Coding Theory

Prof. Dr.-Ing. Thomas Kürner
Parts of this script are based on the manuscript of the lecture of the same title held by Prof. Dr.-Ing. Andreas Czylwik at Braunschweig Technical University in the winter semester 2001/2002.

The version on hand was drawn up by Dipl.-Ing. Andreas Hecker in the years 2004 and 2005.
Organisation

- Lecture and practical exercise in the winter semester (2 + 1 SWS):
  - on Tuesdays; 08.50 am – 09.35 am; lecture hall SN 23.2
  - on Wednesdays; 11.30 am – 01.00 pm; lecture hall SN 23.2
- Assistant: Dipl.-Ing. M. Sc. Thomas Jansen
- Examination: written or oral
Objectives and Contents of the Lecture

Objectives:
This lecture is intended to develop the understanding of the information-theoretical limiting factors in data transmission and to build knowledge of methods of source and error control coding in theory and in practical applications.

Contents:
Introduction
1. Fundamentals of information theory
2. Fundamentals of error control coding
3. Single-error correcting block codes
4. Burst error correcting block codes
5. Convolutional codes
6. Special coding techniques
Literature

Literature recommended for the lecture:
- H. Rohling: Einführung in die Informations- und Codierungstheorie, Teubner

Further literature:
- B. Friedrichs: Kanalcodierung, Springer
- H. Schneider-Obermann: Kanalcodierung, Vieweg
- M. Bossert: Kanalcodierung, Teubner
- J. Bierbrauer: Introduction to Coding Theory, Chapman & Hall/CRC
- O. Mildenberger: Informationstheorie und Codierung, Vieweg
- S. Lin, D. J. Costello Jr.: Error Control Coding, Pearson Prentice Hall

Literature dealing with digital transmission:
- J.G. Proakis, M. Saleki: Grundlagen der Kommunikationstechnik, Pearson Studium
**Introduction**

**Code / Coding:**

A rule for the *unambiguous assignment* of characters of one character set to those of another character set is called a code.

Example: The assignment of names of towns to number plates:
- Berlin → B
- Braunschweig → BS

The conversion of one character set into the other is referred to as *coding* (to code).

Depending on the problem, coding is classified as follows:

- Cryptology
- Source coding
- Error control coding  

}  

subject of this lecture
Coding for encryption (cryptology):
Messages are to be protected against unauthorised monitoring. Decryption can only be effected if the code is known.

Source coding:
Messages are compressed to a minimum number of characters without loss of information (reduction of redundancy). With further compression, loss of information is accepted (e. g. considering video and audio transmission).

Error control coding:
Messages are to be protected against transmission errors by controlled adding of redundancy. Applications can be found, among other things, in the field of data transfer over waveguides and radio channels (especially mobile radio channels) to be secured or with the storage of data.
Block diagram of a digital broadcasting system:

Digital source

Source \[\overset{A}{\longrightarrow}\] \[\overset{D}{\longrightarrow}\] Source Coder \[\overset{\text{Error Control Coder}}{\longrightarrow}\] Modulator

Transmission Channel \[\overset{\text{Interference}}{\longrightarrow}\] Discrete channel

Digital sink

Sink \[\overset{D}{\longrightarrow}\] \[\overset{A}{\longrightarrow}\] Source Decoder \[\overset{\text{Error Control Decoder}}{\longrightarrow}\] Demodulator
Error control coding for error correction
FEC – forward error correction

Simplified model:

Data Source → Channel Coder → Channel → Channel Decoder (Error Correction) → Data Sink

The channel quality is characterised by the residual-error probability after decoding. The data rate, however, is independent from the channel quality.
Overview of some FEC codes:

FEC codes

Block codes
- non-linear
- linear
- non-cyclic
- cyclic
  - Hamming
  - Reed-Muller
  - Reed-Solomon (RS)
  - BCH
  - cyclic Hamming
  - Abrahamson

Convolutional codes
- recursive
- non-recursive
- serial

Concatenated codes
- Product codes
- parallel

The grey-coloured codes will not be dealt with in this lecture.
Error control coding for error detection
CRC – cyclic redundancy check;
Application in combination with ARQ methods (automatic repeat request)

ARQ Control → Channel Coder → Channel → Channel Decoder (Error Detection) → ARQ Control

return channel

For this method, a return channel is required, that is why it is not applicable for “broadcast“ or time-critical applications. Redundancy is added adaptively, that means only in case of errors. Thus, the residual-error probability is independent, however, the data rate depends on the channel quality.

ARQ, for example, is applied for GPRS.
Using error control coding, the error probability can be reduced to any value, if the data rate is smaller than the channel capacity.

Claude E. Shannon, Warren Weaver,  
*The mathematical theory of communication*, Urbana, University of Illinois Press, 1949

Unfortunately, Shannon, the founder of the information theory, does not give a practical construction rule for his theorem.
Channel interference effects and examples of correction results:

Images after transmission:

via BSC channel with $p_{err} = 0.005$
without error control coding

via BSC channel with $p_{err} = 0.5$
without error control coding, but also not corrigible!

via BSC channel with $p_{err} = 0.005$
with (7,4,3)-Hamming error control coding
(not all errors were fixed)

via BSC channel with $p_{err} = 0.005$
with (31,16,15)-BCH error control coding
(almost error-free)
Fundamental idea of error control coding

For the protection of the message to be transmitted, redundancy is added for error detection (FE) and possibly for error correction (FK).

Assignment in the coder; block coding (BC) is given as an example:

\[
\begin{array}{c}
\text{length } k \\
\mid \mid \mid \mid \mid \mid \mid \mid \mid \\
\text{input vector } \mathbf{u} \\
\end{array}
\rightarrow
\begin{array}{c}
\text{length } n \\
\mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mid \mi...
Introduction

Code dice with $n = 3$ and $k = 3$

- $R_C = 1$
- $d_{\text{min}} = 1$
- no FE and no FK possible

- $R_C = 2/3$
- $d_{\text{min}} = 2$
- one error noticeable; no error corrigible

- $R_C = 1/3$
- $d_{\text{min}} = 3$
- two errors noticeable; one error corrigible
1 Fundamentals of Information Theory

In information theory, broadcasting is described mathematically.

Pivotal questions deal with the
- quantitative calculation of the information content of messages,
- determination of the transmission capacity of transmission channels and
- analysis and optimisation of source and error control coding.

Message from the point of view of the information theory:
1.2 Mathematical Description of Messages

Messages are classified according to their quality. The less a message is predictable, the higher its relevance.

Example:

- Tomorrow the sun rises.
- Tomorrow the weather will be bad.
- For tomorrow we expect a severe storm and, as a consequence, a power cut.

The message from a digital source corresponds to a sequence of characters.
Maximum entropy $H_0$ of a source:

It is assumed that a source with a character set of $N$ characters is given. $H_0$ is then defined as:

$$H_0 = \log(N) \text{ bit/character} \quad (\text{bit} = \text{binary digit}) \quad (1.13)$$

$H_0$ describes the number of binary decisions required for the selection of the message. Thereby, the occurrence probability of the source symbols is not considered.

Examples:

Text in German with 71 alphanumerical characters as source
(26-2 characters, 3-2 umlaut, ß, 12 special characters incl. space characters „ “ ()-,.;:!?)

$$H_0 = \log(71) = \ln(71)/\ln(2) = 6.15 \text{ bit/character}$$

One page of text in German with 40 lines and 70 characters per line as source, that means all in all $N = 71^{40 \cdot 70}$ different pages are possible

$$H_0 = \log(71^{40 \cdot 70}) = 40 \cdot 70 \cdot \log(71) = 17.22 \text{ kbit/page}$$
1.2 Mathematical Description of Messages

Information content $I$:

It is assumed that a character set (alphabet) $X$ with $X = \{x_1, \ldots, x_N\}$ and the single occurrence probabilities of the characters $p(x_1), \ldots, p(x_N)$ is given. In this regard, $I$ is to map the content of a message in dependency on the probability $p$ [$I = f(p)$]. For this, the following characteristics are to be fulfilled:

- $I(x_i) \geq 0$ for $0 \leq p(x_i) \leq 1$
- $I(x_i) \to 0$ for $p(x_i) \to 1$
- $I(x_i) > I(x_j)$ for $p(x_i) < p(x_j)$
- $I(x_i, x_j) = I(x_i) + I(x_j)$ for $p(x_i, x_j) = p(x_i) \cdot p(x_j)$, d. h.
  for two consecutive, statistically independent characters $x_i$ and $x_j$

These conditions are fulfilled by the general solution $I(x_i) = -k \cdot \log_b(p(x_i))$.

Definition of information content:

$\begin{align*}
I(x_i) &= \log\left(\frac{1}{p(x_i)}\right) \text{ bit/character} \\
\text{(with } k = 1 \text{ and } b = 2) 
\end{align*}$ (1.14)
1.2 Mathematical Description of Messages

Entropy $H(X)$:

$H(X)$ describes the mean information content of a source:

$$H(X) = \langle I(x_i) \rangle = \sum_{i=1}^{N} p(x_i) \cdot I(x_i)$$

$$= \sum_{i=1}^{N} p(x_i) \cdot \text{ld} \left( \frac{1}{p(x_i)} \right) \text{bit/character}$$

(1.15)

Redundancy of a source $R_Q$:

Information content of a source that is under-utilised due to the particular occurrence probabilities of the characters:

$$R_Q = H_0 - H(X)$$

(1.16)
1.2 Mathematical Description of Messages

Entropy of a binary source:
A binary source has a character set $X$ of exactly two characters:
$X = \{x_1, x_2\}$ with the occurrence probabilities $p(x_1) = p$ and $p(x_2) = 1 - p$.
Then the entropy $H(X)$ is: $H(X) = -p \log(p) - (1 - p) \log(1 - p)$.  
(1.17)
This function in dependence of $p$ is also called Shannon function:

$$S(p) = H(X).$$  
(1.18)

Repetition of mathematics:

$$\log\left(\frac{1}{x}\right) = -\log(x)$$

$$\lim_{x \to 0} x \cdot \log(x) = 0$$
The entropy becomes maximum for equiprobable characters, that means:

\[ H(X) \leq H_0 = \text{ld} N. \]  \hspace{1cm} (1.19)

Evidence:

\[
H(X) - \text{ld} N = \sum_{i=1}^{N} p(x_i) \cdot \text{ld} \left( \frac{1}{p(x_i)} \right) - \sum_{i=1}^{N} p(x_i) \cdot \text{ld} N = \sum_{i=1}^{N} p(x_i) \cdot \text{ld} \left( \frac{1}{p(x_i) \cdot N} \right)
\]

Utilisation of the inequation:

\[ \ln x \leq x - 1 \]  \hspace{1cm} (1.20)

Using \( \ln x \leq x - 1 \Rightarrow \text{ld} x \leq \frac{1}{\ln 2} (x - 1) \)

\[
H(X) - \text{ld} N \leq \sum_{i=1}^{N} p(x_i) \cdot \frac{1}{\ln 2} \left( \frac{1}{p(x_i) \cdot N} - 1 \right)
\]

\[
= \frac{1}{\ln 2} \sum_{i=1}^{N} \left( \frac{1}{N} - p(x_i) \right) = \frac{1}{\ln 2} \left( N \cdot \frac{1}{N} - 1 \right) = 0 \quad \text{q.e.d.}
\]
1.3 Methods and Algorithms of Source Coding

In communications technology, source coding is deployed in order to reduce the data volume to be transmitted. Examples for this can be found in many applications frequently used such as data compression by „zapping“, MP3 or MPEG compression. It can also be found in GSM speech coding where a digitalisation has to be carried out first.

In the following, drafts for binary coding of discrete sources are considered in terms of the transmission via a binary channel.
1.3.1 Fields of Application of Source Coding

The character set \( X = \{x_1, \ldots, x_N\} \) of an arbitrary source is given. With source coding, a binary code of the code word length \( L(x_i) \) is assigned to a character \( x_i \) of the character set \( X \). Regarding this, the objective of the coding is the minimisation of the mean code word length \( \bar{L} \):

\[
\bar{L} = \langle L(x_i) \rangle = \sum_{i=1}^{N} p(x_i) \cdot L(x_i)
\]  

(1.21)

Examples for binary coding of characters:

- **ASCII code:**
  
  Fixed code word length \( L(x_i) = 8 \) (block code)

- **Morse code:**
  
  Letters are represented by dots and dashes; code words are separated by gaps; letters that often occur are assigned to short code words.
Prefix characteristic of a code:
This characteristic means that a code word cannot form the beginning of another code word at the same time. A given bit sequence is then definite without use of further separation rules („point-free code“).

Examples for codes with and without prefix characteristic:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

The definite decoding of a bit sequence is guaranteed. Decoding of the sequence 010010110111100, for example, provides definitely: $x_1 x_2 x_1 x_2 x_3 x_4 x_2 x_1$.

For a successful decoding, however, the synchronisation to the beginning of the sequence is required.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>01</td>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>

The definite decoding of a bit sequence is not possible. For example, for the sequence 010010 the following decoding results are possible: $x_1 x_3 x_2 x_1$, $x_2 x_1 x_3$, $x_1 x_3 x_1 x_3$, $x_2 x_1 x_2 x_1$, $x_3 x_3$. 
Codes with a prefix characteristic can be decoded using a decision tree.

\[
\begin{array}{c|c}
 x_1 & 0 \\
 x_2 & 10 \\
 x_3 & 110 \\
 x_4 & 111 \\
\end{array}
\]

1.3.1 Fields of Application of Source Coding

Redundancy of a code \( R_C \):

\[
R_C = \bar{L} - H(X) \text{ (Do not mistake this for the redundancy } R_Q \text{ of a source!)}
\]
Kraft’s inequation:
A binary code with prefix characteristic and the code word lengths $L(x_1), L(x_2), \ldots, L(x_N)$ only exists if:

$$\sum_{i=1}^{N} 2^{-L(x_i)} \leq 1$$  \hfill (1.23)

Possible proof:
The length of a tree structure is equal to the maximum code word length $L_{\text{max}}$:

$$L_{\text{max}} = \max(L(x_1), L(x_2), \ldots, L(x_N))$$

The code word $x_i$ on the layer $L(x_i)$ eliminates $2^{L_{\text{max}} - L(x_i)}$ of the possible code words on the layer $L_{\text{max}}$.
The sum of all eliminated code words is less than or equal to the maximum number of the code words on the layer $L_{\text{max}}$:

$$\sum_{i=1}^{N} 2^{L_{\text{max}} - L(x_i)} \leq 2^{L_{\text{max}}}$$

The equals sign in Kraft’s inequation is valid, if all endpoints of the code tree are taken by code words.
Lower limit of the mean code word length $\bar{L}$:

Every prefix code has a mean code word length $\bar{L}$, which corresponds at least to the entropy of the source $H(X)$: $\bar{L} \geq H(X)$.

Possible proof:
The evidence is similar to the evidence of inequation (1.19). Additionally, Kraft’s inequation (1.23) is utilised.

The equals sign is fulfilled if the following applies:

$\sum p(x_i) = \sum\frac{1}{2^k} = 1$

and the code word lengths are selected to:

$L(x_i) = k_i = I(x_i)$.

With general $p(x_i)$, $I(x_i)$ has to be transferred to an integer value by rounding up, so that the following applies:

$I(x_i) \leq L(x_i) < I(x_i) + 1$. 

$L(x_i)$ then continue to fulfil inequation (1.23).

For the example on the right:

$\bar{L} = H(X) = \frac{15}{8}$

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$p(x_i)$</th>
<th>Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1/2</td>
<td>1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>1/4</td>
<td>00</td>
</tr>
<tr>
<td>$x_3$</td>
<td>1/8</td>
<td>010</td>
</tr>
<tr>
<td>$x_4$</td>
<td>1/16</td>
<td>0110</td>
</tr>
<tr>
<td>$x_5$</td>
<td>1/16</td>
<td>0111</td>
</tr>
</tbody>
</table>
Shannon’s coding theorem:
For every source, a binary coding can be found:

\[ H(X) \leq \overline{L} < H(X) + 1 \]  \hspace{1cm} (1.28)

Possible proof:
Multiply inequation (1.27) by \( p(x_i) \) and sum up over all \( i \).

Using the left side of inequation (1.27), proof is provided that with this condition a code with a prefix characteristic exists:

\[ I(x_i) = \text{ld} \left( \frac{1}{p(x_i)} \right) \leq L(x_i) \quad \Rightarrow \quad p(x_i) \geq 2^{-L(x_i)} \]

Summation over all \( i \):

\[ \sum_{i=1}^{N} 2^{-L(x_i)} \leq \sum_{i=1}^{N} p(x_i) = 1 \]

From this, the inequation (1.23) results that guarantees the existence of prefix codes. \hspace{1cm} \text{q.e.d.}
1.3.2 Shannon Coding

Algorithm:
1. Arranging of the occurrence probabilities
2. Determination of the code word lengths \( L(x_i) \) according to inequation (1.27)
3. Calculation of the accumulated occurrence probabilities \( P_i \):
   \[
P_i = \sum_{j=1}^{i-1} p(x_j)
   \]
4. Code words are the \( L(x_i) \) positions after decimal points of the binary representation of \( P_i \).

Example:
0.90 = \(1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} + 1 \cdot 2^{-7} + 0 \cdot 2^{-8} + 0 \cdot 2^{-9} + 1 \cdot 2^{-10} + \ldots\)
Conversion of a decimal number into a binary number:
For example, for the decimal number 0.9 the binary number is to be
developed. This search corresponds to the search for coefficients of the
following equation that can then be solved in the following way:

\[ 0.90 = a_1 \cdot 2^{-1} + a_2 \cdot 2^{-2} + a_3 \cdot 2^{-3} + a_4 \cdot 2^{-4} + \ldots \]
\[ 1.80 = 1 + 0.80 = a_1 \cdot 2^{0} + a_2 \cdot 2^{-1} + a_3 \cdot 2^{-2} + a_4 \cdot 2^{-3} + \ldots \]

Since \(2^0 = 1\), also \(a_1 = 1\) provided that
\[ a_2 \cdot 2^{-1} + a_3 \cdot 2^{-2} + a_4 \cdot 2^{-3} + \ldots < 1 \]

However, this corresponds directly to Kraft’s inequation (1.23), which is
proven.
Thus, by multiplying repeatedly, the coefficients can be determined.
Simplified, this method can then be described:

\[
\begin{align*}
0.90 \cdot 2 &= 0.8 + 1 \\
0.80 \cdot 2 &= 0.6 + 1 \\
0.60 \cdot 2 &= 0.2 + 1 \\
0.20 \cdot 2 &= 0.4 + 0 \
\end{align*}
\]
The tree diagram shows the disadvantage of Shannon coding. Not all endpoints of the tree are taken, that means that the code word length can be reduced. Therefore, this code is not optimal.
1.3.3 Huffman Coding

Huffman coding is based on a recursive approach. The fundamental idea is that the two symbols with the smallest probabilities must have the same code word length; otherwise a reduction of the code word length is possible. Thus, the symbols with the smallest probabilities form the initial point.

Algorithm:
1. Arranging of the symbols according to their probability
2. Assignment of 0 and 1 to the two symbols with the smallest probability
3. Combination of the two symbols with the smallest probability $x_{N-1}$ and $x_N$ to a new symbol with the probability $p(x_{N-1}) + p(x_N)$
4. Repetition of the steps 1 - 3, until only one symbol remains.
### Coding of the example from 1.3.2:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$p(x_i)$</th>
<th>Coding</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0,22</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>0,19</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>0,15</td>
<td>0,41 0,33 0,59</td>
<td>001</td>
</tr>
<tr>
<td>4</td>
<td>0,12</td>
<td>1</td>
<td>011</td>
</tr>
<tr>
<td>5</td>
<td>0,08</td>
<td>0,26 0,18 0</td>
<td>0001</td>
</tr>
<tr>
<td>6</td>
<td>0,07</td>
<td>0,14 0,18 0</td>
<td>0100</td>
</tr>
<tr>
<td>7</td>
<td>0,07</td>
<td>1</td>
<td>0101</td>
</tr>
<tr>
<td>8</td>
<td>0,06</td>
<td>0,0 0,14 0,1</td>
<td>00000</td>
</tr>
<tr>
<td>9</td>
<td>0,04</td>
<td>1</td>
<td>00001</td>
</tr>
</tbody>
</table>
Tree structure of the example Huffman Code:
1.4 Discrete Source Models

1.4.1 Discrete Memoryless Source

The coding possibilities for those sources are considered the characters of which are read out statistically independent from each other.

