4,7,4,5,8)$

$$(1,\Gamma_1,\Gamma_2,\Gamma_3)\begin{pmatrix}7&4&5&8\\4&7&4&5\\8&4&7&4\\5&8&4&7\end{pmatrix}\stackrel{!}{=}(0,0,0,0)$$

$$(1)\cdot(5)=(5)$$

$$(1,0)\begin{pmatrix}8&4\\5&8\end{pmatrix}=(8,4)$$

$$(1,0)-\tfrac{8}{5}(0,1)=(1,5)$$

$$(1,5)\begin{pmatrix}8&4\\5&8\end{pmatrix}=(0,0)$$

$$(1,5,0)\begin{pmatrix}4&7&4\\8&4&7\\5&8&4\end{pmatrix}=(0,5,6)$$

$$(1,5,0,0)\begin{pmatrix}7&4&5&8\\4&7&4&5\\8&4&7&4\\5&8&4&7\end{pmatrix}=(5,6,3,0)$$

$$(1,5,0,0)-\tfrac{5}{5}(0,0,0,1)=(1,5,0,10)$$

$$(1,5,0,10)\begin{pmatrix}7&4&5&8\\4&7&4&5\\8&4&7&4\\5&8&4&7\end{pmatrix}=(0,9,10,4)$$

$$(1,5,0,10)-\tfrac{9}{5}(0,1,5,0)=(1,1,2,10)$$

$$(1,1,2,10)\begin{pmatrix}7&4&5&8\\4&7&4&5\\8&4&7&4\\5&8&4&7\end{pmatrix}=(0,0,8,3)$$

$$(1,1,2,10,0)\begin{pmatrix}4&5&8&?&?\\7&4&5&8&?\\4&7&4&5&8\\8&4&7&4&5\\5&8&4&7&4\end{pmatrix}=(0,8,3,?,?)$$

$$(1,1,2,10,0) - \tfrac{8}{5}(0,0,1,5,0) = (1,1,7,2,0)$$

$$(1,1,7,2,0)\begin{pmatrix} 4 & 5 & 8 & ? & ? \\ 7 & 4 & 5 & 8 & ? \\ 4 & 7 & 4 & 5 & 8 \\ 8 & 4 & 7 & 4 & 5 \\ 5 & 8 & 4 & 7 & 4 \end{pmatrix} = (0,0,0,?,?)$$

$$(1,1,7,2)\begin{pmatrix} 7 & 4 & 5 & 8 \\ 4 & 7 & 4 & 5 \\ 8 & 4 & 7 & 4 \\ 5 & 8 & 4 & 7 \end{pmatrix} = (0,0,0,0)$$

$$\Gamma(x) = 1 + x + 7x^2 + 2x^3$$

$$\begin{aligned}
\Gamma(6^0) &= \Gamma(1) = 0 \longleftarrow \\
\Gamma(6^1) &= \Gamma(6) = 9 \\
\Gamma(6^2) &= \Gamma(3) = 0 \longleftarrow \\
\Gamma(6^3) &= \Gamma(7) = 3 \\
\Gamma(6^4) &= \Gamma(9) = 0 \longleftarrow \\
\Gamma(6^5) &= \Gamma(10) = 5 \\
\Gamma(6^6) &= \Gamma(5) = 10 \\
\Gamma(6^7) &= \Gamma(8) = 7 \\
\Gamma(6^8) &= \Gamma(4) = 3 \\
\Gamma(6^9) &= \Gamma(2) = 3
\end{aligned}$$

Error positions: 0, 2, *and* 4

10.2 Massey's description of the Berlekamp algorithm

As was mentioned, the algorithm was originally described by Berlekamp in [63] and later illustrated as shift-register synthesis by Massey in [64]. In Massey's paper, all main proofs are given, especially the one showing that successively always the shift register of shortest length is found. Also the exact step size for length changes is derived there. In the following we provide the algorithm in Massey's formulation, which may directly be used for programming.

$$
\begin{array}{ll}
1) & 1 \to \Gamma(x) \quad 1 \to \Gamma_v(x) \quad 1 \to \tau \\
 & 0 \to L \quad 1 \to d_m \quad 0 \to n \\
2) & \text{IF } n = M-1, \text{ STOP. OTHERWISE COMPUTE} \\
 & d_n \;=\; S_n \;+\; \sum_{i=1}^{L} \Gamma_i S_{n-i}\,. \\
3) & \text{IF } d_n = 0, \text{ THEN } \tau + 1 \to \tau, \text{ GO TO 6).} \\
4) & \text{IF } d_n \neq 0 \text{ AND } 2L > n, \text{ THEN} \\
 & \Gamma(x) - d_n d_m^{-1} x^{\tau} \Gamma_v(x) \to \Gamma(x) \\
 & \tau + 1 \to \tau \\
 & \text{GO TO 6).} \\
5) & \text{IF } d_n \neq 0 \text{ AND } 2L \leq n, \text{ THEN} \\
 & \Gamma(x) \to \Lambda(x) \\
 & \Gamma(x) - d_n d_m^{-1} x^{\tau} \Gamma_v(x) \to \Gamma(x) \\
 & n + 1 - L \to L \\
 & \Lambda(x) \to \Gamma_v(x) \\
 & d_n \to d_m \\
 & 1 \to \tau. \\
6) & n + 1 \to n \text{ RETURN TO 2).}
\end{array}
\tag{10.15}
$$

(τ corresponds to the shift $n - m$.)

It is assumed that the operations become clear from the previous sections. In the following few paragraphs, some remarks for a deeper understanding of the BMA are provided.

10.3 Error and erasure decoding in the BMA

As we already know, erasures can be seen as errors at known positions. In analogy to the error-locator polynomial, we define an erasure polynomial

$$\Lambda(x) \;=\; \prod_{j \in \mathbb{E}} (x - z^j)\;, \tag{10.16}$$

where $\mathbb{E}$ denotes the index set of the erasure positions ($|\mathbb{E}| = a_E$).

With the matrix description, it is obvious that the algorithm should now start with the subset of equations

$$(1, \Lambda_1, \Lambda_2, \ldots, \Lambda_{a_E}) \cdot \begin{pmatrix} S_{a_E} & & S_{2a_E} \\ & \ddots & \\ S_0 & & S_{a_E} \end{pmatrix} \;=\; (\rho_0, \rho_1, \rho_2, \ldots, \rho_{a_E})\;, \tag{10.17}$$

i.e., the vector $\mathbf{\Gamma}$ will be initialized by $\mathbf{\Lambda}$. Then, the appending ($\rightarrow \Gamma(x)$) and prepending ($\rightarrow \Gamma_v(x)$) of zeros follows. The roots of the erasure polynomial $\Lambda(x)$ will be preserved when initializing the BMA with it. $\Lambda(x)$ will be a factor both in $\Gamma(x)$ and in $\Gamma_v(x)$, meaning that this factor will be preserved in all linear combinations of the two polynomials.

The first usable discrepancy in the above set of equations is ρ_0. Hence, the syndrome length that is left for determining the remaining errors is $M - a_E$. The number of correctable additional errors will then be $\left\lfloor \frac{M-a_E}{2} \right\rfloor$.