First, the joint entropy of character strings is to be calculated:

\[ H(X,Y) = - \sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \log(p(x_i, y_k)) \]

\[ = - \sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i) \cdot p(y_k) \cdot [\log(p(x_i)) + \log(p(y_k))] \]

\[ = - \sum_{k=1}^{N} p(y_k) \cdot \sum_{i=1}^{N} p(x_i) \cdot \log(p(x_i)) - \sum_{i=1}^{N} p(x_i) \cdot \sum_{k=1}^{N} p(y_k) \cdot \log(p(y_k)) \]

\[ H(X,Y) = H(X) + H(Y) \]  \hspace{1cm} (1.29)

For \( M \) independent characters of the same source, the following applies:

\[ H(X_1, X_2, \ldots, X_M) = M \cdot H(X) \]  \hspace{1cm} (1.30)
1.4.1 Discrete Memoryless Source

As the general form of the Shannon coding theorem shows, the coding of character strings represents a more efficient method:

\[
H(X_1, \ldots, X_M) \leq \bar{L}_M(X_1, \ldots, X_M) \leq H(X_1, \ldots, X_M) + 1
\]

\[
M \cdot H(X) \leq M \cdot \bar{L} \leq M \cdot H(X) + 1
\]

\[
H(X) \leq \bar{L} \leq H(X) + 1/M
\]

(1.31)

The disadvantage of this method is the strongly increasing coding complexity, since the number of possible characters increases exponentially.

As an example, a binary source with \( X = \{x_1, x_2\} \) and the occurrence probabilities \( p(x_1) = 0.2 \) and \( p(x_2) = 0.8 \), respectively, are considered. Thus, the entropy of the source is \( H(X) = 0.7219 \) bit/character.

As coding algorithm, the Huffman coding is used.
1.4.1 Discrete Memoryless Source

Coding of single characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>$p(x_i)$</th>
<th>Code</th>
<th>$L(x_i)$</th>
<th>$p(x_i) \cdot L(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.2</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Mean code word length: $\bar{L} = 1$ bit/character

Coding of character pairs:

<table>
<thead>
<tr>
<th>Character Pair</th>
<th>$p(x_i, x_j)$</th>
<th>Code</th>
<th>$L(x_i, x_j)$</th>
<th>$p(x_i, x_j) \cdot L(x_i, x_j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1x_1$</td>
<td>0.04</td>
<td>101</td>
<td>3</td>
<td>0.12</td>
</tr>
<tr>
<td>$x_1x_2$</td>
<td>0.16</td>
<td>11</td>
<td>2</td>
<td>0.32</td>
</tr>
<tr>
<td>$x_2x_1$</td>
<td>0.16</td>
<td>100</td>
<td>3</td>
<td>0.48</td>
</tr>
<tr>
<td>$x_2x_2$</td>
<td>0.64</td>
<td>0</td>
<td>1</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Mean code word length: $\bar{L} = 0.78$ bit/character

$\Sigma = 1$
$\Sigma = 1.56$
1.4.1 Discrete Memoryless Source

<table>
<thead>
<tr>
<th>Character Triple</th>
<th>$p(x_i,x_j,x_k)$</th>
<th>Code</th>
<th>$L(x_i,x_j,x_k)$</th>
<th>$p(x_i,x_j,x_k) \cdot L(x_i,x_j,x_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1x_1x_1$</td>
<td>0.008</td>
<td>11111</td>
<td>5</td>
<td>0.040</td>
</tr>
<tr>
<td>$x_1x_1x_2$</td>
<td>0.032</td>
<td>11100</td>
<td>5</td>
<td>0.160</td>
</tr>
<tr>
<td>$x_1x_2x_1$</td>
<td>0.032</td>
<td>11101</td>
<td>5</td>
<td>0.160</td>
</tr>
<tr>
<td>$x_1x_2x_2$</td>
<td>0.128</td>
<td>100</td>
<td>3</td>
<td>0.384</td>
</tr>
<tr>
<td>$x_2x_1x_1$</td>
<td>0.032</td>
<td>11110</td>
<td>5</td>
<td>0.160</td>
</tr>
<tr>
<td>$x_2x_1x_2$</td>
<td>0.128</td>
<td>101</td>
<td>3</td>
<td>0.384</td>
</tr>
<tr>
<td>$x_2x_2x_1$</td>
<td>0.128</td>
<td>110</td>
<td>3</td>
<td>0.384</td>
</tr>
<tr>
<td>$x_2x_2x_2$</td>
<td>0.512</td>
<td>0</td>
<td>1</td>
<td>0.512</td>
</tr>
</tbody>
</table>

Mean code word length: $\bar{L} = 0.728$ bit/character

Summary:

<table>
<thead>
<tr>
<th>$M$</th>
<th>$H(X)$</th>
<th>$\bar{L}$</th>
<th>$H(X) + 1/M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7219</td>
<td>1.0</td>
<td>1.7219</td>
</tr>
<tr>
<td>2</td>
<td>0.7219</td>
<td>0.78</td>
<td>1.2219</td>
</tr>
<tr>
<td>3</td>
<td>0.7219</td>
<td>0.728</td>
<td>1.0552</td>
</tr>
</tbody>
</table>
1.4.2 Discrete Source With Memory

Coding possibilities for those sources, the characters of which are read out interdependently, are considered here. This fact applies to real sources, if correlations occur between the single characters. In this case, the joint entropy is calculated to be:

\[
H(X,Y) = -\sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \cdot \log(p(x_i, y_k))
\]

\[
= -\sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i) \cdot p(y_k | x_i) \cdot \left[ \log(p(x_i)) + \log(p(y_k | x_i)) \right]
\]

\[
= -\sum_{i=1}^{N} \sum_{k=1}^{N} p(y_k | x_i) \cdot p(x_i) \cdot \log(p(x_i)) - \sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \cdot \log(p(y_k | x_i))
\]

\[
H(X,Y) = H(X) + H(Y | X)
\] (1.32a)

with the conditional entropy \( H(Y | X) = -\sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \cdot \log(p(y_k | x_i)) \) (1.32b)
1.4.2 Discrete Source with Memory

Between the source entropy and the conditional entropy, the following relation exists:

\[ H(Y) \geq H(Y \mid X) \] (1.33)

Proof:

\[
H(Y) = -\sum_{k=1}^{N} p(y_k) \cdot \text{ld}(p(y_k)) = -\sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \cdot \text{ld}(p(y_k))
\]

\[
H(Y \mid X) - H(Y) = \sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \cdot \text{ld} \left( \frac{p(y_k)}{p(y_k \mid x_i)} \right)
\]

Estimation with:

\[ \ln x \leq x - 1 \quad \Rightarrow \quad \text{ld} x \leq \frac{1}{\ln 2} (x - 1) \]

\[
H(Y \mid X) - H(Y) \leq \sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \cdot \frac{1}{\ln 2} \cdot \left( \frac{p(y_k)}{p(y_k \mid x_i)} - 1 \right)
\]

with \( p(x, y_k) = p(x_i) \cdot p(y_k \mid x_i) \)

\[
H(Y \mid X) - H(Y) \leq \frac{1}{\ln 2} \left[ \sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i) p(y_k) - \sum_{i=1}^{N} \sum_{k=1}^{N} p(x_i, y_k) \right] = 0 \quad \text{q.e.d.}
\]
With this result, based on equiprobable single characters, the entropy of a source with memory is accordingly smaller than the entropy of a memoryless source. In other words: With sources with memory, a considerably more efficient source coding is possible by coding of character strings utilising the correlations among the single characters, as if the source was considered memoryless.

Example for correlations in texts in German: The probability that the letter „Q“ is followed by „u“, is near 1.

Sources with memory can be modelled as Markoff processes. The respective source models are referred to as Markoff sources.
1.4.2 Discrete Source with Memory

Markoff source:
A discrete source with memory can generally be described as Markoff source. The Markoff sources are based on Markoff processes.
As a basis, a sequence of successively occurring random variables is considered: \( z_0, z_1, z_2, \ldots, z_n \). If these are statistically independent, the following applies in general:

\[
 f_{z_n|z_{n-1}, z_{n-2}, \ldots, z_0} (z_n \mid z_{n-1}, z_{n-2}, \ldots, z_0) = f_{z_n} (z_n)
\]  

(1.34a)

If the values for \( z_i \) are statistically independent, the equation is extended to:

\[
 f_{z_n|z_{n-1}, z_{n-2}, \ldots, z_0} (z_n \mid z_{n-1}, z_{n-2}, \ldots, z_0) = f_{z_n|z_{n-1}, \ldots, z_{n-m}} (z_n \mid z_{n-1}, \ldots, z_{n-m})
\]  

(1.34b)

Frequently, first order Markoff processes are considered \((m = 1)\):  

\[
 f_{z_n|z_{n-1}, z_{n-2}, \ldots, z_0} (z_n \mid z_{n-1}, z_{n-2}, \ldots, z_0) = f_{z_n|z_{n-1}} (z_n \mid z_{n-1})
\]  

(1.34c)
1.4.2 Discrete Source with Memory

$z_i$ can reach a finite number of discrete values: $z_i \in \{x_1, \ldots, x_N\}$. The Markoff process emits these values. A set of values dependent on each other represents a Markoff chain. This chain can be described in full by transition probabilities:

$$p(z_n = x_j \mid z_{n-1} = x_{i_{n-1}}, \ldots, z_0 = x_{i_0}) = p(z_n = x_j \mid z_{n-1} = x_{i_{n-1}}, \ldots, z_{n-m} = x_{i_{n-m}})$$

(1.35a)

If the transition probabilities do not depend on the absolute time, this is referred to as **homogenous** Markoff chain:

$$p(z_n = x_j \mid z_{n-1} = x_{i_1}, \ldots, z_{n-m} = x_{i_m}) = p(z_k = x_j \mid z_{k-1} = x_{i_1}, \ldots, z_{k-m} = x_{i_m})$$

(1.35b)

If the steady state does not depend on the initial probabilities, this is referred to as **stationary** Markoff chain:

$$\lim_{n \to \infty} p(z_{k+n} = x_j \mid z_k = x_i) = \lim_{n \to \infty} p(z_{k+n} = x_j) = p(x_j) = w_j$$

(1.35c)

For first order homogenous and stationary Markoff chains ($y = 1$) applies:

$$p(z_n = x_j \mid z_{n-1} = x_i) = p_{ij}$$

(1.35d)
In case of the homogenous 
stationary Markoff chain, the 
single transition probabilities can be 
combined in a so-called transition matrix:

\[
P = \begin{pmatrix}
    p_{11} & p_{12} & \cdots & p_{1N} \\
    p_{21} & p_{22} & \cdots & p_{2N} \\
    \vdots & \vdots & \ddots & \vdots \\
    p_{N1} & p_{N2} & \cdots & p_{NN}
\end{pmatrix}
\] (1.36a)

The single rows of the transition matrix
must add up to 1:

\[
\sum_{j=1}^{N} p_{ij} = 1 
\] (1.36b)

The transition probabilities of the single discrete values can be combined to
a probability vector:

\[
\mathbf{w} = (w_1, w_2, \ldots, w_N) = (p(x_1), p(x_2), \ldots, p(x_N)) 
\] (1.37)

The probability vector can be determined by means of the steady state:

\[
\mathbf{w} = \mathbf{w} \mathbf{P} 
\] (1.38)
A stationary Markoff source emitting first order Markoff chains is now considered.

For this configuration described by $\mathbf{w}$ and $\mathbf{P}$, the entropy is to be calculated. Then, this one is equal to the entropy in the steady state:

\[
H_\infty(Z) = \lim_{n \to \infty} H(z_n | z_{n-1}, z_{n-2}, \ldots, z_0) = H(z_n | z_{n-1})
\]

\[
= - \sum_{i=1}^{N} \sum_{j=1}^{N} p(z_n = x_j, z_{n-1} = x_i) \cdot \ln(p(z_n = x_j | z_{n-1} = x_i))
\]

\[
= - \sum_{i=1}^{N} w_i \sum_{j=1}^{N} p(z_n = x_j | z_{n-1} = x_i) \cdot \ln(p(z_n = x_j | z_{n-1} = x_i))
\]

\[
H_\infty(Z) = \langle H(z_n | z_{n-1} = x_i) \rangle_i = \sum_{i=1}^{N} w_i \cdot H(z_n | z_{n-1} = x_i)
\]  \hspace{1cm} (1.39)
Since the entropy $H_{\infty}(Z)$ includes characters from the past which have already been known, this is about a conditional entropy. Analogue to inequation (1.33), the following applies to this:

$$
H_{\infty}(Z) = H(z_n \mid z_{n-1}) \leq H(z_n) \leq H_0(z_n)
$$

(1.40)

The source coding of a Markoff source can now be carried out considering the memory. For this, for example, the Huffman coding can be applied taking into account the current state of the source.

The problem caused by this coding represents the **disastrous error propagation**. However, it is a basic problem of the **source codes of a variable length**. An error occurring on the channel that has not been corrected can then lead to a loss of the complete information sequence.
In the following, the example of a Markoff source on the right is considered. Its states correspond to the transmitted characters: $z_n \in \{x_1, x_2, x_3\}$.

The transition probabilities for this example are as follows:

\[
\begin{array}{c|ccc}
   z_n & x_1 & x_2 & x_3 \\
\hline
   z_{n-1} & \\
   x_1 & 0.2 & 0.4 & 0.4 \\
   x_2 & 0.3 & 0.5 & 0.2 \\
   x_3 & 0.6 & 0.1 & 0.3 \\
\end{array}
\]

In the following, $\mathbf{w}$ and $H_\infty(Z)$ are to be calculated.
Calculation of the stationary probabilities $w_i$:

As an approach, equation (1.38) as well as the following equation are applied:

$$\sum_{i=1}^{N} w_i = 1. \quad (1.41)$$

From equation (1.38) three single equations can be obtained here to calculate the three unknowns. Each of the three single equations, however, is linearly dependent on the two other ones so that the second approach to obtain a third independent equation has to be taken into account:

$$\begin{align*}
    w_1 &= 0.2 \ w_1 + 0.3 \ w_2 + 0.6 \ w_3 \\
    w_2 &= 0.4 \ w_1 + 0.5 \ w_2 + 0.1 \ w_3 \\
    w_3 &= 0.4 \ w_1 + 0.2 \ w_2 + 0.3 \ w_3
\end{align*}$$

Strike out one of the three equations!

Solution to the system of equations:

$$\begin{align*}
    w_1 &= \frac{33}{93} \approx 0.3548 \\
    w_2 &= \frac{32}{93} \approx 0.3441 \\
    w_3 &= \frac{28}{93} \approx 0.3011
\end{align*}$$
1.4.2 Discrete Source with Memory

**Calculation of the Entropy $H_\infty(Z)$:**

The approach for this calculation is represented by equation (1.39). Use of the notation for the stationary probabilities $w_i$ and transition probabilities $p_{ij}$ leads to the following notation:

$$H_\infty(Z) = \sum_{i=1}^{N} w_i \cdot H(z_n | z_{n-1} = x_i) = -\sum_{i=1}^{N} w_i \cdot \sum_{j=1}^{N} p_{ij} \cdot \log_2(p_{ij}) \quad (1.42)$$

At first, the single calculations of the state entropy are the obvious procedure for the overview:

$$H(z_n | z_{n-1} = x_1) = 0.2 \cdot \log_2\left(\frac{1}{0.2}\right) + 0.4 \cdot \log_2\left(\frac{1}{0.4}\right) + 0.4 \cdot \log_2\left(\frac{1}{0.4}\right) \approx 1.5219 \text{ bit / character}$$

$$H(z_n | z_{n-1} = x_2) = 0.3 \cdot \log_2\left(\frac{1}{0.3}\right) + 0.5 \cdot \log_2\left(\frac{1}{0.5}\right) + 0.2 \cdot \log_2\left(\frac{1}{0.2}\right) \approx 1.4855 \text{ bit / character}$$

$$H(z_n | z_{n-1} = x_3) = 0.6 \cdot \log_2\left(\frac{1}{0.6}\right) + 0.1 \cdot \log_2\left(\frac{1}{0.1}\right) + 0.3 \cdot \log_2\left(\frac{1}{0.3}\right) \approx 1.2955 \text{ bit / character}$$

The entropy is then:

$$H_\infty(Z) = w_1 H(z_n | z_{n-1} = x_1) + w_2 H(z_n | z_{n-1} = x_2) + w_3 H(z_n | z_{n-1} = x_3)$$

$$\approx 1.441 \text{ bit / character}$$
A comparison of the different values shows:

\[ H_\infty(Z) \cong 1,441 \text{ bit / character} \]

\[ H(Z) = \lim_{n \to \infty} H(z_n) = \sum_{i=1}^{N} w_i \cdot \text{ld} \frac{1}{w_i} \cong 1,5814 \text{ bit / character} \]

\[ H_0 = \text{ld}3 \cong 1,5850 \text{ bit / character} \]

See also inequation (1.40).
For the considered example, the state-dependent Huffman codings are listed below. For every state $z_{n-1}$, an own coding exists that here is also different from the respective other ones:

<table>
<thead>
<tr>
<th>$z_{n-1}$</th>
<th>$z_n$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.3</td>
<td>0.5</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
<td></td>
</tr>
</tbody>
</table>

The mean code word length is then calculated to be:

$$\bar{L} = \langle L(z_n = x_j \mid z_{n-1} = x_i) \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i \cdot p_{ij} \cdot L(z_n = x_j \mid z_{n-1} = x_i)$$ (1.43)

$\cong 1.5054$ bit / character

For comparison: $H_\infty(Z) \cong 1.441$ bit / character

$H(Z) \cong 1.5814$ bit / character
1.5 Communication over a Discrete Memoryless Channel

1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

The discrete memoryless channel (DMC) describes a model for the transmission of discrete values via a channel. Memoryless means that the transmission probabilities of the symbols are independent from each other.

The input signal $X$ and the output signal $Y$ each can reach different discrete values, whereas the possible values at the input and at the output as well as the number of values do not have to be equal:

$$X \in \{x_1, x_2, \ldots, x_{N_x}\} \quad Y \in \{y_1, y_2, \ldots, y_{N_y}\}$$
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

With the transmission, each of the $x_i$ is now transferred into $y_j$ with a certain probability $p_{ij}$:

The transition probabilities $p_{ij}$ are again combined to the transition matrix $\mathbf{P}$. Thereby, the sum of $p_{ij}$ over one line each has to equal 1 (equation (1.36b)).

$$\mathbf{P} = \left( p(Y = y_j \mid X = x_i) \right) = \left( p_{ij} \right)$$

$$= \begin{pmatrix}
  p_{11} & p_{12} & \cdots & p_{1N_Y} \\
p_{21} & p_{22} & \cdots & p_{2N_Y} \\
  \cdots & \cdots & \ddots & \cdots \\
p_{N_X1} & p_{N_X2} & \cdots & p_{N_XN_Y}
\end{pmatrix}$$

(Similar to equation (1.36a), where, however, transitions in sources are concerned!)

(1.44)
A special case of the DMC is the binary channel. X as well as Y consist of only two symbols. The transition matrix is reduced to:

\[
P = \begin{pmatrix}
p_{11} & p_{12} \\
p_{21} & p_{22}
\end{pmatrix}
\]  

(1.45)

The probability \( p(\text{err}) \) that a transition error occurs is calculated to be:

\[
p(\text{err}) = p(x_1) \cdot p(y_2 \mid x_1) + p(x_2) \cdot p(y_1 \mid x_2)
\]

\[
= p(x_1) \cdot p_{12} + p(x_2) \cdot p_{21}
\]

For the binary symmetric channel (BSC) the relations are simplified to:

\[
P = \begin{pmatrix}
1 - p_{\text{err}} & p_{\text{err}} \\
p_{\text{err}} & 1 - p_{\text{err}}
\end{pmatrix}
\]  

(1.46)

\[
p(\text{err}) = p(x_1) \cdot p_{12} + p(x_2) \cdot p_{21}
\]

\[
= \left[ p(x_1) + p(x_2) \right] \cdot p_{\text{err}} = p_{\text{err}}
\]  

(1.47)
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

For the **interference-free channel** it is obvious that the **information at the output has to be the same as the information at the input** of the channel. At the same time, that means that **transmitted information is equal to the entropy of the source**.

With the **interfered channel**, information gets lost with the transmission due to false assignment of several symbols. That means that the information at the output has to be less than the information at the input. This also leads to a decrease of the **transmitted information**, which has then to be **less than the entropy of the source**.

For this reason, the **transinformation** $T(X,Y)$ is defined as a measure for the information actually transmitted. In the following, this transinformation is correlated to the other information streams.
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

Information streams:

Source \( H(U) \) → Source Coder → Channel → Decoder → Sink

- **transinformation**: \( T(X,Y) \)
- **equivocation**: \( H(X|Y) \)
- **irrelevance**: \( H(Y|X) \)
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

Definitions of the entropies in terms of the discrete memoryless channel

**Input entropy:** mean information content of the input symbols

\[ H(X) = \sum_{i=1}^{N_X} p(x_i) \cdot \text{ld} \frac{1}{p(x_i)} \]  
(1.48)

**Output entropy:** mean information content of the output symbols

\[ H(Y) = \sum_{j=1}^{N_Y} p(y_j) \cdot \text{ld} \frac{1}{p(y_j)} \]  
(1.49)

**Joint entropy:** mean uncertainty of the complete transmission system

\[ H(X,Y) = \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} p(x_i, y_j) \cdot \text{ld} \frac{1}{p(x_i, y_j)} \]  
(1.50)
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

**Entropy of irrelevance**: mean information content at the output in case of a known input symbol (conditional entropy)

\[
H(Y | X) = \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} p(x_i, y_j) \cdot \text{ld} \frac{1}{p(y_j | x_i)}
\]  
(1.51)

**Entropy of equivocation**: mean information content at the input in case of a known output symbol (conditional entropy); entropy of information getting lost on the channel

\[
H(X | Y) = \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} p(x_i, y_j) \cdot \text{ld} \frac{1}{p(x_i | y_j)}
\]  
(1.52)
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

From the diagramm of information streams as well as from equation (1.32a) the following relations can be derived:

\[ H(X, Y) = H(Y, X) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y) \]  \hspace{1cm} (1.53)

\[ H(X \mid Y) \leq H(X) \]  \hspace{1cm} (1.54a)

\[ H(Y \mid X) \leq H(Y) \]  \hspace{1cm} (1.54b)

The mean transinformation can be determined from these relations:

\[
T(X, Y) = H(X) - H(X \mid Y) \\
= H(Y) - H(Y \mid X) \\
= H(X) + H(Y) - H(X, Y)
\]  \hspace{1cm} (1.55)
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

In terms of transinformation, two special examples are to be considered, for which the entropy can be derived easily.

For the ideal, not interfered channel applies: 
\[ p_{ij} = \begin{cases} 
1 & \text{für } i = j \\
0 & \text{für } i \neq j
\end{cases} \]

For the entropy then applies: 
\[ H(X|Y) = 0, H(Y|X) = 0, \]
\[ H(X,Y) = H(X) = H(Y) \text{ and thus} \]
\[ T(X,Y) = H(X) = H(Y). \]

In terms of the useless, completely interfered channel the input and output symbols are statistically independent from each other, so that the following applies: 
\[ p(x_i,y_j) = p(x_i) \cdot p(y_j) = p(y_j|x_i) \cdot p(x_i) \Rightarrow p(y_j|x_i) = p(y_j) \Rightarrow p_{ij} = p_{kj}. \]

That means that the transition probabilities are all the same. Regarding the entropy, that means: 
\[ H(X|Y) = H(X), H(Y|X) = H(Y), \]
\[ H(X,Y) = H(X) + H(Y) \text{ and thus} \]
\[ T(X,Y) = 0. \]
1.5.1 Definition and Characteristics of the Discrete Memoryless Channel

Concluding, a concrete example regarding transinformation:
1000 binary, statistically independent and equiprobable symbols \( p(0) = p(1) = 0.5 \) are transmitted via a binary symmetric channel \( p_{\text{err}} = 0.01 \).

Consideration in terms of quality: The mean number of correctly transmitted symbols is 990. However, the exact position of the errors is unknown so that the following has to apply: \( T(X,Y) < 0.99 \) bit/character.

Consideration in terms of quantity:
\[
T(X,Y) = H(Y) - H(Y \mid X)
\]
\[
= \sum_{i=1}^{N_x} p(x_i) \cdot \log \frac{1}{p(x_i)} - \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} p(x_i) \cdot p(y_j \mid x_i) \cdot \log \frac{1}{p(y_j \mid x_i)}
\]
\[
= 1 - (p(0) + p(1)) \left[ (1 - p_{\text{err}}) \log \frac{1}{1 - p_{\text{err}}} + p_{\text{err}} \log \frac{1}{p_{\text{err}}} \right]
\]
\[
= 1 - S(p_{\text{err}}) \approx 0.9192 \text{ bit/character}
\]
1.5.2 Channel Capacity

Considering a given channel, it is interesting to know how much information can be sent via it at most. The transinformation, however, depends on the probability distribution of the source symbols and thus on the source so that it is not suitable to be a measure for the description of channels. Nevertheless, decoupling from the source is possible by considering the maximum value of the transinformation. Thus, the channel capacity is defined as:

\[
C = \frac{1}{\Delta T} \max_{p(x_i)} T(X,Y)
\]

(1.56)

The channel capacity \(C\) is the maximum of the flow of information and has the dimension bit/s. \(\Delta T\) describes the time period of the characters transmitted via the channel. With this definition, \(C\) only depends on the channel characteristics, that means on the transition probabilities and not on the source.
1.5.2 Channel Capacity

Overview of a communication system:

Definition of flow of information:
Flow of information (entropy / time): \( H'(X) = H(X) / \Delta T \)
Flow of transinformation (transinformation / time): \( T(X,Y) = T(X,Y) / \Delta T \)
Flow of decision (decision content / time): \( H_0'(X) = H_0(X) / \Delta T \)
1.5.2 Channel Capacity

Taking the binary symmetric channel (BSC) as an example, the channel capacity is to be derived. For this, the transinformation is to be determined and subsequently the maximum is to be found.

Transinformation of the BSC:

\[
T(X,Y) = H(Y) - H(Y \mid X)
\]

\[
= \left( p_1 + p_{err} - 2p_1p_{err} \right) \log_2 \frac{1}{p_1 + p_{err} - 2p_1p_{err}} \]

\[
+ \left( 1 - p_1 - p_{err} + 2p_1p_{err} \right) \log_2 \frac{1}{1 - p_1 - p_{err} + 2p_1p_{err}} \]

\[
- \left[ p_{err} \log_2 \frac{1}{p_{err}} + \left( 1 - p_{err} \right) \log_2 \frac{1}{1 - p_{err}} \right]
\]

(1.57)
Graph of the transinformation over different transmission error probabilities:

The position where the maximum of the transinformation at the BSC occurs is independent from the transmission error probability. It is $p_1 = p(x_1) = 0.5$ (equiprobable source symbols).
1.5.2 Channel Capacity

Graph of the channel capacity. It can be derived from the graph of the transinformation on the previous page by plotting the respective maxima of the transinformation over $p_{\text{err}}$:

$$C \cdot \Delta T / \text{bit}$$

$$C \cdot \Delta T = \max_{p(x_i)} T(X,Y) = 1 - \left[ p_{\text{err}} \log \frac{1}{p_{\text{err}}} + (1 - p_{\text{err}}) \log \frac{1}{1 - p_{\text{err}}} \right] = 1 - S(p_{\text{err}})$$ (1.58)
1.5.2 Channel Capacity

As a further example, the binary erasure channel (BEC) is considered. It models the scenario that shows that a received signal cannot be assigned to one of the transmitted signals. The data and results are as follows:

\[
P = \begin{pmatrix} 1 - p_{\text{err}} & 0 & p_{\text{err}} \\ 0 & 1 - p_{\text{err}} & p_{\text{err}} \end{pmatrix}
\]

(1.59) \hspace{1cm} C \cdot \Delta T = 1 - p_{\text{err}} \hspace{1cm} (1.60)

\[C \cdot \Delta T / \text{bit}\]

\[\text{p}_{\text{err}}\]
1.5.2 Channel Capacity

Theorem of the channel capacity (Shannon 1948):

For every \( \epsilon > 0 \) and every flow of information of a source \( R_{\text{inf}} \) less than the channel capacity \( C \) (\( R_{\text{inf}} < C \)), a binary block code of the length \( n \) (\( n \) above a threshold) exists, so that the residual-error probability after decoding at the receiver is less than \( \epsilon \).

Inversion: Whatever the effort, for \( R_{\text{inf}} > C \), the residual-error probability cannot go below a certain limit.

The argumentation is carried out using random block codes (random coding argument). In doing so, the proof of the average over all codes is carried out. It can be seen here that all known codes are bad codes. The theorem of the channel capacity, however, does only state that codes with such characteristics exist. There is no construction rule given by Shannon.
1.5.2 Channel Capacity

In terms of the channel capacity, use of infinite code word lengths (that means $n$ is very large) would be optimal. This, however, would bring about an infinite delay and complexity.

Thus, Gallager improves the theorem of the channel capacity applying his error exponent $E_G(R_C)$ for a DMC with $N_X$ input symbols. The objective is an estimate that is valid for smaller code word lengths.

$$E_G(R_C) = \max_{0 \leq s \leq 1} \max_{-s \cdot R_C - \log \sum_{j=1}^{N_Y} \left( \sum_{i=1}^{N_X} p(x_i) \cdot p(y_j | x_i)^{1+s} \right)}^{1+s} (1.61)$$

An $(n,k)$ block code ($k$: information word length) always exists using

$$R_C = \frac{k}{n} \log N_X < C \Delta T, \quad (1.62)$$

so that the following applies for the word error probability:

$$P_w < 2^{-n \cdot E_G(R_C)} \quad (1.63)$$
The error exponent according to Gallager has the following characteristics:

\[ E_G(R_C) > 0 \quad \text{for} \quad R_C < C \]
\[ E_G(R_C) = 0 \quad \text{for} \quad R_C \geq C \]

The \( R_0 \) value is defined as:

\[ R_0 = E_G(R_C = 0) \] (1.64)
The maximum for $E_G(R_C = 0)$ is $s = 1$. The $R_0$ value, which is also referred to as „computational cut-off rate“, is then:

$$R_0 = E_G(R_C = 0) = \max_{p(x_i)} \left[ -\log \sum_{j=1}^{N_Y} \left( \sum_{i=1}^{N_X} p(x_i) \cdot \sqrt{p(y_j \mid x_i)} \right)^2 \right]$$  (1.65)

If $R_C$ is not put to zero, the equation reads as follows:

$$E_G(R_C) \geq \max_{p(x_i)} \left[ -R_C - \log \sum_{j=1}^{N_Y} \left( \sum_{i=1}^{N_X} p(x_i) \cdot \sqrt{p(y_j \mid x_i)} \right)^2 \right] = R_0 - R_C$$  (1.66)
1.5.2 Channel Capacity

\textbf{$R_0$ theorem:}

An \((n,k)\) block code with

\[ R_C = \frac{k}{n} \log N_X < C \Delta T, \]

always exists so that for the word error probability (with Maximum-Likelihood decoding) the following applies:

\[ P_w < 2^{-n(R_0-R_C)} \quad (1.67) \]

As in case of the theorem of the channel capacity, this theorem only provides information about the existence and does not provide any construction rule for good codes.

Co-domain for the code rate:

\begin{align*}
0 & \leq R_C \leq R_0 & \text{estimate of } P_w \text{ using (1.67)} \\
R_0 & \leq R_C \leq C \Delta T & \text{estimate of } P_w \text{ is difficult to calculate} \\
R_C & > C \Delta T & P_w \text{ cannot become arbitrarily small}
\end{align*}
Comparison of channel capacity and $R_0$ value for a BSC:

The $R_0$ graph is below the graph representing the channel capacity and indicates a limit predicting worse characteristics of the code. However, this limit is more realistic in the sense that it can be reached by finite code word lengths.
2 Fundamentals of Channel Coding

In chapter 1, the influence of interferences on the channel caused by reduction of the transmitted information was described. Practically, interference results in random errors occurring with the transmission of symbols.

Dependent on a quality criterion like the bit error rate (BER), a certain number of (residual) errors can be accepted, for the limitation of which channel codes have to be applied. With voice transmission, e.g., a BER of $10^{-3}$ is applied.

In this chapter, the fundamental parameters of channel coding are to be described:
- code rate,
- minimum distance and
- error probabilities.

Furthermore, decoding rules are introduced and bounds theoretically achievable are identified.
2.1 Introduction

Basic principle of channel coding:

The transmission line with channel coding/decoding is divided into:

- Classification of the source symbol stream to information words $u$ of the length $k$,
- Assignment of $u$ to a transmit word $x \in C$ of the length $n$,
- Transmission of $x$; with this, superposition of an error word $f$,
- Estimation of the receive word $y = x + f$, that means assignment of $y$ to the most likely code word $\hat{x} \in C$,
- Assignment of $\hat{x}$ to the estimated information word $\hat{u}$.
Regarding this, the task of coding is to add redundancy to the error protection (redundant codes).

The code space, that means the set of all code words, is referred to as C.

For the coding

\[ u = (u_0, u_1, \ldots, u_{k-1}) \rightarrow c = (c_0, c_1, \ldots, c_{n-1}) \]

applies \( n \geq k \). The \( m = n - k \) added symbols are referred to as check symbols. Each of the single symbols \( u_i \) and \( c_i \), respectively, can reach \( q \) values in general (\( q \): number of symbols). Then there are \( N = q^k \) different messages and code words, respectively and \( q^n - q^k \) invalid words.

In this chapter, only binary block codings are considered, that means \( q = 2 \), and \( k \) and \( n \), respectively, are values that are constant for a code. In the following, the repetition codes and the parity check codes are considered as examples.
2.1 Introduction

Repetition code:
The information consists of an information bit \( k = 1 \) that is repeated \( (n - 1) \)-fold. With this, all in all \( 2^k = 2 \) code words exist which are
\[
\mathbf{c}_1 = (0 \ 0 \ 0 \ ... \ 0) \quad \text{and} \quad \mathbf{c}_2 = (1 \ 1 \ 1 \ ... \ 1).
\]
The decoding is carried out by majority decision, if \( n \) is odd.

Example: \( n = 5 \implies R_C = 1/5 \)
\[
\begin{align*}
\mathbf{u}_1 &= (0) & \mathbf{c}_1 &= (0 \ 0 \ 0 \ 0 \ 0) \quad \text{and} \\
\mathbf{u}_2 &= (1) & \mathbf{c}_2 &= (1 \ 1 \ 1 \ 1 \ 1)
\end{align*}
\]

Interfered receive sequences with:
\[
\begin{align*}
\mathbf{f}_1 &= (0 \ 1 \ 0 \ 0 \ 1) \quad \text{and} \quad \mathbf{x}_1 &= \mathbf{c}_2 = (1 \ 1 \ 1 \ 1 \ 1) \\
\mathbf{y}_1 &= \mathbf{x}_1 + \mathbf{f}_1 = (1 \ 0 \ 1 \ 1 \ 0) \quad \implies \hat{\mathbf{u}}_1 &= (1) \quad \text{(successful decoding)}
\end{align*}
\]
\[
\begin{align*}
\mathbf{f}_2 &= (1 \ 1 \ 0 \ 1 \ 0) \quad \text{and} \quad \mathbf{x}_2 &= \mathbf{c}_1 = (0 \ 0 \ 0 \ 0 \ 0) \\
\mathbf{y}_2 &= \mathbf{x}_2 + \mathbf{f}_2 = (1 \ 1 \ 0 \ 1 \ 0) \quad \implies \hat{\mathbf{u}}_2 &= (1) \quad \text{(decoding error)}
\end{align*}
\]
The addition is made bit by bit without carry.
2.1 Introduction

Parity Check Code:
The code consists of \( k \) information bits and a control bit \((m = 1)\). With this, \( n = k + 1 \). There are \( 2^k \) code words of the form

\[
c = (u \ p) \quad \text{with} \quad p = u_0 + u_1 + u_2 + \ldots + u_{k-1}.
\]

From this, an even number of ones in the code words always results.

The parity check code reads then as follows:

\[
s_0 = y_0 + y_1 + y_2 + \ldots + y_{n-1} = 0
\]

\[\rightarrow \text{no error}\]

\[
s_0 = y_0 + y_1 + y_2 + \ldots + y_{n-1} = 1
\]

\[\rightarrow \text{error}\]

Example: \( k = 3 \)

\[
y_1 = (0110) \quad \rightarrow \text{no error detected}
\]

\[
y_2 = (1110) \quad \rightarrow \text{error detected}
\]
2.2 Code Rate

A source generates information with a mean rate of $R_i$ bit per second (bps), that means every $T_i = 1/R_i$ second one bit is transmitted. The channel coder assigns a code word of the length $n$ to every message of the length $k$ so that the channel bit rate reads as follows:

$$R_k = R_i \cdot \frac{n}{k} \quad (2.1)$$

The code rate is then:

$$R_C = \frac{k}{n} = R_i \cdot \frac{T_K}{T_i} \quad (2.2)$$

since the following applies: $n \geq k$, ist $R_C \leq 1$.

The code rate measures the **relative information content in every code word** and is one of the **key values for the performance of a channel code**. Though a high value of $R_C$ means that more information is transmitted, protection from transmission errors is more difficult due to the smaller number of redundant symbols.
2.2 Code Rate

In some systems, $R_i$ and $R_K$ are fixed parameters. For a given message length $k$, $n$ is then the greatest integer that is less than $k \cdot R_K / R_i$. If this term itself is no integer, dummy bits have to be included in order to maintain the synchronisation.

Example:
A system is given by $R_i = 3 \text{ bps}$ and $R_K = 4 \text{ bps}$. With this, $R_C = \frac{3}{4}$.

If a channel coder is to be made available with $k = 3$, $n = 4$ has to be chosen; loss of synchronisation does not occur.

If a channel coder with $k = 5$ is to be made available, for the product results $k \cdot R_K / R_i = 6 + \frac{2}{3}$. Thus, $n = 6$. However, 2 dummy bits have to be added after every third code word in order to close the gap of $\frac{2}{3}$.

$$
\begin{array}{ccc}
\hline
k = 5 & k = 5 & k = 5 \\
X X X X X & X X X X X & X X X X X \\
\downarrow & \downarrow & \downarrow \\
O O O O O O & O O O O O O & O O O O O O \\
n = 6 & n = 6 & n = 6 \\
\hline
\end{array}
$$

15 bits

$$
R_C = \frac{3}{4}
$$

d d
20 bits
dummy bits
2.3 Decoding Rules

The channel decoder has to use a decoding rule for the receive word $y$ in order to decide which transmit word $x$ was transmitted. That means that a decision rule $D(.)$ is searched for, for which the following applies: $x = D(y)$.

Let $p(y \mid x)$ be the probability that $y$ was received after transmitting $x$. For a discrete memoryless channel it can be expressed by the transmission probabilities:

$$p(y \mid x) = \prod_{i=0}^{n-1} p(y_i \mid x_i) \quad (2.3)$$

Let $p(x)$ be the a-priori probability for the message $x$ to be sent. By means of Bayes' rule (equation 1.9) the probability that $x$ was transmitted when $y$ was received is given by:

$$p(x \mid y) = \frac{p(y \mid x) \cdot p(x)}{p(y)} \quad (2.4)$$
From the transmission point of view, the 3 possible results are as follows:

- $y = x$, that means **error-free transmission**: $\hat{x} = x$, $\hat{u} = u$,
- $y \neq x$, $y \in C$, that means **falsification of the receive word** into another code word (not corrigible): $\hat{x} \neq x$, $\hat{u} \neq u$
- $y \neq x$, $y \notin C$, that means **erroneous transmission**, in which the error is observable and possibly corrigible: $\hat{x} \neq x$, $\hat{u} \neq u$

From the receiver’s point of view, there are 3 possible decoding results:

- **Correct decoding**, where either no errors have occurred on the channel or all errors have been corrected properly: $\hat{x} = x$, $\hat{u} = u$
- **Erroneous decoding**, that means an error has occurred in such a way that the estimation has assigned an erroneous code word to the receive word: $\hat{x} \neq x$, $\hat{u} \neq u$
- **Decoding failure**, that means an error was detected, however, the decoder does not find a solution for this error.
The objective is now to minimise the word error probability. For this, \( \hat{x} \) has to be selected in such a way that \( p(y \mid \hat{x}) \) becomes maximum. Using equation (2.4), there is also the possibility to maximise \( p(\hat{x} \mid y) \).

Word error probability for a DMC:

\[
p_W = p(\hat{u} \neq u) = p(\hat{x} \neq x) = \sum_{x \in C} p(\hat{x} \neq x \mid x \ \text{gesendet}) \cdot p(x \ \text{gesendet}) \quad (2.5)
\]

The limited probability for an estimation error is:

\[
p(\hat{x} \neq x \mid x \ \text{gesendet}) = \sum_{y, \hat{x} \neq x} p(y \ \text{empfangen} \mid x \ \text{gesendet}) = \sum_{y, \hat{x} \neq x} p(y \mid x) \quad (2.6)
\]

It is assumed that the code words are transmitted with the same probability. Then for \( p_w \) the following applies:

\[
p_W = \sum_{x \in C} \sum_{y, \hat{x} \neq x} p(y \mid x) \cdot 2^{-k} = 2^{-k} \cdot \left[ \sum_{x \in C, y} p(y \mid x) - \sum_{x \in C, y, \hat{x} = x} p(y \mid x) \right]
\]

\[
= 1 - 2^{-k} \cdot \sum_{x \in C, y, \hat{x} = x} p(y \mid x) = 1 - 2^{-k} \cdot \sum_{y} p(y \mid \hat{x}) \quad (2.7)
\]

The objective is now to minimise the word error probability.
Maximum A-Posteriori rule:
The probability that the decision for the receive word $y$ to the code word $x$ is correct is just $p(x \mid y)$. The probability that an error occurs is then $1 - p(x \mid y)$. Thus, the minimisation of the estimation error with the selection of $x$ is the maximisation of the a-posteriori probability $p(x \mid y)$.

The corresponding maximum a-posteriori rule (MAP rule) reads as follows:

$$\hat{x} = D_{\text{MAP}}(y), \quad \text{with } \hat{x} \in \mathbf{C}, \quad \text{(2.8a)}$$

so that

$$p(x = \hat{x} \mid y) \geq p(x \mid y) \quad \text{for all } x \in \mathbf{C} \quad \text{(2.8b)}$$

Using equation (2.4) and since $p(y)$ is independent from $x$, equation (2.8b) is simplified to:

$$p(y \mid x = \hat{x})p(x = \hat{x}) \geq p(y \mid x)p(x) \quad \text{for all } x \in \mathbf{C} \quad \text{(2.8c)}$$

This decoding rule results in a minimum decoding error.
2.3 Decoding Rules

Maximum Likelihood decoding rule:
An alternative decoding rule is based on the maximisation of \( p(y \mid x) \), the probability that with the transmission of \( x \) the word \( y \) was received. This rule does not guarantee a minimum of errors, but it is easier to implement, since the channel input probabilities are not required.

The corresponding Maximum Likelihood Decoding rule (MLD rule) reads as follows:

\[
\hat{x} = D_{\text{MLD}}(y), \quad \text{with} \quad \hat{x} \in C,
\]

so that

\[
p(y \mid x = \hat{x}) \geq p(y \mid x) \quad \text{for all} \quad x \in C \quad (2.9a)
\]

For equiprobable input parameters, the MLD and the MAP rule lead to the same results (compare the following example).
2.3 Decoding Rules

Example: A BSC with $p_{err} = 0.4$ and a channel coder with $(k,n) = (2,3)$ are assumed, that means a channel code with $N = 2^2 = 4$ code words of the length $n = 3$. The code words and their occurrence probabilities are assumed to be:

<table>
<thead>
<tr>
<th>Code word</th>
<th>$p(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 = (000)$</td>
<td>0.4</td>
</tr>
<tr>
<td>$x_2 = (011)$</td>
<td>0.2</td>
</tr>
<tr>
<td>$x_3 = (101)$</td>
<td>0.1</td>
</tr>
<tr>
<td>$x_4 = (110)$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Assumption: $y = (111)$. The MLD rule provides:
- $p(y | x_1) = p(111 | 000) = p(1|0) \cdot p(1|0) \cdot p(1|0) = 0.064$
- $p(y | x_2) = p(111 | 011) = p(1|0) \cdot p(1|1) \cdot p(1|1) = 0.144$
- $p(y | x_3) = p(111 | 101) = p(1|1) \cdot p(1|0) \cdot p(1|1) = 0.144$
- $p(y | x_4) = p(111 | 110) = p(1|1) \cdot p(1|1) \cdot p(1|0) = 0.144$

This rule provides three equal maximum probabilities, so that a unique decision is not possible.

The MAP rule, however, provides (by means of equation (2.8c)):
- $p(y | x_1) \cdot p(x_1) = 0.064 \cdot 0.4 = 0.0256$ Minimising the error is unique here by selection of
- $p(y | x_2) \cdot p(x_2) = 0.144 \cdot 0.2 = 0.0288$ the code word $x_4$, that means
- $p(y | x_3) \cdot p(x_3) = 0.144 \cdot 0.1 = 0.0144$ $x_4 = D_{MAP}(y = (111))$.  
- $p(y | x_4) \cdot p(x_4) = 0.144 \cdot 0.3 = 0.0432$
Decoding sphere:
The decoding sphere describes an $n$-dimensional sphere around a code word with the radius $t$. All $n$-digit receive words located within a definite decoding sphere are decoded as the respective code word.

It applies that the total number of all vectors within decoding spheres is less than or equal the total number of all possible words.

The decoding sphere forms the basis for the Hamming bound (see below).
Decoding methods:
A block code that can definitely correct $t$ errors is considered.

**Maximum-Likelihood Decoding (MLD):**
The receive word is compared to all code words ($2^k$ necessary vector comparisons). Though this method is optimal, it is very complex. Possibly, more than $t$ errors are corrigible.

**Limited Distance Decoding (BDD):**
Every code word is surrounded by decoding spheres with the radius $t_0$, which can overlap. Decoding takes place only for those receive vectors $y$ exactly located in a sphere. All receive vectors $y$ located in no sphere or in several spheres are not decoded.

**Limited Minimum Distance Decoding (BMD):**
Every code word is surrounded by a decoding sphere with the radius $t$. The decoding spheres are disjoint. Decoding takes place only for the receive vectors $y$ located in a sphere.
2.3 Decoding Rules

Further decoding methods are e.g. the **syndrome decoding** and the **majority decision**. The syndrome decoding is easy to realise. However, it is suboptimal since not the complete information is used. The majority decision has already been known from the repetition codes and is suboptimal as well.
2.4 Hamming Distance

Assumption: The two binary code words $x$ and $y$ of the length $n$ are given. The Hamming distance between $x$ and $y$, $d_H(x,y)$, is defined as the number of position bits in which $x$ and $y$ differ. For arbitrary binary words $x$, $y$ and $z$, the Hamming distance fulfills the following conditions:

- $d_H(x,y) \geq 0$, with equality, if $x = y$, \hspace{1cm} (2.10a)
- $d_H(x,y) = d_H(y,x)$ and \hspace{1cm} (2.10b)
- $d_H(x,y) + d_H(y,z) \geq d_H(x,z)$ (triangle inequality). \hspace{1cm} (2.10c)

Example: Let $n = 8$ and $x = (11010001)$, $y = (00010010)$ and $z = (01010011)$. The Hamming distances for all combinations are given in the table.

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_H$</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$y$</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$z$</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Check of the triangle inequality (equation (2.10c)) results in:

$$d_H(x,y) + d_H(y,z) \geq d_H(x,z) \Rightarrow 4 + 2 \geq 2$$
2.4 Hamming Distance

(Hamming) weight:
A binary vector \( x \) of the length \( n \) is given. Then the Hamming weight \( w_H(x) \) is defined as the number of ones in \( x \):

\[
w_H(x) = \sum_{i=0}^{n-1} x_i
\]  \hspace{1cm} (2.11)

Example: \( w_H(0\ 1\ 1\ 1\ 0\ 1\ 0\ 1) = 5 \).

Between the Hamming distance \( d_H \) and the Hamming weight \( w_H \) the following relation exists:

\[
d_H(x,y) = w_H(x + y).
\]  \hspace{1cm} (2.12)

Example: \( x = (0\ 1\ 1\ 1\ 0\ 1\ 0\ 1) \)
\( y = (1\ 0\ 1\ 0\ 0\ 1\ 0\ 1) \) \( \Rightarrow \) \( d_H(x,y) = 3 \)
\( x + y = (1\ 1\ 0\ 1\ 0\ 0\ 0\ 0) \) \( \Rightarrow \) \( w_H(x + y) = 3 \)
2.4.1 Decoding Rule for the BSC

A BSC with the bit error probability $p_{\text{err}}$ is assumed. With regard to the MLD rule, for the conditional probability the following results:

$$p(y \mid x) = \prod_{i=0}^{n-1} p(y_i \mid x_i) = \prod_{i=0}^{n-1} \left\{ \begin{array}{ll}
1 - p_{\text{err}} & \text{für } y_i = x_i \\
p_{\text{err}} & \text{für } y_i \neq x_i
\end{array} \right. $$

$$= (1 - p_{\text{err}})^{n-d_H(y,x)} \cdot p_{\text{err}}^{d_H(y,x)} = (1 - p_{\text{err}})^n \cdot \left( \frac{p_{\text{err}}}{1 - p_{\text{err}}} \right)^{d_H(y,x)} \tag{2.13}$$

Equation (2.13) is maximum, if $d_H(x,y)$ is minimum.

According to the MLD decoding rule, the estimated vector $\hat{x}$ is to be searched in that way that it is nearest to the receive word $y$:

$$d_H(y,\hat{x}) \leq d_H(y,x) \quad \text{for all } x \in \mathbf{C} \tag{2.14}$$

Moreover, equation (2.13) states that, if $p_{\text{err}} < 0.5$, the probability for one error is greater than for two or three errors etc., that means

$$p(i \text{ errors}) > p(i + 1 \text{ errors}), \quad 0 \leq i \leq n - 1. \tag{2.15}$$
Example: The following channel code is assumed:

Since \( n = 3 \), \( N = 8 \) possible receive words exist, 4 of which are no code words. If the receive word is one of the code words, the decoding rule also assigns this code word to the estimate \( \hat{x} = y \). If \( y \) is no code word, however, the decoding rule reads as follows:

<table>
<thead>
<tr>
<th>Message</th>
<th>Code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>000</td>
</tr>
<tr>
<td>01</td>
<td>001</td>
</tr>
<tr>
<td>10</td>
<td>011</td>
</tr>
<tr>
<td>11</td>
<td>111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( y )</th>
<th>( d_H(y, e_i) )</th>
<th>Next code word</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>1, 2, 1, 2</td>
<td>000; 011</td>
<td>1 bit error detected</td>
</tr>
<tr>
<td>100</td>
<td>1, 2, 3, 2</td>
<td>000</td>
<td>1 bit error corrected</td>
</tr>
<tr>
<td>101</td>
<td>2, 1, 2, 1</td>
<td>001; 111</td>
<td>1 bit error detected</td>
</tr>
<tr>
<td>110</td>
<td>2, 3, 2, 1</td>
<td>111</td>
<td>1 bit error corrected</td>
</tr>
</tbody>
</table>

The separate consideration of possible pairs of words is the disadvantage of this approach. A parameter describing the general detection and correction characteristics for a code is more interesting.
2.4.2 Error Detection and Error Correction

Minimum distance of a linear code*:
The minimum distance \( d_{\text{min}} \) of a linear block code \( \mathbf{C} \) is given by the smallest Hamming distance of all pairs of different code words:

\[
d_{\text{min}} = \min \left( d_H(c_1, c_2) \mid c_1, c_2 \in \mathbf{C} \land c_1 \neq c_2 \right)
\]  

(2.16)

The minimum distance of a linear code word corresponds to the minimum weight of the code without considering the zero word:

\[
d_{\text{min}} = \min \left( d_H(c_1, c_2) \mid c_1, c_2 \in \mathbf{C}; c_1 \neq c_2 \right) = \min \left( w_H(c) \mid c \in \mathbf{C}; c \neq 0 \right) = w_{\text{min}}
\]

(2.17)

Proof:

\[
d_{\text{min}} = \min \left( d_H(c_1, c_2) \mid c_1, c_2 \in \mathbf{C}; c_1 \neq c_2 \right)
\]

\[
= \min \left( d_H(0, c_1 + c_2) \mid c_1, c_2 \in \mathbf{C}; c_1 \neq c_2 \right)
\]

\[
= \min \left( d_H(0, c) \mid c \in \mathbf{C}; c \neq 0 \right)
\]

\[
= \min \left( w_H(c) \mid c \in \mathbf{C}; c \neq 0 \right)
\]

(*) linear codes: see chapter 3
2.4.2 Error Detection and Error Correction

Error detection characteristic:
A block code $\mathbf{C}$ detects up to $t$ errors, if its minimum distance is greater than $t$:

$$d_{\text{min}} > t$$  \hspace{1cm} (2.18a)

This is an important characteristic for ARQ methods, where codes for error detection are required.

Error correction characteristic:
A block code $\mathbf{C}$ corrects up to $t$ errors, if its minimum distance is greater than $2t$:

$$d_{\text{min}} > 2t$$  \hspace{1cm} (2.18b)

This is an important characteristic for FEC methods where codes for error correction are required.
Thus, the number of detectable errors is given by $t_e = d_{\text{min}} - 1$. (2.19)

The number of corrigible errors is given by

$$t = \frac{(d_{\text{min}} - 2)}{2}, \text{ if } d_{\text{min}} \text{ is even and}$$
$$t = \frac{(d_{\text{min}} - 1)}{2}, \text{ if } d_{\text{min}} \text{ is odd.}$$

(2.20a) (2.20b)

With $d_{\text{min}} = 1$, neither an error detection nor an error correction can be guaranteed; with $d_{\text{min}} = 2$ at least one single error is always detectable; with $d_{\text{min}} = 3$ at least one single error is always corrigible and at least two errors are detectable etc.
Examples:
1) With the repetition code, \((n - 1)/2\) errors (rounded down) are corrigible and \(n - 1\) errors are detectable.
2) With the parity check, no error is corrigible and an odd number of errors is detectable in general.
3) In the example of the table below, an error is definitely detectable since \(d_{\text{min}} = 2\).

<table>
<thead>
<tr>
<th>Code word</th>
<th>Information</th>
<th>Control bit</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_0</td>
<td>000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c_1</td>
<td>001</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>c_2</td>
<td>010</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>c_3</td>
<td>011</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>c_4</td>
<td>100</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>c_5</td>
<td>101</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>c_6</td>
<td>110</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>c_7</td>
<td>111</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
2.5 Bounds and Perfect Codes

As observed in 2.4, the minimum distance $d_{\text{min}}$ specifies the ability of a code $C$ to detect errors. Based on different problems, interesting questions result:

- A code $C$ of the length $n$ is to be generated which shall have the minimum distance $d_{\text{min}}(C)$. Does then a maximum number (upper bound) of possible code words (messages) $N$ exist?
- A code $C$ of the length $n$ for messages of the length $k$ is to be developed. Which is the maximum error protection (that means maximum $d_{\text{min}}$) obtainable by these parameters?
- A code $C$ for messages of the length $k$ with a given error protection ($t$ bits) is to be developed. Which is the shortest length $n$ of a code word that can be used for this?

In the following, various bounds are to be considered.
Singleton bound:
The minimum distance and the minimum weight of a linear code $C$ are bounded by

$$d_{\text{min}} = w_{\text{min}} \leq 1 + n - k = 1 + m$$  \hspace{1cm} (2.21)

Proof: In general, $q^k$ different code words exist, which have a minimum distance of $d_{\text{min}}$ to each other. If the last $d_{\text{min}} - 1$ symbols are removed from every code word, at least $q^k$ different words still have to remain, however. Otherwise the coding would not have a minimum distance of $d_{\text{min}}$.

All in all, $q^n$ different words of the length $n$ exist in general (all possible combinations). If the last $d_{\text{min}} - 1$ symbols are removed here, at most $q^n - (d_{\text{min}} - 1)$ words are different from each other. Thus, the following has to apply: $n - (d_{\text{min}} - 1) = n - d_{\text{min}} + 1 \geq k$ or $d_{\text{min}} \leq n - k + 1$.

A code for which the equals sign in equation (2.21) applies, is called Maximum Distance Separable (MDS).
From equation (2.21), using equation (2.19) the following can be derived for the error detection:

\[ t_e + 1 = d_{\text{min}} \leq 1 + m \Rightarrow m \geq t_e \quad (2.22a) \]

That means that at least one control bit per observable error is required.

For error correction, the following can be derived from equations (2.20a/b):

\[
\frac{(d_{\text{min}} - 1)}{2} \geq t \geq \frac{(d_{\text{min}} - 2)}{2} \\
2t + 1 \leq d_{\text{min}} \leq 1 + m \Rightarrow m \geq 2t \quad (2.22b)
\]

That means that at least two control bits per corrigible error are required.
Hamming bound:
For a binary \((n,k)\) block code with \(N = 2^k\) code words of the length \(n\) and the correction capability \(t\) the following applies:

\[
2^k \sum_{i=0}^{t} \binom{n}{i} \leq 2^n \quad (2.23)
\]

Proof: The total number of all words within the \(2^k\) existing decoding spheres is at most equal to the number of all possible words \((2^n)\), i.e.:

\[
2^k \left[ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{t} \right] \leq 2^n \quad \text{with} \quad \binom{n}{t} = \frac{n \cdot (n-1) \cdots (n-(t-1))}{t \cdot (t-1) \cdots 1}
\]

- number of words around a code word with \(d_H = t\)
- \(\vdots\)
- number of words around a code word with \(d_H = 2\)
- number of words around a code word with \(d_H = 1\)
- code word (centre of a decoding sphere)
2.5 Bounds and Perfect Codes

Perfect codes:
A code $C$ with the variables $n$, $k$ and $d_{\text{min}}$ is referred to as perfect if equality within the Hamming bound (equation (2.23)) applies. Graphically, this means that all $q^n$ possible receive words are located within the correction spheres of the $q^k$ code words. Thus, with perfect codes all corrupted code words can be assigned unambiguously to a correct code word.

The repetition code of odd length, e.g., is a perfect code.

Further bounds:
Furthermore, there are the Plotkin and the Elias bound representing upper bounds. They are justified by different focuses in different ranges of parameters.

The Gilbert-Varshamov bound is a lower bound, the adherence of which guarantees the existence of a code.

(for further information, see Schneider-Obermeier, Friedrichs)
Asymptotic bounds for the minimum distance:
Eventually, the dependency between the code rate $R_C$ and the normalised distance $d_{\text{min}}/n$ is to be considered. For large code word lengths ($n \to \infty$) the following terms arise for the mentioned bounds:

- **Singleton bound:**
  \[ R_C \leq 1 - \frac{d_{\text{min}}}{n} \] \hspace{1cm} (2.24a)

- **Hamming bound:**
  \[ R_C \leq 1 - S\left(\frac{1}{2} \cdot \frac{d_{\text{min}}}{n}\right) \] \hspace{1cm} (2.24b)

- **Plotkin bound:**
  \[ R_C \leq 1 - 2 \cdot \frac{d_{\text{min}}}{n} \] \hspace{1cm} (2.24c)

- **Elias bound:**
  \[ R_C \leq 1 - S\left(\frac{1}{2} \left(1 - \sqrt{1 - 2 \cdot \frac{d_{\text{min}}}{n}}\right)\right) \] \hspace{1cm} (2.24d)

- **Gilbert-Varshamov bound:**
  \[ R_C \geq 1 - S\left(\frac{d_{\text{min}}}{n}\right) \] \hspace{1cm} (2.24e)
The upper bounds form different limits in terms of the error protection \((d_{\text{min}}/n)\) at which a certain information content \((R_C)\) can be transmitted. The Gilbert-Varshamov bound forms the lower limit. Codes the values of which are lower than the lower limit have to be considered as bad codes, since in theory it is said that, based on a bad code, a better one with a higher code rate exists. These codes can then be considered as good codes.
2.6 Error Probabilities

2.6.1 Error Occurrence Probability

It is assumed that a BSC with the error probability $p_{\text{err}}$ is given. From this, a mean number of errors $n_{\text{Fehler}}$ per code word of $\langle n_{\text{Fehler}} \rangle = n \cdot p_{\text{err}}$ results. The occurrence probability of which is calculated as follows:

The probability that no error occurs is:

$$p(f = 0) = (1 - p_{\text{err}}) \cdot (1 - p_{\text{err}}) \cdot \ldots \cdot (1 - p_{\text{err}}) = (1 - p_{\text{err}})^n$$  \hspace{1cm} (2.25a)

The probability that a single error occurs at a certain point ($f = f_1$, $w_H(f_1) = 1$) is:

$$p(f = f_1) = p_{\text{err}} \cdot (1 - p_{\text{err}})^{n-1}$$  \hspace{1cm} (2.25b)

The probability that a single error occurs at an arbitrary point ($f = f_1$, $w_H(f_1) = 1$) is:

$$p(f = f_1) = n \cdot p_{\text{err}} \cdot (1 - p_{\text{err}})^{n-1}$$  \hspace{1cm} (2.25c)

The probability of $n_e$ errors at arbitrary points ($f = f_{n_e}$, $w_H(f_{n_e}) = n_e$) is:

$$p(f = f_{n_e}) = \binom{n}{n_e} \cdot p_{\text{err}}^{n_e} \cdot (1 - p_{\text{err}})^{n-n_e}$$  \hspace{1cm} (2.25d)
For the binomial coefficients (eq. 2.26) generally applies:
\[
\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \binom{n}{n-k} ; \quad \binom{n}{0} = \binom{n}{n} = 1
\] (2.26)

Using these, the binomial formula can be represented as:
\[
\sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} = (a + b)^n ; \quad \sum_{k=0}^{n} \binom{n}{k} = 2^n
\] (2.27)

For \( n = 7 \), the error occurrence probability according to bit errors is represented here.
2.6.2 Probability of Undetected Errors

Let $x_i$ and $x_j$ be code words; $x_i$ was sent and $y = x_j$ was received. The decoding rule will assume that no error has occurred and see $x_j$ as the code word that was transmitted. This special case of a decoding error cannot be avoided; the error remains undetected. For codes that shall detect the errors, the probability $p_{ue}$ (undetected error) that an error remains undetected is an important parameter calculated by:

$$p_{ue} = \sum_{i=0}^{N-1} p(x_i) \sum_{j=0 \atop j \neq i}^{N-1} p(x_j \mid x_i)$$

$$= \sum_{i=0}^{N-1} p(x_i) \sum_{j=0 \atop j \neq i}^{N-1} (1 - p_{err})^{n - d_H(x_j, x_i)} p_{err} d_H(x_j, x_i)$$

$$= \frac{1}{M} \sum_{i=0}^{N-1} \sum_{j=0 \atop j \neq i}^{N-1} (1 - p_{err})^{n - d_H(x_j, x_i)} p_{err} d_H(x_j, x_i)$$

(2.28a)
2.6.2 Probability of Undetected Errors

It is assumed that all code words have equal occurrence probability. The double sum in (eq. 2.28a) and the need for all possible Hamming distances are the main difficulty in terms of the calculation. For the linear binary codes discussed in the next chapter, it applies that the distribution of \( d(x_i, x_j) \) can be determined by the number \( A_i \) of code words with the weight \( i \).

\[
p_{ue} = \sum_{i=d_{\text{min}}}^{n} A_i \cdot (1 - p_{\text{err}})^{n-i} \cdot p_{\text{err}}^{i}
\]

(2.28b)

Using the equation

\[
A \left( \frac{p_{\text{err}}}{1 - p_{\text{err}}} \right) = 1 + \sum_{i=d_{\text{min}}}^{n} A_i \cdot p_{\text{err}}^{i} \cdot (1 - p_{\text{err}})^{-i}
\]

(2.29)

the following notation results:

\[
p_{ue} = (1 - p_{\text{err}})^{n} \left[ A \left( \frac{p_{\text{err}}}{1 - p_{\text{err}}} \right) - 1 \right]
\]

(2.30a)
2.6.2 Probability of Undetected Errors

For small error probabilities $p_{\text{err}}$, $p_{\text{ue}}$ can be calculated approximately by:

$$p_{\text{ue}} \approx \sum_{i=d_{\text{min}}}^{n} A_i \cdot p_{\text{err}}^i = A(p_{\text{err}}) - 1 \approx A_{d_{\text{min}}} \cdot p_{\text{err}}^{d_{\text{min}}}$$  \hspace{1cm} (2.30b)

Example:
Consider the $(7,4)$ Hamming code, which is a linear binary code. This code has one code word each with the weight 0 and the weight 7 and 7 code words each with the weights 3 and 4.
The minimum distance is 

$$d_{\text{min}} = 3.$$  

From this, a probability $p_{\text{ue}}$ results:

$$p_{\text{ue}} = 7 (1 - p_{\text{err}})^4 p_{\text{err}}^3 + 7 (1 - p_{\text{err}})^3 p_{\text{err}}^4 + p_{\text{err}}^7$$

$$= 7 p_{\text{err}}^3 - 21 p_{\text{err}}^4 + 21 p_{\text{err}}^5 - 7 p_{\text{err}}^6 + p_{\text{err}}^7 \approx 7 p_{\text{err}}^3$$
2.6.3 Residual-Error Probability

Let \( C = \{c_0, c_1, c_2, \ldots, c_{2^k-1}\} \) be a linear block code with the correction capability \( t \). Usage of a limited minimum distance decoding (BMD) is assumed. Let the error vector be referred to as \( f \). Then for the word error probability with BMD \( p_{w,BMD} \) the following applies:

\[
p_{w,BMD} = 1 - p(\text{correct decoding})
= 1 - p(w_H(f) \leq t) = p(w_H(f) \geq t + 1)
= 1 - \sum_{i=0}^{t} p(w_H(f) = i) = \sum_{i=t+1}^{n} p(w_H(f) = i)
\]

(2.31a)

With equation (2.25d), the probability of error vectors with the weight \( w_H(f) = i \) results

\[
p(w_H(f) = i) = \binom{n}{i} \cdot p_{err}^i \cdot (1 - p_{err})^{n-i}
\]

(2.31b)

From this, the following results:

\[
p_{W,BMD} = 1 - \sum_{i=0}^{t} \binom{n}{i} p_{err}^i (1 - p_{err})^{n-i} = \sum_{i=t+1}^{n} \binom{n}{i} p_{err}^i (1 - p_{err})^{n-i}
\]

(2.31c)
2.6.3 Residual-Error Propability

For small error probabilities, the following estimation applies:

\[ p_{W,BMD} < \binom{n}{t+1} p_{err}^{t+1} \]  

(2.31d)

Between the word error probabilities with MLD and BMD the following relation exists:

\[ p_{W,MLD} \leq p_{W,BMD} \]

Example: For the (7,4) Hamming code (see above) results:

\[
p_W = 1 - \binom{7}{0} p_{err}^0 (1 - p_{err})^7 - \binom{7}{1} p_{err}^1 (1 - p_{err})^6
\]

\[
= 1 - (1 - p_{err})^7 - 7 p_{err} (1 - p_{err})^6
\]

\[
= 1 - (1 - 7 p_{err} + 21 p_{err}^2 - p_{err}^3 \ldots) - 7 p_{err} (1 - 6 p_{err} + 2 p_{err}^2 \ldots)
\]

\[
\approx 21 p_{err}^2 = \binom{7}{2} p_{err}^2 \quad \text{equals sign, since the code is a perfect one}
\]
2.6.3 Residual-Error Probability

Eventually, the bit error probability $p_{\text{bit}}$ after decoding is to be estimated:

$$p_{\text{bit}} = \frac{1}{k} \sum_{i=0}^{n} \left\langle \text{no. of bit errors per decod. word} \mid w_{H}(\mathbf{f}) = i \right\rangle \cdot p(w_{H}(\mathbf{f}) = i) \quad (2.32)$$

The number of bit errors is limited to the number of information bits $k$ and to $i + t$, because:

$$d_{H}(\mathbf{u}, \hat{\mathbf{u}}) \leq d_{H}(\mathbf{x}, \hat{\mathbf{x}}) \leq d_{H}(\mathbf{x}, \mathbf{y}) + d_{H}(\mathbf{y}, \hat{\mathbf{x}}) \leq i \quad \text{triangle inequality}$$

$$\Rightarrow p_{\text{bit}} \leq \frac{1}{k} \sum_{i=t+1}^{n} \min(k, i + t) \cdot p(w_{H}(\mathbf{f}) = i) \quad (2.34)$$

And finally:

$$p_{\text{bit}} \leq \sum_{i=t+1}^{n} \min \left(1, \frac{i + t}{k}\right) \cdot \binom{n}{i} \cdot p_{\text{err}}^{i} \cdot (1 - p_{\text{err}})^{n-i} \quad (2.35a)$$

For small error probabilities the estimate applies:

$$p_{\text{bit}} \leq \min \left(1, \frac{d_{\text{min}}}{k}\right) \cdot \binom{n}{t + 1} \cdot p_{\text{err}}^{t+1} \quad (2.35b)$$
3 Single-Error Correcting Block Codes

In this chapter, fundamental knowledge of linear block coding is imparted. Besides the mathematical dependencies and their properties, two simple code examples are considered which are of great importance: the general cyclic codes and the Hamming codes. Furthermore, practical approaches for the realisation of such codes by means of shift register circuits are considered.

Based on a block code with the number of levels \( q \), the number of information bits \( k \), the number of code word bits \( n \) and the minimum distance \( d_{\min} \), the general term reads: \( (n,k,d_{\min})_q \) block code

In this chapter, only binary codes are considered \( (q = 2) \). Therefore, the index is skipped. With most terms, instead of the minimum distance the number of control bits \( m \) are used or they are skipped as well:

\( (n,k,m) \) block code and \( (n,k) \) block code respectively
An essential indicator is the classification into **systematic / unsystematic codes**.

With systematic codes, information bits and control bits can be separated from each other:

\[ \mathbf{c} = (\mathbf{u}, \mathbf{p}) \]  

(3.1)

\[
\begin{array}{cccccccc}
  u_0 & u_1 & u_2 & u_3 & \ldots & \ldots & u_{k-1} \\
  \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
  c_0 & c_1 & c_2 & c_3 & \ldots & \ldots & c_{k-1} & c_k & c_{k+1} & \ldots & \ldots & c_{n-1} \\
\end{array}
\]

By contrast, information bits and control bits cannot be separated by non-systematic codes. However, block codes can always be converted (i.e. by transposing of bits) to equivalent systematic codes.

\[ m = n - k \]  

(3.2)
3.1 Mathematical Principles

Calculating in the binary number system (modulo 2):

<table>
<thead>
<tr>
<th>Addition:</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multiplication:</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In an equation, a modulo calculation is often clarified explicitly by stating it, e.g.: \(1 + 1 = 0 \mod 2\).

Mathematically, this number system with its calculation rules represents the simplest example of a field \(K\) (chapter 4). Practically, they can be realised by an XOR and an AND relation, respectively.

Binary words form so-called \(n\)-tuples of elements of \(K^n\) (space over all \(n\)-digit bit sequences). The addition in \(K^n\) is effected component by component,

example from \(K^4\): \(0011 + 0101 = 0110\),

whereas the multiplication by elements of \(K\) is simply defined as:

\(0b = 0\) and \(1b = b\) for all \(b \in K^n\).
3.1 Mathematical Principles

Representation of block codes as a matrix:
Let \( u \) be an information vector of the length \( k \) and let \( x \) be the respective code word vector of a block code of the length \( n \). Then \( u \) and \( x \) are related by

\[
x = u \cdot G,
\]

where \( G \) represents the generator matrix \((k \times n) - \text{matrix}\) of the block codes.

If a code can be described by this matrix multiplication for any \( u \), it is **linear**.
The rows of the generator matrix have to be linearly independent.

For systematical block codes, \( G \) has the form

\[
G = [I_k \ P]
\]

with the \( k \times k \) - unit matrix \( I_k \) and the \( k \times (n - k) \) – control bit matrix \( P \).
Polynomial representation of block codes:
An n-digit code word \( c \) can be represented as a polynomial \( c(x) \) as follows:

\[
\begin{align*}
\mathbf{c} &= (c_0 \ c_1 \ ... \ c_{n-1}) & \Leftrightarrow & & c(x) = \sum_{i=0}^{n-1} c_i \ x^i
\end{align*}
\] (3.5)

Examples:

- \( \mathbf{c}_1 = 0011 \) \( \Leftrightarrow \) \( c_1(x) = x^2 + x^3 \);
- \( \mathbf{c}_2 = 0101 \) \( \Leftrightarrow \) \( c_2(x) = x + x^3 \);

(see also table for further polynomial examples of code words of the length \( n \)).

The addition of vectors corresponds to the addition of polynomials.

Example: \( \mathbf{c}_1 + \mathbf{c}_2 = 0110 \)

\( \Leftrightarrow c_1(x) + c_2(x) = x^2 + x^3 + x + x^3 = x + x^2 \).
3.1 Mathematical Principles

Weighting function:
Let $C$ be a block code. Then let $A_i$ be the number of code words of $C$ that have the weight $i$ $(0 \leq i \leq n)$. Then let be the weighting function $A(z)$ and $W(x,y)$, respectively:

\[
A(z) = \sum_{i=0}^{n} A_i z^i \quad (3.6a) \quad W(x, y) = \sum_{i=0}^{n} A_i x^{n-i} y^i \quad (3.7a)
\]

Between these two notations the following relations exist:

\[
A(z) = W(1, z) \quad (3.6b) \quad W(x, y) = x^n A(y) \quad (3.7b)
\]

The weighting function can be calculated in closed form only for a small number of codes.
Some codes have a symmetrical weight distribution: $A_i = A_{n-i}$.

The weight distribution makes the accurate calculation of the residual-error probability possible (see chapter 2).
For the weight distribution of a linear code with the minimum distance $d_{\text{min}}$ applies:

\[
A_0 = 1; \quad A_n \leq 1 \text{ and } A_i = 0 \text{ for } 0 < i <
\]
### 3.1 Mathematical Principles

#### Example: (4,3) parity check

<table>
<thead>
<tr>
<th>Code word</th>
<th>Information</th>
<th>Control bit</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{c}_0 )</td>
<td>000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \mathbf{c}_1 )</td>
<td>001</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( \mathbf{c}_2 )</td>
<td>010</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( \mathbf{c}_3 )</td>
<td>011</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>( \mathbf{c}_4 )</td>
<td>100</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( \mathbf{c}_5 )</td>
<td>101</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>( \mathbf{c}_6 )</td>
<td>110</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>( \mathbf{c}_7 )</td>
<td>111</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

The weight distribution from the table results in a weighting function:

\[
A(z) = 1 + 6z^2 + z^4
\]
3.2 Terms and Features of Linear Block Codes

For a linear block code \( C \), which is described by equation (3.3), the following applies:

- Every code word is a linear combination of rows of the generator matrix \( G \);
- The code is composed of all linear combinations of rows of \( G \);
- The sum of code words results in a code word and
- in the code the zero vector \((0 \ 0 \ 0 \ldots \ 0)\) is included.

Fundamental row operations in \( G \) do not change a code, that means the code space \( C \) remains constant:

- Permutation of two rows,
- Multiplication of a row by a scalar unequal 0 and
- Addition of a row to another one.

However, the assignment of the information words \( u \) to the code words \( x \) changes.
3.2 Terms and Features of Linear Block Codes

Parity-check matrix:
For a linear block code $C$, a parity-check matrix $H$ is defined by:

$$c H^T = 0 \quad \text{for all } c \in C \quad \text{and} \quad x H^T \neq 0 \quad \text{for all } x \notin C \quad (3.8a/b)$$

$H$ is a $(n - k) \times n$ matrix. It applies:

$$0 = c H^T = (u G) H^T = u (G H^T) \quad \rightarrow \quad G H^T = 0 \quad (3.9)$$

That means that the generator matrix $G$ and the parity-check matrix $H$ are orthogonal.

As for $G$, also for $H$ fundamental row operations are allowed.

If $C$ is a systematic code with $G = (I_k P)$, for $H$ applies:

$$H = (P^T I_{n-k}) \quad (3.10)$$

For this case, orthogonality can be proven as follows:

$$G H^T = \begin{pmatrix} I_k & P \end{pmatrix} \begin{pmatrix} P \\ I_{n-k} \end{pmatrix} = I_k P + P I_{n-k} = P + P = 0$$
3.2 Terms and Features of Linear Block Codes

Syndrome:
Let $H$ be the parity-check matrix of a linear $(n,k,m)$ block code and $x \in C$ any code word to be sent. With the transmission, an error word $f$ is superposed so that the receive word $y$ reads:

$$y = x + f. \quad (3.11)$$

For the product of $y$ and $HA^T$ then applies:

$$yH^T = (x + f)H^T = xH^T + fH^T = 0 + fH^T = fH^T = s \quad (3.12)$$

The product $yH^T$ is referred to as syndrome $s$, the represented calculation is the calculation of the syndrome in matrix notation.

The features of $s$ are:
- $s$ is a zero vector only then if $y$ is a code word,
- All those error vectors $f$ are recognised that are no code words and
- $s$ is independent from the code word.
3.2 Terms and Features of Linear Block Codes

Dual code:
Based on a code $\mathbf{C}$ with the generator matrix $\mathbf{G}$ and the parity-check matrix $\mathbf{H}$, let a dual code $\mathbf{C}_d$, which is described by its dual generator matrix $\mathbf{G}_d$ and its dual parity-check matrix $\mathbf{H}_d$, be defined as:

$$\mathbf{G}_d = \mathbf{H} \quad \text{and} \quad \mathbf{H}_d = \mathbf{G} \quad (3.13a/b)$$

The code words of the two codes are orthogonal. Using $\mathbf{c} = \mathbf{u} \mathbf{G} \in \mathbf{C}$ and $\mathbf{c}_d = \mathbf{v} \mathbf{H} \in \mathbf{C}_d$, the following applies:

$$\mathbf{c} \mathbf{c}_d^T = (\mathbf{u} \mathbf{G})(\mathbf{v} \mathbf{H})^T = \mathbf{u} \mathbf{G} \mathbf{H}^T \mathbf{v}^T = \mathbf{u} \mathbf{0} \mathbf{v}^T = \mathbf{0}$$

Examples for codes being dual to each other are the repetition code (index W) and the parity check where the first digit is the control bit (index P):

$$\mathbf{G}_W = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \end{pmatrix} = \mathbf{H}_p \quad \mathbf{H}_W = \begin{pmatrix} 1 & 1 & 0 & \ldots & 0 \\ 1 & 0 & 1 & 0 & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 1 & 0 & \ldots & 0 & 1 \end{pmatrix} = \mathbf{G}_p.$$
3.2 Terms and Features of Linear Block Codes

Weighting function of the dual code (MacWilliams identity):
If \( A(z) \) is the weighting function of an \((n,k)\) block code, then the weighting function \( A_d(z) \) of the corresponding dual code is:

\[
A_d(z) = 2^{-k} \cdot (1 + z)^n \cdot A\left(\frac{1-z}{1+z}\right)
\]
\[
W_d(x, y) = 2^{-k} \cdot W(x + y, x - y)
\]

\[
A(z) = 2^{-(n-k)} \cdot (1 + z)^n \cdot A_d\left(\frac{1-z}{1+z}\right)
\]
\[
W(x, y) = 2^{-(n-k)} \cdot W_d(x + y, x - y)
\]

Example: For the \((3,2)\) parity check, the code \( \mathbf{C} \) is:

\[
\mathbf{C} = \{000,011,101,110\}.
\]

The weighting function reads as follows:

\[
A(z) = 1 + 3z^2.
\]

The corresponding dual code reads: \( \mathbf{C_d} = \{000,111\} \) (repetition code).

The weighting function is calculated to be:

\[
A_d(z) = 2^{-2} \cdot (1 + z)^3 \cdot \left[ 1 + 3\left(\frac{1-z}{1+z}\right)^2 \right] = 1 + z^3
\]
3.2 Terms and Features of Linear Block Codes

Modifications of linear codes:
An \((n,k,d_{\text{min}})_q\) code can be modified to an \((n',k',d_{\text{min}}')_q\) code as follows:

- **Extending**: appending additional control bits
  \[n' > n, \; k' = k, \; m' > m, \; R_C' < R_C, \; d_{\text{min}}' \geq d_{\text{min}}\]
- **Puncturing**: reduction of control bits
  \[n' < n, \; k' = k, \; m' < m, \; R_C' > R_C, \; d_{\text{min}}' \leq d_{\text{min}}\]
- **Lengthening**: appending additional information bits
  \[n' > n, \; k' > k, \; m' = m, \; R_C' > R_C, \; d_{\text{min}}' \leq d_{\text{min}}\]
- **Shortening**: reduction of information bits
  \[n' < n, \; k' < k, \; m' = m, \; R_C' < R_C, \; d_{\text{min}}' \geq d_{\text{min}}\]

Adding a parity check bit, the extension of a code word with an uneven minimum distance can result in an augmentation of the minimum distance by 1, so that the code can detect a further error (example: see Hamming codes).
### 3.3 Cyclic Codes

**Definition of cyclic codes:**

A linear \((n,k)\) block code is called cyclic, if every cyclic shift of a code word results in a code word:

\[
(c_0, c_1, c_2, \ldots, c_{n-1}) \in C \Rightarrow (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in C
\]  

(3.16)

The set of cyclic codes is a subset of the linear block codes. Accordingly, between an information word

\[
u = (u_0, u_1, u_2, \ldots, u_{k-1})
\]

and a code word

\[
c = (c_0, c_1, c_2, \ldots, c_{n-1})
\]

the following relation exists

\[
c = u \cdot G.
\]

(3.17)

Thus, a cyclic code is also described by a generator matrix \(G\). Thereby, all cyclic codes have at least a generator matrix with band structure (see equation (3.18) and the following example).

---

3 Single-Error Correcting Block Codes

---
3.3 Cyclic Codes

Example of a cyclic code:

\[
G = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1
\end{pmatrix}
\]

\[
\begin{array}{c|c}
\text{u} & \text{x} \\
0000 & 0000000 \\
1000 & 1101000 \\
0100 & 0110100 \\
0010 & 0011010 \\
0001 & 0001101 \\
1110 & 1000110 \\
0111 & 0100011 \\
1101 & 1010001 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{u} & \text{x} \\
1011 & 1111111 \\
1010 & 1110010 \\
0101 & 0111001 \\
1100 & 1011100 \\
0110 & 0101110 \\
1111 & 1001011 \\
1001 & 1100101 \\
\end{array}
\]

Due to the cyclic feature, the code can easily be derived. In this case it is sufficient to know the code words \{0000000, 1101000, 1110010, 1111111\}. At first, however, these have to be found (using linear combinations of rows of \(G\) + zero word). The remaining words result from cyclic shift. Stop criterion: \(2^k\) words found.
3.3 Cyclic Codes

Generator matrix with band structure:

\[
G = \begin{bmatrix}
g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & 0 & \cdots & 0 \\
g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & 0 & \cdots & 0 \\
g_0 & 0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 \\
0 & 0 & \cdots & 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} \\
\end{bmatrix}
\]

(3.18)

Caution: Generator matrices with band structure do not necessarily describe cyclic codes!

Example: \( C = \{00000, 11010, 01101, 10111\} \) with the generator matrix

\[
G = \begin{bmatrix}
1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
\end{bmatrix}
\]

is not a cyclic code.
Generator polynomial:
The generator matrix $G$ of a cyclic code is thus characterised completely by a row of $G$. It includes at most $n - k + 1$ of 0 different entries: $g_0, g_1, g_2, \ldots, g_{n-k}$. These coefficients provide a special polynomial:

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{n-k} x^{n-k} \quad (3.19a)$$

g$(x)$ is called generator polynomial.

With binary codes, the coefficients of the generator polynomial can only take values of 0 or 1. For an $(n,k)$ block code, the following must continue to apply:

$$g_0 = g_{n-k} = 1$$

$$\Rightarrow g(x) = 1 + g_1 x + g_2 x^2 + \ldots + x^{n-k} \quad (3.19b)$$

Using polynomials, for cyclic codes a simplified description of generator matrix, message and code vectors results.
3.3 Cyclic Codes

The polynomial and the matrix notation are listed below side by side:

**Information words:**
\[ \mathbf{u} = (u_0, u_1, u_2, \ldots, u_{k-1}) \]
\[ u(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_{k-1} x^{k-1} \quad (3.20a/b) \]

**Code words:**
\[ \mathbf{c} = (c_0, c_1, c_2, \ldots, c_{n-1}) \]
\[ c(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1} \quad (3.20c/d) \]

**Coding rules:**
\[ \mathbf{c} = \mathbf{u} \mathbf{G} \]
\[ c(x) = u(x) \cdot g(x) \quad (3.20e/f) \]

With the equations (3.18/19b), both rules provide the same results:

\[ c_0 = u_0 g_0 \]
\[ c_1 = u_0 g_1 + u_1 g_0 \]
\[ c_2 = u_0 g_2 + u_1 g_1 + u_2 g_0 \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]

Consequently, the general presentation of a code word polynomial is:
\[ c(x) = (u_0 + u_1 x + u_2 x^2 + \ldots + u_{k-1} x^{k-1}) \cdot g(x) \]

\[ \Rightarrow g(x) \text{ is the code word polynomial with the smallest degree greater than zero.} \]
Period of a generator polynomial:
Let \( g(x) \) be the generator polynomial of a cyclic code \( C \). And let \( r \) be the smallest exponent for which applies
\[
x^r + 1 = 0 \mod g(x).
\] (3.21)
g(\( x \)) is then the divisor of \( x^r + 1 \), and \( r \) is the period of \( g(x) \).
A cyclic feature only exists if for the code word length \( n \) the following applies:
\[
n = r
\] (3.22)
For the proof the following theorem is applied, which is quite plausible using equation (3.20f):

**Theorem:** Every code word polynomial \( c(x) \) is divisible by \( g(x) \) without remainder.

**Example:**
\[
\begin{align*}
g(x) &= 1 + x + x^3, \quad u(x) = 1 + x^2 + x^3 \\
\Rightarrow \quad c(x) &= u(x) \cdot g(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6 \iff c = (1111111)
\end{align*}
\]
3.3 Cyclic Codes

Proof of the condition (3.22): It shall apply that
\[ c = (c_0, c_1, c_2, \ldots, c_{n-1}) \in C \Rightarrow d = (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in C. \]

\[ c(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1} \] is divisible by \( g(x) \) without remainder:
\[ c(x) = 0 \mod g(x) \Rightarrow x \cdot c(x) = 0 \mod g(x) \]

\[ d(x) = c_{n-1} + c_0 x + c_1 x^2 + \ldots + c_{n-2} x^{n-1} \]
\[ = c_{n-1} + x \left( c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1} \right) + c_{n-1} x^n \]
\[ = c(x) \]
\[ = x \cdot c(x) + c_{n-1} (1 + x^n) \]

\( d \) is a code word only then, if \( d(x) \) is divisible by \( g(x) \) without remainder. Therefore, the following has to apply:
\[ 1 + x^n = 0 \mod g(x). \] (3.23)

Only if the generator polynomial fulfills condition (3.23), a cyclic code results.
3.3 Cyclic Codes

Determination of a generator polynomial:
g(x) is the divisor of 1 + x^n. In general, that means that 1 + x^n can be split into single coefficients. Every coefficient can then be applied as a generator polynomial.

Example: \( x^7 + 1 = (1 + x) \cdot (1 + x + x^3) \cdot (1 + x^2 + x^3) \)

Calculation of the period of a given generator polynomial: This is done by reversion of the Gaussian Division Algorithm. The polynomial is shifted that way that always the lowest powers cancel out each other.

Example: \((x^3 + x + 1) \cdot p(x) = x^n + 1\)

\[
\begin{array}{c|c|c|c|c|c|c|c|c}
& x^7 & + & 0 & + & 0 & + & 0 & + & 1 \\
\hline
x^3 & + & 0 & + & x & + & 1 & \cdot 1 \\
\hline
x^4 & + & 0 & + & x^2 & + & x & \cdot x \\
\hline
x^5 & + & 0 & + & x^3 & + & x^2 & \cdot x^2 \\
\hline
x^7 & + & 0 & + & x^5 & + & x^4 & \cdot x^4 \\
\hline
\end{array}
\]

\( \Rightarrow p(x) = x^4 + x^2 + x + 1 \)

\( \Rightarrow h(x) = x^4 + x^2 + x + 1 \)

\( \Rightarrow n = 7 \)
3.3 Cyclic Codes

Check matrix and check polynomial:
For the generator polynomial \( g(x) \) of a cyclic \((n,k)\) block code applies:
\[
x^n + 1 = g(x) \cdot h(x).
\]
(3.24)
\( h(x) \) is the check polynomial and has the degree \( k \):
\[
h(x) = h_0 + h_1 x + h_2 x^2 + \ldots + h_k x^k
\]
with \( h_0 = h_k = 1 \).

The respective \((n - k) \times n\) parity-check matrix \( H \) is composed as follows:
\[
H = \begin{pmatrix}
  h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & 0 & 0 & \cdots & 0 \\
  0 & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\
  0 & 0 & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & \cdots & 0 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & 0 & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 \\
  0 & 0 & \cdots & 0 & 0 & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \\
\end{pmatrix}
\]  
(3.25b)
Systematic Coding:
In general, cyclic codes are not systematic. In order to achieve this, the following construction rule is used:

1. Multiplication of the message polynomial $u(x)$ by $x^{n-k}$:
   \[ u(x) \cdot x^{n-k} = u_0 x^{n-k} + u_1 x^{n-k+1} + u_2 x^{n-k+2} + \ldots + u_{k-1} x^{n-1} \]  
   (3.26a)
   (corresponds to the shift of the message digits to the highest coefficients).

2. Division of the polynomial $u(x) \cdot x^{n-k}$ by $g(x)$:
   \[ \frac{u(x) \cdot x^{n-k}}{g(x)} = q(x) + \frac{r(x)}{g(x)} \]  
   that means, $u(x) \cdot x^{n-k} = q(x) \cdot g(x) + r(x)$ with $q(x)$ as whole-number division result and $r(x)$ as division remainder (in general: $r(x) \neq 0$ and $r(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{n-k-1} x^{n-k-1}$).

3. Whole-number divisibility is enforced by addition of $r(x)$:
   \[ u(x) \cdot x^{n-k} + r(x) = q(x) \cdot g(x) = c(x). \]  
   (3.26b)  
   (3.27)

Resulting code word: $c = (r_0, r_1, r_2, \ldots, r_{n-k-1}, u_0, u_1, u_2, \ldots, u_{k-1})$
Example:
Using the cyclic (7,4) block code with the generator matrix
\[ g(x) = x^3 + x + 1, \]
for the information word
\[ u = (1 \ 0 \ 0 \ 1), \text{ d. h. } u(x) = x^3 + 1, \]
a systematic code word is to be developed.

1. For \( n = 7 \) and \( k = 4 \), \( n - k = 3 \). From this results: \( u(x) \cdot x^{n-k} = x^6 + x^3 \).

2. Division of the polynomial \( u(x) \cdot x^{n-k} \) by \( g(x) \):
   \[
   \begin{align*}
   (x^6 + 0 + 0 + 0 + x^3) : (x^3 + 0 + x + 1) &= x^3 + x + \frac{x^2 + x}{x^3 + x + 1} \\
   + (x^6 + 0 + x^4 + x^3) &
   \end{align*}
   \]

3. \( c(x) = u(x) \cdot x^{n-k} + r(x) = x^6 + x^3 + x^2 + x \quad \Rightarrow \quad c = (0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1) \)
Decoding of cyclic codes:
The receive vector \( y \) and the receive polynomial \( y(x) \), respectively, are to be checked

\[
y = c + f \iff y(x) = c(x) + f(x).
\]  \hspace{1cm} (3.28)

It applies that a code word \( c \) without remainder is divisible by the generator polynomial \( g(x) \). The result is the original information word \( u \), which can be taken out again in this way. For any \( y \), this method is a possibility to check whether or not it is about a code word. In case of error, with the division a remainder is left:

\[
\frac{y(x)}{g(x)} = q(x) + \frac{s(x)}{g(x)}
\]  \hspace{1cm} (3.29a)

\( s(x) \) represents the syndrome in polynomial notation:

\[
y(x) = q(x) \cdot g(x) + s(x).
\]  \hspace{1cm} (3.29b)

If \( s(x) \neq 0 \), the error correction takes place by means of a syndrome table. The polynomial division is practically realised by shift register circuits. It is a great advantage that with the coding and decoding no matrix multiplications are required.
3.4 Hamming Codes

Definition of the Hamming codes:
Let be $m \geq 3$. A Hamming code is a code with code words of the length $n = 2^m - 1$, consisting of $k = 2^m - m - 1$ information bits and $m$ check bits. The minimum distance is 3.

Thus, the Hamming code is a $(2^m - 1, 2^m - 1 - m)$ block code. For the code, inter alia, the following values for $n$, $k$ and $m$ result:

<table>
<thead>
<tr>
<th>$m$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>7</td>
<td>15</td>
<td>31</td>
<td>63</td>
<td>127</td>
<td>255</td>
</tr>
<tr>
<td>$k$</td>
<td>4</td>
<td>11</td>
<td>26</td>
<td>57</td>
<td>120</td>
<td>247</td>
</tr>
</tbody>
</table>

Due to its minimum distance, the Hamming code can correct exactly one error. In reality, it is constructed for this case.
3.4 Hamming Codes

Derivation of the construction rule for Hamming codes:

Aim: construction of a code that can correct exactly one error.

The syndrome $s$ of the length $m$ only depends on the error vector $f$ of the length $n$ (eq. (3.12)). A single error at the position $i$ ($f_i = 1; 0 \leq i \leq n - 1$) results in a syndrome complying with the respective row of $H^T$. In order to differentiate all single-error positions $i$ unambiguously, all rows of $H^T$ have to differ from each other; none of these rows may be the zero vector.

All in all, there are $2^m - 1$ different rows without the zero vector, thus $n = 2^m - 1$. From $n = k + m$ results $k = 2^m - m - 1$.

Virtually, the rows of $H^T$ and the columns of $H$, respectively, are formed by all kinds of sequences except the zero vector.

In doing so, the $m = n - k$ control bits address the $2^m - 1$ positions.
Since it is irrelevant which syndrome describes an error position, the bit sequences may be randomised. In doing so, the parity-check matrix can be arranged in that way that the resulting Hamming code is systematic:

\[
H = (P^T I_{n-k}) = \begin{bmatrix}
1 & 0 & \cdots & \cdots & | & 1 & 0 & \cdots & 0 \\
1 & 1 & \cdots & \cdots & | & 0 & 1 & 0 & \vdots \\
\vdots & \vdots & \cdots & \cdots & | & \vdots & 0 & \ddots & 0 \\
1 & 1 & \cdots & \cdots & | & 0 & \cdots & 0 & 1
\end{bmatrix}
\]

(3.30)

all possible column vectors with more than one 1

The generator matrix can then be derived easily using equation (3.4).
Example: For the (7,4) Hamming code the following systematic notation results:

\[
H = (P^T I_{n-k}) = \begin{bmatrix}
1 & 1 & 1 & 0 & | & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & | & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & | & 0 & 0 & 1
\end{bmatrix} \quad \Rightarrow \quad P = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 1
\end{bmatrix}
\]

With this, the following systematics with the respective equations for the calculation of the control bits results:

\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 & | & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & | & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & | & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & | & 0 & 1 & 1
\end{bmatrix}
\]

control bits:

\[
\begin{align*}
c_4 &= c_0 + c_1 + c_2 \\
c_5 &= c_0 + c_1 + c_3 \\
c_6 &= c_0 + c_2 + c_3
\end{align*}
\]
### Example (continuation):

**Table of code words of the (7,4) Hamming codes:**

<table>
<thead>
<tr>
<th>Code word</th>
<th>Information</th>
<th>Control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_0$</td>
<td>0000</td>
<td>000</td>
</tr>
<tr>
<td>$c_1$</td>
<td>0001</td>
<td>011</td>
</tr>
<tr>
<td>$c_2$</td>
<td>0010</td>
<td>101</td>
</tr>
<tr>
<td>$c_3$</td>
<td>0011</td>
<td>110</td>
</tr>
<tr>
<td>$c_4$</td>
<td>0100</td>
<td>110</td>
</tr>
<tr>
<td>$c_5$</td>
<td>0101</td>
<td>101</td>
</tr>
<tr>
<td>$c_6$</td>
<td>0110</td>
<td>011</td>
</tr>
<tr>
<td>$c_7$</td>
<td>0111</td>
<td>000</td>
</tr>
<tr>
<td>$c_8$</td>
<td>1000</td>
<td>111</td>
</tr>
<tr>
<td>$c_9$</td>
<td>1001</td>
<td>100</td>
</tr>
<tr>
<td>$c_{10}$</td>
<td>1010</td>
<td>010</td>
</tr>
<tr>
<td>$c_{11}$</td>
<td>1011</td>
<td>001</td>
</tr>
<tr>
<td>$c_{12}$</td>
<td>1100</td>
<td>001</td>
</tr>
<tr>
<td>$c_{13}$</td>
<td>1101</td>
<td>010</td>
</tr>
<tr>
<td>$c_{14}$</td>
<td>1110</td>
<td>100</td>
</tr>
<tr>
<td>$c_{15}$</td>
<td>1111</td>
<td>111</td>
</tr>
</tbody>
</table>
3.4 Hamming Codes

Error correction with Hamming codes:

Example (continuation): error correction using the (7,4) Hamming code

The respective syndrome reads in general: \( s = (s_0 \ s_1 \ s_2) \).

From the parity-check matrix, the check equations result:

\[
\begin{align*}
    s_0 &= y_0 + y_1 + y_2 + y_4 \\
    s_1 &= y_0 + y_1 + y_3 + y_5 \\
    s_2 &= y_0 + y_2 + y_3 + y_6
\end{align*}
\]

These are analysed according to the table of syndromes.

<table>
<thead>
<tr>
<th>Error position</th>
<th>Syndrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>No error</td>
<td>0 0 0</td>
</tr>
<tr>
<td>Error in 0th pos.</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Error in 1st pos.</td>
<td>1 1 0</td>
</tr>
<tr>
<td>Error in 2nd pos.</td>
<td>1 0 1</td>
</tr>
<tr>
<td>Error in 3rd pos.</td>
<td>0 1 1</td>
</tr>
<tr>
<td>Error in 4th pos.</td>
<td>1 0 0</td>
</tr>
<tr>
<td>Error in 5th pos.</td>
<td>0 1 0</td>
</tr>
<tr>
<td>Error in 6th pos.</td>
<td>0 0 1</td>
</tr>
</tbody>
</table>

The decoding steps are then:
1. Evaluation of the check equations \( \rightarrow \) syndrome;
2. Determination of the error position from the table of syndromes;
3. Realisation of the error correction: add „1“ at the position of error.
3.4 Hamming Codes

Minimum distance of general Hamming codes:
All columns of the check matrix are different from each other. With this, two columns are linearly independent at any time. However, since all kinds of bit combinations exist, three columns are linearly independent at any time, e. g.:

100000..., 010000..., und 110000....

With this, the minimum distance is always

\[ d_{\text{min}} = 3 \quad (3.31a) \]

and the number of corrigible errors

\[ t = 1. \quad (3.31b) \]

Weighting function of general Hamming codes:

\[
A(z) = \frac{1}{n+1} \left[ (1+z)^n + n \cdot (1+z)^{\frac{n-1}{2}} \cdot (1-z)^{\frac{n+1}{2}} \right] \quad (3.32)
\]
3.4 Hamming Codes

Word error probability and probability of undetected errors of Hamming codes:

On pages 109 and 111 both probabilities for a (7,4) Hamming code have already been calculated. For $n$ equal to 3, 7, 15, 31, 63 and 127 they are depicted in the adjoining diagram.
Extended Hamming code:
Through evaluation of the syndrome, one error is corrigible. With a minimum distance of 3, however, also two errors are observable which result in \( s \neq 0 \) as well. However, it cannot be distinguished whether one or two errors have occurred.

The extended Hamming code differentiates between the 1-error and 2-error situation by using an additional control bit. Therefore, it is a \((2^m, 2^m - 1 - m)\) block code. The respective generator matrix of the extended Hamming code reads as follows:

\[
\mathbf{G}_{H,\text{ext}} = \begin{bmatrix}
\mathbf{G}_H & 1 \\
1 & 1 & \ddots & \vdots \\
& & 1 & 1
\end{bmatrix}
\] (3.33)
3.4 Hamming Codes

Possible error events are then:

- no error: \( s = 0; \)
- one error: \( s \neq 0, s_{m+1} = 1; \)
- two errors: \( s \neq 0, s_{m+1} = 0. \)

Decoding is then carried out according to the following rules:

- \( s = 0: \)
  receive vector = code word

- \( s \neq 0, s_{m+1} = 1: \)
  uneven number of errors ⇒ one error accepted and corrected by syndrome evaluation

- \( s \neq 0, s_{m+1} = 0: \)
  even number of errors ⇒ errors cannot be corrected.
3.5 Realisation of Coding and Decoding by Shift Registers

Basic circuits:

Shift registers consist of memory cells, each delaying by one pulse. The content of the memory cell $j$ at the time $i$ is referred to as $S_j(i)$. With this, the following applies:

$$S_{j+1}(i+1) = S_j(i) \quad (3.34)$$
3.5 Realisation of Coding and Decoding Using Shift Register Circuits

Cyclic shifting of a polynomial:
Shift register circuits are an excellent possibility to realise cyclic coding. The mathematical operations required for this can all be carried out using this technique. One of these is the cyclic shifting of a polynomial:

\[ x^i \cdot a(x) \mod (x^n + 1) \tag{3.35} \]

With this circuit, \[ x^i \cdot a(x) \mod (x^n + 1) \] is calculated, forming the simplest example of a regenerated shift register.
Multiplication of two polynomials:
An equation of the following form is to be calculated
\[ c(x) = a(x) \cdot [b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n]. \] (3.36)
3.5 Realisation of Coding and Decoding Using Shift Register Circuits

Example of a multiplication:
The following is to be calculated
\[ c(x) = a(x) \cdot b(x) = (x^4 + x^2 + x + 1) \cdot (x^3 + x^2 + 1) = x^7 + x^6 + x^5 + x^4 + 0 + 0 + x + 1 \]

\[
\begin{array}{cccccc}
\hline
i & s_0 & s_1 & s_2 & s_3 & c_i \\
\hline
-1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 \\
2 & 1 & 1 & 1 & 0 & 0 \\
3 & 0 & 1 & 1 & 1 & 0 \\
4 & 1 & 0 & 1 & 1 & 1 \\
5 & 0 & 1 & 0 & 1 & 1 \\
6 & 0 & 0 & 1 & 0 & 1 \\
7 & 0 & 0 & 0 & 1 & 1 \\
8 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
\]

\[ c(x) = a(x) \cdot b(x) = (x^4 + x^2 + x + 1) \cdot (x^3 + x^2 + 1) = x^7 + x^6 + x^5 + x^4 + 0 + 0 + x + 1 \]
3.5 Realisation of Coding and Decoding Using Shift Register Circuits

Division of two polynomials:

\[ q(x^{-1}) = a(x^{-1}) x^m + q(x^{-1}) \left[ b_0 x^m + b_1 x^{m-1} + \ldots + b_{m-1} x \right] \]
\[ \Rightarrow a(x^{-1}) = q(x^{-1}) \left[ b_0 + b_1 x^{-1} + \ldots + b_{m-1} x^{-(m-1)} + x^{-m} \right] = q(x^{-1}) b(x^{-1}) \]

Substitution \( x^{-1} \rightarrow x \) \( \Rightarrow a(x) = q(x) b(x) \) \( \Rightarrow q(x) = \frac{a(x)}{b(x)} \) (3.37a)

Division of two polynomials with remainder \( \frac{a(x)}{b(x)} = q(x) + \frac{s(x)}{b(x)} \) (3.37b)

is discontinued after having written the last digit of \( a(x^{-1}) \) in the shift register. Then \( q(x^{-1}) \) is the initial word and \( s(x^{-1}) \) the content of the memory cells.
3.5 Realisation of Coding and Decoding Using Shift Register Circuits

Example of a division of two polynomials with remainder:

\[ a(x^{-1}) = 000100001 \]
\[ q(x^{-1}) = 001110000 \]

\[ a(x^{-1}) : q(x^{-1}) = (x^8 + x^3) : (x^4 + x^3 + x + 1) \]
\[ = x^4 + x^3 + x^2 + \frac{x^3 + x^2}{x^4 + x^3 + x + 1} \]

<table>
<thead>
<tr>
<th>( i )</th>
<th>( a_i )</th>
<th>( s_0 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
<th>( q_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1+0=1</td>
<td>0+0=0</td>
<td>0</td>
<td>0+0=0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0+0=0</td>
<td>1+0=1</td>
<td>0</td>
<td>0+0=0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0+0=0</td>
<td>0+0=0</td>
<td>1</td>
<td>0+0=0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0+0=0</td>
<td>0+0=0</td>
<td>0</td>
<td>1+0=1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0+1=1</td>
<td>0+1=1</td>
<td>0</td>
<td>0+1=1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1+1=0</td>
<td>1+1=0</td>
<td>1</td>
<td>0+1=1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0+1=1</td>
<td>0+1=1</td>
<td>0</td>
<td>1+1=0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0+0=0</td>
<td>1+0=1</td>
<td>1</td>
<td>0+0=0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0+0=0</td>
<td>0+0=0</td>
<td>1</td>
<td>1+0=1</td>
<td>0</td>
</tr>
</tbody>
</table>
3.5 Realisation of Coding and Decoding Using Shift Register Circuits

Circuit for coding of a systematic cyclic code:

\[ \begin{align*}
  b_0 & \quad b_1 & \quad b_2 & \quad \ldots & \quad b_{n-k-1} \\
  u_0 & \quad u_1 & \quad u_2 & \quad \ldots & \quad u_{k-2} & \quad u_{k-1} \\
  c_0 & \quad c_1 & \quad \ldots & \quad c_{n-1} \\
\end{align*} \]
Mode of operation:

1. Reading in the information bits $u_0, u_1, u_2, \ldots, u_{k-1}$, starting with $u_{k-1}$.
   Reading in from the „right“ corresponds to the multiplication by $x^{n-k}$.
   After reading in the $k$ information bits, the shift register contains the division remainder $r(x)$.
2. Switch-off of the switches $S_1$ and $S_2$ (break of the feedback loop).
3. Output of the control bits.
Example:
The following has to be coded: \( u(x) = x^3 + x^2 + 1 \)
with \( g(x) = x^3 + x + 1 \).

\[
\begin{array}{c|ccccc}
 t & s_1 & b_0 & b_1 & b_2 & s_2 & s_3 \\
\hline
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
 2 & 1 & 1 & 0 & 1 & 1 & 1 \\
 3 & 0 & 1 & 0 & 0 & 1 & 0 \\
 4 & 1 & 1 & 0 & 0 & 1 & 1 \\
 5 & 0 & 1 & 0 & 0 & 0 & 0 \\
 6 & 0 & 0 & 1 & 0 & 0 & 0 \\
 7 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]
3.5 Realisation of Coding and Decoding Using Shift Register Circuits

Circuit for the decoding:
With this circuit, the receive word is divided by the generator polynomial and the syndrome is calculated.
This circuit is also applied for decoding. However, the receive word is here read in from the right.
3.5 Realisation of Coding and Decoding Using Shift Register Circuits

Meggitt decoder: decoding with error correction at the same time
4 Burst-Error Correcting Block Codes

A burst error of the length \( l \geq 1 \) is understood by \( l \) consecutive digits of an error vector \( \mathbf{f} \), the first and the last digit of which are not equal to zero. The values of the digits located between the first and the last digit of the error burst are arbitrary.

With the Reed Solomon codes (RS codes), this chapter introduces representatives of the best known and often used codes being able to correct burst errors. They are closely related to the Bose-Chaudhuri-Hocquenghem codes (BCH codes), the strength of which is rather to correct statistically distributed single errors. They are introduced at the end of this chapter as a special case of the RS codes.

In order to understand these two codes, knowledge of the algebra of finite fields (Galois fields) and of their features is required. At first, this knowledge will be built in the following section.
4.1 Fundamental Algebraic Terms for Codes

In this section, a brief introduction of the algebra required for coding theory is given. For this, basic knowledge of set theory is assumed. The concept of function is summarised briefly.

Binary function:
In a not empty set $M$, a binary function $\circ$ is understood by an allocation of two elements $a, b$ of $M$ ($a, b \in M$) in such a way:

$$a \circ b = c. \quad (4.1a)$$

Here, it is not absolutely necessary that: $a \circ b = b \circ a. \quad (4.1b)$

If the condition (4.1b) is fulfilled, however, the set is commutative.

In terms of the binary function, this set is called closed (closed set), if:

$$a \circ b \in M \text{ for all } a, b \in M. \quad (4.1c)$$

A binary function is called associative, if:

$$a \circ (b \circ c) = (a \circ b) \circ c, \quad a, b, c \in M. \quad (4.1d)$$
4.1 Fundamental Algebraic Terms for Codes

Modulo-\(m\) addition:
For a set of whole numbers \(M = \{0, 1, 2, \ldots, m-1\}\) with \(m > 0\), let the modulo-\(m\) addition „\(\oplus\)“ be defined in such a way that for two elements \(a, b \in M\) the following applies:

\[
a \oplus b = r,
\]

where \(r\) is the remainder of the division of \(a + b\) by \(m\).

For the demonstration of eq. (4.2a):

\[
\frac{a + b}{m} = q + \frac{r}{m} \iff a + b = q \cdot m + r
\]

(4.2b)

According to eq. (4.2b), the remainder \(r\) is again an element of the set \(M\). The set \(M\) is closed in terms of the modulo-\(m\) addition.

Example:

\(M = \{0, 1, 2, 3, 4, 5, 6\}\) and modulo-7 addition „\(\oplus\)“.
4.1 Fundamental Algebraic Terms for Codes

Modulo-\( p \) multiplication:
For a set of positive whole numbers \( M = \{1, 2, 3, \ldots, p-1\} \), with \( p \) being a prime number, let the modulo-\( p \) multiplication „\( \otimes \)“ be defined in such a way that for two elements \( a, b \in M \) the following applies:
\[
a \otimes b = r,
\]
with \( r \) being the remainder of the division of \( a \cdot b \) by \( p \).

For the demonstration of eq. (4.3a):
\[
a \cdot b = q \cdot p + r
\]
\[
a \cdot b \quad \frac{a \cdot b}{p} = q + \frac{r}{p} \quad \iff \quad a \cdot b = q \cdot p + r
\]
(4.3b)

\( a \cdot b \) is not divisible by \( p \) without remainder, because \( p \) is a prime number. The remainder is again an element of the set \( M \). Thus, \( M \) is closed.

Example with \( p = 7 \):
\( M = \{1, 2, 3, 4, 5, 6\} \) and modulo-7 multiplication „\( \otimes \)“.

<table>
<thead>
<tr>
<th>( \otimes )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
4.1.1 Groups

4.1 Fundamental Algebraic Terms for Codes

A set G and a function ° are referred to as group (G,°), if the following
conditions for any a,b,c ∈ G are fulfilled:
– The function ° is associative,
– the set G is closed,
– the set G contains a neutral element e with the feature:
a ° e = e ° a = a with e ∈ G and
– for every element a an inverse element a′ ∈ G exists,
exists so that:
a ° a′ = a′ ° a = e.

(4.4a)
(4.4b)

A group is referred to as commutative or Abelian group, if for any a,b ∈ G
equation (4.1b) applies.
Example: The set of integer numbers Z = {... −2, −1, 0, 1, 2, ...} composes a
commutative group in terms of the addition. The neutral element is equal to
0. For the element a, the inverse element is -a.
In terms of the multiplication, Z does not compose a group, since the inverse
elements are not contained in Z.
Prof. Dr.-Ing. Thomas Kürner · Institut für Nachrichtentechnik · Technische Universität Braunschweig

Coding Theory WS 2008/2009

page 165


4.1.1 Groups

The neutral element of a group is unambiguous.
Proof: Assuming that there are two neutral elements $e_1$ and $e_2$ with the feature (4.4a), from this follows: $e_1 = e_1 \cdot e_2 = e_2 \cdot e_1 = e_2$.

Finite group:
If a group has a finite number of elements, it is referred to as finite group.

Examples:
The binary set $B = \{0, 1\}$ with the modulo-2 addition „⊕“. The neutral element is equal to 0 and the respective inverse elements:

\[
\begin{align*}
a &= 0 & \Rightarrow & a' = 0, \\
a &= 1 & \Rightarrow & a' = 1.
\end{align*}
\]

The sets with the modulo-$m$ addition and modulo-$p$ multiplication considered at the beginning of section 4.1 are also finite groups.
Cyclic groups
If all elements of a multiplicative group $G = \{1, g_1, g_2, \ldots, g_{m-1}\}$ can be illustrated as powers of at least one element $g_i$, this group is called cyclic. Then, $g_i$ is called a primitive element of the group of the $m$-th order. For the “one“ element, the representation $1 = g_i^0$ applies. $G$ consists then of $m$ elements of the form $g_i^j$, with $0 \leq j \leq m - 1$:

$$G = \{g_i^0, g_i^1, g_i^2, \ldots, g_i^{m-1}\}. \quad (4.5)$$

For any $g_k \in G$, the order $(g_k)$ describes the number of elements that can be composed of $g_k^j$.

Example:

Multiplicative modulo-5 group
$G = \{1, 2, 3, 4\}$.
The elements $z_1 = 2$ and $z_2 = 3$ are primitive elements.
4.1.2 Rings

A set $\mathcal{R}$, at which the addition $\oplus$ as well as the multiplication $\otimes$ are defined, is called a ring if the following conditions are fulfilled:

- $\mathcal{R}$ is a commutative group in terms of $\oplus$,
- $\mathcal{R}$ is closed in terms of $\otimes$,
- $\mathcal{R}$ is associative in terms of $\otimes$ and
- the distributive law is applied: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ with $a,b,c \in \mathcal{R}$.

A commutative ring exists if $\otimes$ is also commutative.

A ring with a neutral element exists, if:

$$a \otimes 1 = 1 \otimes a = a \quad \text{with} \quad a \in \mathcal{R}.$$  \hspace{1cm} (4.6a)

Example of a commutative ring with neutral element:
whole numbers $\mathbb{Z}$ with common addition and multiplication.

Example of a commutative ring with neutral element:
whole numbers $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$ with modulo-$m$ addition and modulo-$m$ multiplication. There, an unambiguous neutral element exists only, if $m$ is a prime number.
4.1.3 Fields

A set $K$, where the binary functions addition $\oplus$ and multiplication $\otimes$ are explained, is a finite field in terms of $\oplus$ and $\otimes$, if the following conditions are fulfilled:

- $K$ is a commutative group in terms of $\oplus$ with 0 as neutral element,
- $K$ without 0 is a commutative group in terms of $\otimes$ with 1 as neutral element and
- the distributive law is applied: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ with $a,b,c \in K$. 

(4.7)

Galois field:

A field with a finite number of elements forms a **Galois field $GF$**.

The finite set $Z_m = \{0,1, 2, \ldots, m-1\}$ of whole numbers is a Galois field $GF(m) = Z_m$, if $m$ is a prime number, that means: $m = p$.

Example: $GF(2)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
4.1.3 Fields

Several features of Galois fields are listed here without proof:

\[ a \otimes 0 = 0 \otimes a = 0, \]

(4.8a)

From the closure of the multiplication results that no zero divisors exist:

\[ a \neq 0 \text{ and } b \neq 0 \rightarrow a \otimes b \neq 0 \]

(4.8b)

\[ a \otimes b = 0 \text{ and } a \neq 0 \rightarrow b = 0, \]

(4.8c)

\[ -(a \otimes b) = (-a) \otimes b = a \otimes (-b), \]

(4.8d)

\[ a \otimes b = a \otimes c \text{ and } a \neq 0 \rightarrow b = c, \]

(4.8e)

\[ 1 \oplus 1 \oplus 1 \oplus ... \oplus 1 = 0, \]

\[ m \text{ summands} \]

(4.8f)

Every Galois field has at least one primitive element.
4.1.3 Fields

Example: \( GF(5) \)

<table>
<thead>
<tr>
<th>( a )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-a)</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( a^{-1} )</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \oplus )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Direct addition using table:

\[ 1 + 2 = 3, \quad 2 + 4 = 1, \quad 4 + 4 = 3. \]

Inverse elements of the addition: \( a + (-a) = 0. \)

Subtraction with inverse elements of the addition:

\[ 3 - 2 = 3 + (-2) = 3 + 3 = 1 \]
\[ 1 - 4 = 1 + (-4) = 1 + 1 = 2 \]

<table>
<thead>
<tr>
<th>( \otimes )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Direct multiplication using table:

\[ 1 \cdot 2 = 2, \quad 2 \cdot 4 = 3, \quad 4 \cdot 4 = 1. \]

Inverse elements of the multiplication: \( a \cdot (a^{-1}) = 1 \)

Division by inverse elements of the multiplication:

\[ 3 \div 2 = 3 \cdot (2^{-1}) = 3 \cdot 3 = 4 \]
\[ 1 \div 3 = 1 \cdot (3^{-1}) = 1 \cdot 2 = 2 \]
Example:
Solution to a linear system of equations in $GF(5)$

The following linear system of equations is given:

\begin{align*}
I: & \quad 2x_0 + x_1 = 2 \\
II: & \quad 3x_1 + x_2 = 3 \\
III: & \quad x_0 + x_1 + 2x_2 = 3
\end{align*}

Solution steps

\begin{align*}
I: & \quad 2x_0 = 2 - x_1 \\
II: & \quad x_2 = 3 - 3x_1 \\
2 \cdot III: & \quad (2 - x_1) + 2x_1 + 4(3 - 3x_1) = 2 \cdot 3 \\
\Rightarrow & \quad 2 + 4 \cdot 3 - 2 \cdot 3 = x_1(1 - 2 + 4 \cdot 3) \quad \Rightarrow \quad 3 = x_1 \\
\Rightarrow & \quad x_2 = 3 - 3x_1 = 4 \\
\Rightarrow & \quad x_0 = 2^{-1}(2 - x_1) = 2
\end{align*}
4.1.4 Expansion Fields

Problem:
In 4.1.3, finite fields are built, which are, however, limited to prime numbers with regard to the number of their elements. For example, \( G = \{0, 1, 2, 3\} \) does not provide a Galois field due to the required modulo-4 multiplication!

A much more general and, at the same time, more flexible construction of Galois fields can be achieved by the expansion of \( p^m \) elements. These are represented as polynomials of degree \( < m \) with the coefficients \( p_i \in GF(p) \):

\[
p(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_{m-1} x^{m-1} = \sum_{i=0}^{m-1} p_i x^i \quad \text{mit } p_{m-1} \neq 0. \quad (4.9)
\]

These overall \( p^m \) possible polynomials can be divided into:
- splittable or reducible polynomials and
- nonsplittable or irreducible polynomials.

In digital communication technology, particularly the expansion fields \( GF(2^m) \) are of great importance.
4.1.4 Expansion Fields

Reducible polynomials:
A polynomial $p(x)$ with coefficients from $GF(p)$ is called reducible in terms of $GF(p)$, if it can be decomposed into a product of two polynomials $p_a(x)$, $p_b(x) \neq 0, 1$ of lower order with coefficients from $GF(p)$.

Example: $p(x) = (x + 1) \cdot (x^2 + x + 1) = (x^3 + x^2 + x) + (x^2 + x + 1) = x^3 + 1$.

Irreducible polynomials:
A polynomial $p(x)$ with coefficients from $GF(p)$ is called irreducible in terms of $GF(p)$, if it cannot be represented as a product of two polynomials $p_a(x)$, $p_b(x) \neq 0, 1$ of lower degree with coefficients from $GF(p)$.
Thus, irreducible polynomials do not have a root in $GF(p)$ either.
The characteristics of irreducible polynomials and prime numbers are comparable.

Example: $p(x) = x^2 + x + 1$
Test polynomial of degree 1: $p_a(x) = x$, $p_b(x) = x + 1$.

$(x^2 + x + 1) \div x = x + 1 + \frac{1}{x}$; 
$(x^2 + x + 1) \div (x + 1) = x + \frac{1}{x+1}$. 
4.1.4 Expansion Fields

Roots:
An irreducible polynomial \( p(x) \) of degree > 1 with coefficients from \( \mathbb{GF}(p) \) does not have roots in \( \mathbb{GF}(p) \). An element \( \alpha \in \mathbb{GF}(p^m) \) of the expansion field is called root of \( p(x) \), if the following condition is fulfilled:

\[
p(\alpha) = 0
\]

(4.10)

There is an analogy between the expansion of Galois fields and the expansion of real numbers to complex numbers.

Definition of complex numbers;

\[
\mathbb{R} \rightarrow \mathbb{C}
\]

No solution of \( x^2 + 1 = 0 \); for \( x \in \mathbb{R} \);

field expansion:

\[
\alpha^2 + 1 = 0 \iff \alpha^2 = -1 \iff \alpha = \sqrt{-1};
\]

Definition of complex numbers;

\[
c = c_0 + c_1 \alpha \text{ with } c_0, c_1 \in \mathbb{R};
\]

of an expanded Galois field:

\[
\mathbb{GF}(2) \rightarrow \mathbb{GF}(2^2)
\]

\[
x^2 + x + 1 = 0 \quad x \in \mathbb{GF}(2)
\]

\[
\alpha^2 + \alpha + 1 = 0 \iff \alpha^2 = \alpha + 1 \mod 2
\]

of the elements of \( \mathbb{GF}(2^2) \):

\[
c = c_0 + c_1 \alpha \text{ with } c_0, c_1 \in \mathbb{GF}
\]
4.1.4 Expansion Fields

**Primitive polynomials:**
An irreducible polynomial \( p(x) \) of degree \( m \) and with the coefficients \( p_i \in GF(p) \) is referred to as primitive, if it has a root \( \alpha (p(\alpha) = 0) \) with the following characteristic:

\[
\alpha^i \mod p(\alpha), \text{ for } i = 0, 1, ..., p^m - 2 \tag{4.11}
\]

provides all possible \( p^m - 1 \) of \( 0 \) different elements of the Galois field \( GF(p^m) \). The element \( \alpha \in GF(p^m) \) is called **primitive element**, for which applies:

\[
\alpha^0 = \alpha^n = 1 \quad \text{für} \quad n = p^m - 1 \tag{4.12} \quad \text{and} \quad \alpha^k = \alpha^k \mod (p^m-1) \tag{4.13}
\]

Primitive polynomials are irreducible, in general the reversal does not apply. For every prime field \( GF(p^m) \) and every \( m \), at least one primitive polynomial \( p(x) \) exists. If more than one primitive polynomial exists, equivalent expansion fields result from the construction.
4.1.4 Galois Fields

Primitive polynomials for the construction of Galois fields $GF(2^m)$:

<table>
<thead>
<tr>
<th>$m$</th>
<th>primitive polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x + 1$</td>
</tr>
<tr>
<td>2</td>
<td>$x^2 + x + 1$</td>
</tr>
<tr>
<td>3</td>
<td>$x^3 + x + 1$</td>
</tr>
<tr>
<td>4</td>
<td>$x^4 + x + 1$</td>
</tr>
<tr>
<td>5</td>
<td>$x^5 + x^2 + 1$</td>
</tr>
<tr>
<td>6</td>
<td>$x^6 + x + 1$</td>
</tr>
<tr>
<td>7</td>
<td>$x^7 + x + 1$</td>
</tr>
<tr>
<td>8</td>
<td>$x^8 + x^6 + x^5 + x^4 + 1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$m$</th>
<th>primitive polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>$x^9 + x^4 + 1$</td>
</tr>
<tr>
<td>10</td>
<td>$x^{10} + x^3 + 1$</td>
</tr>
<tr>
<td>11</td>
<td>$x^{11} + x^2 + 1$</td>
</tr>
<tr>
<td>12</td>
<td>$x^{12} + x^7 + x^4 + x^3 + 1$</td>
</tr>
<tr>
<td>13</td>
<td>$x^{13} + x^4 + x^3 + x + 1$</td>
</tr>
<tr>
<td>14</td>
<td>$x^{14} + x^8 + x^6 + x + 1$</td>
</tr>
<tr>
<td>15</td>
<td>$x^{15} + x + 1$</td>
</tr>
<tr>
<td>16</td>
<td>$x^{16} + x^{12} + x^3 + x + 1$</td>
</tr>
</tbody>
</table>
There are three possibilities to represent the elements of a Galois field:

- Component representation (polynomial notation),
- Exponential representation (according to equation 4.11) and
- Vectorial representation (representation of the coefficients).

Example of $GF(2^2)$:
The elements of this Galois field in component representation are all $2^2 = 4$ polynomials of degree $< 2$, which can be built with the coefficients from $GF(2)$: $GF(2^2) = \{0, 1, \alpha, 1+\alpha\}$.

The representation of the coefficients only results in the vectorial representation: $GF(2^2) = \{0, 1, 01, 11\}$.

In exponential representation, the zero element is represented symbolically as $\alpha^{-\infty}$:

$GF(2^2) = \{\alpha^{-\infty}, \alpha^0, \alpha^1, \alpha^2\}$.

For the exponential representation, with equation 4.13 applies: $\alpha^k = \alpha^{k \mod 3}$.
### 4.1.4 Expansion Fields

Addition and multiplication tables for $GF(2^2)$ in

#### Exponential representation:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>$\alpha$</th>
<th>$\alpha^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\alpha$</td>
<td>$\alpha^2$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>$\alpha^2$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha^2$</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Component representation:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>$\alpha$</th>
<th>$1+\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\alpha$</td>
<td>$1+\alpha$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$1+\alpha$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>$1+\alpha$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$1+\alpha$</td>
<td>$1+\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Vectorial representation:

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>10</th>
<th>01</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>10</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>00</td>
<td>11</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>10</th>
<th>01</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>01</td>
</tr>
</tbody>
</table>
4.1.4 Expansion Fields

Polynomial residue class:
Vice versa, the single elements of the Galois fields can be also described as polynomial remainders of a modulo-\(p(x)\) calculation, with \(p(x)\) being a primitive polynomial.

Example in \(GF(2^2)\):
For this, \(p(x) = x^2 + x + 1\) is a primitive polynomial. The condition
\[
x^2 + x + 1 = 0
\]
corresponds to a calculation modulo \(x^2 + x + 1\). For any product, e. g. \((1 + \alpha) \cdot (1 + \alpha) = 1 + \alpha^2\), with the division by \(p(x = \alpha)\) results:
\[
(\alpha^2 + 1) : (\alpha^2 + \alpha + 1) = 1 + \frac{\alpha}{\alpha^2 + \alpha + 1}
\]
\[
\alpha = r(\alpha) \quad \text{(comp. component representation of } GF(2^2))
\]
\(GF(2^2)\) is defined by all possible results of arbitrary
\[f(\alpha) = p(\alpha) \text{ modulo } \alpha^2 + \alpha + 1.\]
4.1.4 Expansion Fields

Conjugate roots:
Let \( p(x) \), with \( \text{grad } p(x) = m \) and coefficients \( p_i \in \mathbb{GF}(p) \), be irreducible in terms of \( \mathbb{GF}(p) \), and let \( \alpha \) be the root of \( p(x) \), then, with \( \alpha \),

\[
\alpha, \alpha^2, ..., \alpha^{m-1}, \text{ where } \alpha^m = \alpha
\]

are also roots of \( p(x) \). These roots are referred to as conjugate.

If two elements \( a, b \in \mathbb{GF}(p^m) \) are conjugate to each other, you can also write: \( a \sim b \). (4.15)

The characteristics are:
- for all \( a \in \mathbb{GF}(p^m) \) applies: \( a \sim a \). (4.16a)
- for all \( a, b \in \mathbb{GF}(p^m) \) applies: \( a \sim b \Rightarrow b \sim a \) and (4.16b)
- for all \( a, b, c \in \mathbb{GF}(p^m) \) applies: \( a \sim b \) and \( b \sim c \Rightarrow a \sim c \). (4.16c)

By the conjugated root, a linear factorisation of \( p(x) \) is given:

\[
p(x) = \prod_{i=0}^{m-1} (x - \alpha^i)
\] (4.17)
4.1.4 Expansion Fields

**Equivalence classes:**
The combination of elements that are conjugate to each other according to
\[ [z_i] = \{z_i, z_j, z_k, z_l, \ldots \} \text{ with } z_i \sim z_j \sim z_k \sim z_l \ldots \] (4.18)
is referred to as equivalence classes and forms disjoint decompositions of \( GF(p^m) \).

**Cyclotomic fields:**
Cyclotomic fields \( K_j \) contain all exponents of the elements that are conjugate to each other:
\[
K_j = \{ j \cdot p^i \mod n \mid i = 0, 1, \ldots, m-1 \} \text{ with } n = p^m - 1
\] (4.19)
The term cyclotomy comes from the complex numbers, where the \( n \)-th root of unit and the one-element of a Galois field behave similarly:
\[ z^n = e^{j2\pi} = 1 \iff z^{p^m-1} = z^n = z^0 = 1. \] (4.20)
Therefore, expansion fields are also referred to as cyclotomic fields.
Minimum polynomial:
For every \( a \in GF(p^n) \), an unambiguously determined normalised polynomial \( m_{[a]}(x) \) of minimum degree with coefficients from \( GF(p) \) exists, the root of which is \( a \). This polynomial is called minimum polynomial. \( m_{[a]}(x) \) is irreducible.

Two conjugate elements have the same minimum polynomial:
\[
a \sim b \implies m_{[a]}(x) = m_{[b]}(x)
\] (4.21)

The degree of the minimum polynomial shows the power of the equivalence class.

The minimum polynomial is calculated using equation (4.17):
\[
m_{[a]}(x) = \prod_{b, b \in [a]} (x - b) = \prod_{i, i \in K_{[a]}} (x - a^{p^i})
\] (4.22)
The product of all minimum polynomials from $GF(p^m)$ results in:

$$\prod_{i} m_{[a_i]}(x) = x^n - 1$$

There, one of the minimum polynomials corresponds to the primitive polynomial.

Example:

$GF(2^4)$ with primitive polynomial $p(x) = x^4 + x + 1$ and primitive element $z$ with $z^4 + z + 1 = 0$. The elements in $GF(2^4)$ are:

$GF(2^4) = \{0, z^0, z^1, z^2, \ldots, z^{n-1}\}$ with $n = p^m - 1 = 2^4 - 1 = 15$.

In the exponent, the calculation is carried out modulo $n$: $z^n = z^0$.

<table>
<thead>
<tr>
<th>Cyclotomic field</th>
<th>Minimum polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_0 = {0}$</td>
<td>$m_0(x) = (x - z^0)$</td>
</tr>
<tr>
<td>$K_1 = {1,2,4,8}$</td>
<td>$m_1(x) = (x - z^1)(x - z^2)(x - z^4)(x - z^8) = x^4 + x + 1$</td>
</tr>
<tr>
<td>$K_3 = {3,6,9,12}$</td>
<td>$m_3(x) = (x - z^3)(x - z^6)(x - z^9)(x - z^{12}) = x^4 + x^3 + x^2 + x + 1$</td>
</tr>
<tr>
<td>$K_5 = {5,10}$</td>
<td>$m_5(x) = (x - z^5)(x - z^{10}) = x^2 + x + 1$</td>
</tr>
<tr>
<td>$K_7 = {7,11,13,14}$</td>
<td>$m_7(x) = (x - z^7)(x - z^{11})(x - z^{13})(x - z^{14}) = x^4 + x^3 + 1$</td>
</tr>
</tbody>
</table>

$m_0(x) \cdot m_1(x) \cdot m_3(x) \cdot m_5(x) \cdot m_7(x) = x^{15} - 1$
4.1.5 Discrete Fourier Transformation

The Discrete Fourier Transformation (DFT) describes in general a transformation. For the coding theory, it provides a very well arranged display format.

Based on a Galois field $GF(p^m)$ with the primitive element $z$, let polynomials of degree $\leq n - 1 = p^m - 2$ with coefficients from $GF(p^m)$ be given in the following form:

\[
\mathbf{a} = (a_0, a_1, \ldots, a_{n-1}) \quad \Leftrightarrow \quad a(x) = \sum_{i=0}^{n-1} a_i x^i \quad (4.24a)
\]

\[
\mathbf{A} = (A_0, A_1, \ldots, A_{n-1}) \quad \Leftrightarrow \quad A(x) = \sum_{j=0}^{n-1} A_j x^j \quad (4.24b)
\]

The transformation then reads:

\[
A_j = -a(z^{-j}) = -\sum_{i=0}^{n-1} a_i z^{-i \cdot j} \quad a_i = A(z^i) = \sum_{j=0}^{n-1} A_j z^{i \cdot j} \quad (4.25a/b)
\]
4.1.5 Discrete Fourier Transform

For the DFT, the following notations exist:

\[ a(x) \rightarrow A(x) \]

\[ a \rightarrow A \]

- time domain
- frequency domain

The DFT has the following characteristics:

- the transformation is one-to-one,
- \( \text{IDFT}(\text{DFT}(a(x))) = a(x) \), \hspace{1cm} (4.26a)
- \( \text{DFT}(\text{DFT}(a(x))) = -a(x) \), \hspace{1cm} (4.26b)
- \( \text{DFT}(\text{DFT}(\text{DFT}(a(x)))) = a(x) \), \hspace{1cm} (4.26c)
- \( z^{-j} \) is the root of \( a(x) \) \( \iff \) \( A_j = 0 \), \hspace{1cm} (4.26d)
- \( a(z^{-j}) = 0 \) \( \iff \) \( A = (A_0, A_1, \ldots, A_{j-1}, 0, A_{j+1}, \ldots, A_{n-1}) \)
- \( z^i \) is the root of \( A(x) \) \( \iff \) \( a_i = 0 \).
- \( A(z^i) = 0 \) \( \iff \) \( a = (a_0, a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_{n-1}) \) \hspace{1cm} (4.26e)
The polynomial transformation according to eq. (4.25a/b) can be also
described for the coefficient vectors by means of matrices:

\[
\begin{align*}
\mathbf{a} \rightarrow \mathbf{A}: \\
\begin{pmatrix}
A_0 \\
A_1 \\
A_2 \\
\vdots \\
A_{n-1}
\end{pmatrix} &= 
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & z^{-1} & z^{-2} & \cdots & z^{-(n-1)} \\
1 & z^{-2} & z^{-4} & \cdots & z^{-2(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & z^{-(n-1)} & z^{-2(n-1)} & \cdots & z^{-(n-1)^2}
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_{n-1}
\end{pmatrix} \\
\text{(4.27a)}
\end{align*}
\]

\[
\begin{align*}
\mathbf{A} \rightarrow \mathbf{a}: \\
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_{n-1}
\end{pmatrix} &= 
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & z & z^2 & \cdots & z^{n-1} \\
1 & z^2 & z^4 & \cdots & z^{2(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & z^{n-1} & z^{2(n-1)} & \cdots & z^{(n-1)^2}
\end{pmatrix}
\begin{pmatrix}
A_0 \\
A_1 \\
A_2 \\
\vdots \\
A_{n-1}
\end{pmatrix} \\
\text{(4.27b)}
\end{align*}
\]
Example:
The polynomial $A(x) = 4 + 5x$ with the coefficients from $GF(7)$ is to be transformed. $z = 5$ is chosen as a primitive element. Application of eq. (4.27b) results in:

\[
\begin{pmatrix}
  a_0 \\
  a_1 \\
  a_2 \\
  a_3 \\
  a_4 \\
  a_5 \\
\end{pmatrix}
\begin{pmatrix}
  1 & 1 & 1 & 1 & 1 & 1 \\
  1 & z & z^2 & z^3 & z^4 & z^5 \\
  1 & z^2 & z^4 & z^0 & z^2 & z^4 \\
  1 & z^3 & z^0 & z^3 & z^0 & z^3 \\
  1 & z^4 & z^2 & z^0 & z^4 & z^2 \\
  1 & z^5 & z^4 & z^3 & z^2 & z^1 \\
\end{pmatrix}
\begin{pmatrix}
  A_0 \\
  A_1 \\
  A_2 \\
  A_3 \\
  A_4 \\
  A_5 \\
\end{pmatrix}
= 4 \cdot \begin{pmatrix}
  1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 5 & 4 & 6 & 2 & 3 \\
  1 & 4 & 2 & 1 & 4 & 2 \\
  1 & 6 & 1 & 6 & 1 & 6 \\
  1 & 2 & 4 & 1 & 2 & 4 \\
  1 & 3 & 2 & 6 & 4 & 5 \\
\end{pmatrix}
\begin{pmatrix}
  4 \\
  5 \\
  0 \\
  0 \\
  0 \\
  0 \\
\end{pmatrix}
+ 5 \cdot \begin{pmatrix}
  1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 5 & 4 & 6 & 2 & 3 \\
  1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 2 & 4 & 1 & 2 & 4 \\
  1 & 3 & 2 & 6 & 4 & 5 \\
\end{pmatrix}
\begin{pmatrix}
  2 \\
  1 \\
  3 \\
  6 \\
  0 \\
  5 \\
\end{pmatrix}
\]
4.1.5 Discrete Fourier Transform

The inverse transform of the polynomial \( a(x) = 2 + x + 3x^2 + 6x^3 + 5x^5 \) with the coefficients from \( GF(7) \) calculated on the previous page is carried out using eq. (4.27a). The primitive element \( z = 5 \) is retained.

\[
\begin{pmatrix}
A_0 \\
A_1 \\
A_2 \\
A_3 \\
A_4 \\
A_5
\end{pmatrix} = -
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & z^{-1} & z^{-2} & z^{-3} & z^{-4} & z^{-5} \\
1 & z^{-2} & z^{-4} & z^{-0} & z^{-2} & z^{-4} \\
1 & z^{-3} & z^{-0} & z^{-3} & z^{-0} & z^{-3} \\
1 & z^{-4} & z^{-2} & z^{-0} & z^{-4} & z^{-2} \\
1 & z^{-5} & z^{-4} & z^{-3} & z^{-2} & z^{-1}
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5
\end{pmatrix}
\]

\[
\begin{pmatrix}
A_0 \\
A_1 \\
A_2 \\
A_3 \\
A_4 \\
A_5
\end{pmatrix} = -
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & z^5 & z^4 & z^3 & z^2 & z^1 \\
1 & z^4 & z^2 & z^0 & z^4 & z^2 \\
1 & z^3 & z^0 & z^3 & z^0 & z^3 \\
1 & z^2 & z^4 & z^0 & z^2 & z^4 \\
1 & z^1 & z^2 & z^3 & z^4 & z^5
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5
\end{pmatrix}
\]
4.1.5 Discrete Fourier Transform

Inverse transform (continuation):

\[
\begin{bmatrix}
A_0 \\
A_1 \\
A_2 \\
A_3 \\
A_4 \\
A_5
\end{bmatrix} = -\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 3 & 2 & 6 & 4 \\
1 & 2 & 4 & 1 & 2 \\
1 & 6 & 1 & 6 & 1 \\
1 & 4 & 2 & 1 & 4 \\
1 & 5 & 4 & 6 & 2
\end{bmatrix} \begin{bmatrix}
2 \\
1 \\
3 \\
6 \\
0 \\
5
\end{bmatrix} = -\begin{bmatrix}
3 \\
2 \\
0 \\
0 \\
0 \\
0
\end{bmatrix} = \begin{bmatrix}
4 \\
5 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

\[A = (4, 5, 0, 0, 0, 0) \quad a = (2, 1, 3, 6, 0, 5)\]
4.1.5 Discrete Fourier Transform

A shift register circuit of the DFT according to equation (4.25a) is shown in the following figure:

Here the \( j \)-th step of the transformation for the calculation of \( A(x) \) is mapped. At the beginning \((j = 0)\), the register is initialised with the time domain \( a_0, a_1, \ldots, a_{n-1} \), so that the following applies:

\[
A_0 = -(a_0 + a_1 + \ldots + a_{n-1}) \mod p^m. \tag{4.28a}
\]
A shift register circuit of the IDFT according to equation (4.25b) is shown in the following figure:

Here the $i$-th step of the transformation for the calculation of $a(x)$ is mapped. At the beginning ($i = 0$) the register is initialised with the frequency domain $A_0, A_1, ..., A_{n-1}$, so that the following applies:

$$a_0 = A_0 + A_1 + ... + A_{n-1} \mod p^m.$$  

(4.28b)
Finally, two theorems of the DFT are to be considered.

Let polynomials $a(x), b(x), c(x)$ of degree $\leq n - 1 = p^m - 2$ be given with the coefficients from $GF(p^m)$, the primitive element of which let be referred to as $z$. For any $x \in GF(p^m), x^n = x^0 = 1$, i.e. modulo-$(x^n - 1)$ calculation, applies. For the product of two polynomials $a(x) \cdot b(x) = c(x)$ then applies:

$$(a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}) \cdot (b_0 + b_1 x + \ldots + b_{n-1} x^{n-1}) = (c_0 + c_1 x + \ldots + c_{n-1} x^{n-1})$$

$$c_0 = b_0 \cdot a_0 + b_1 a_{n-1} + b_2 a_{n-2} + \ldots + b_{n-1} a_1$$
$$c_1 = b_0 \cdot a_1 + b_1 a_0 + b_2 a_{n-1} + \ldots + b_{n-1} a_2$$
$$\vdots$$
$$c_i = \sum_{j=0}^{n-1} b_j \cdot a_{i-j} \mod n \quad (4.29)$$

The product of two polynomials corresponds to the **cyclic convolution** of the coefficients:

$$a * b \iff a(x) \cdot b(x) \mod (x^n - 1) \quad (4.30)$$
4.1.5 Discrete Fourier Transformation

Convolution Theorem of the DFT:

\[
\begin{align*}
    &a(x) \cdot b(x) \mod (x^n - 1) & \rightarrow & - A_j \cdot B_j \\
    &a_i \cdot b_i & \rightarrow & A(x) \cdot B(x) \mod (x^n - 1)
\end{align*}
\]  \hspace{1cm} (4.31a/b)

with \(a(x) \rightarrow A(x)\), \(b(x) \rightarrow B(x)\).

Proof of (4.31a): Using equation (4.3b), for the modulo calculation the representation \(c(x) = a(x) \cdot b(x) - \gamma(x) \cdot (x^n - 1)\) results and thus \(C_j = - c(z^{-j}) = - a(z^{-j}) \cdot b(z^{-j}) + \gamma(z^{-j}) \cdot (z^{-jn} - 1).\)

\[
- A_j \quad -B_j 
\]

Shift Theorem of the DFT:

\[
\begin{align*}
    &x \cdot a(x) \mod (x^n - 1) & \rightarrow & z^{-j} \cdot A_j \\
    &z^i \cdot a_i & \rightarrow & x \cdot A(x) \mod (x^n - 1)
\end{align*}
\]  \hspace{1cm} (4.32a/b)

with \(a(x) \rightarrow A(x)\).

Proof: by inserting \(b(x) = x\) and \(B(x) = x\), respectively, in the Convolution Theorem.
4.2 Reed-Solomon Codes

The Reed-Solomon codes (RS codes) were developed approx. in 1960. They can be constructed analytically. There, the minimum distance can be provided so that their weight distribution is known. The RS codes fulfil the Singleton bound with equality. They are very efficient and vitally important in practice. Among other things, they are used in spacecraft communications, for CDs (Compact Disk) and in mobile radio communications.

RS codes are symbol-oriented codes, that means they are especially dedicated for the correction of burst errors. In contrast, they can be inefficient for the correction of single errors, since always whole symbols have to be corrected. A respective redundancy is required here.
4.2.1 Construction of Reed-Solomon Codes

Fundamental theorem of the linear algebra:
A polynomial \( A(x) = A_0 + A_1 x + A_2 x^2 + \ldots + A_{k-1} x^{k-1} \) of degree \( k - 1 \) with coefficients \( A_i \in GF(p^m) \) and \( A_{k-1} \neq 0 \) has at most \( k - 1 \) different roots.

The proof of this theorem is to confirm that a polynomial can be represented by a number of linear factors restricted by its degree:

\[
A(x) = A_{k-1} \cdot \prod_{i=1}^{k-1} (x - x_i)
\]  

(4.33)

Hamming weight of general vectors:
The Hamming weight \( w_H(c) \) of a vector \( c \) is defined as the number of elements of \( c \) that are not zero.

Eq. (2.11), which calculates the weight of a binary vector, can be seen as a special case here, if the elements of the vector come from \( GF(2) \).
### Theorem for the minimum weight of vectors:

Let \( A(x) = A_0 + A_1 x + A_2 x^2 + \ldots + A_{k-1} x^{k-1} \) be a polynomial with the coefficients \( A_i \in \mathbf{GF}(p^m) \), where its degree is bounded by:

\[
\text{grad } A(x) = k - 1 \leq n - d,
\]

then for the weight of a vector applies \( \mathbf{a} = (a_0, a_1, a_2, \ldots, a_{n-1}) \) with the relation \( \mathbf{a}(x) \bullet A(x) \):

\[
w_H(\mathbf{a}) \geq d. \tag{4.34}
\]

**Proof:**

Equation (4.26e) says that roots of \( A(z^i) \) correspond to elements \( a_i = 0 \). The polynomial \( A(x) \) has at most \( k - 1 \) different roots, so that the vector \( \mathbf{a} \) has at least \( n - (k - 1) = n - k + 1 \geq d \) points different from zero.

According to this theorem vectors can be constructed, the weight of which depends only on the polynomial degree of the transformed vector. These have a defined minimum weight and thus a defined minimum distance.
4.2.1 Construction of Reed-Solomon Codes

Definition of Reed-Solomon codes:

Let \( z \) be a primitive element from \( \text{GF}(p^m) \), then \( a = (a_0, a_1, a_2, ..., a_{n-1}) \) is a code word of the RS code \( C \) of the length \( n = p^m - 1 \), dimension \( k = p^m - d \) and minimum distance \( d = n - k + 1 \), if the following applies:

\[
C = \{ a \mid a_i = A(z^i), \quad \text{grad} \ A(x) \leq k - 1 = n - d \}.
\]

The code vector \( (A_0, A_1, A_2, ..., A_{k-1}) \) corresponds to the information to be coded. By filling with \( n - k = d - 1 \) zeros (also called test symbols or test frequencies), the code word in the frequency domain \( A \) results, by IDFT the code word in the time domain \( a \) results:

\[
a = (a_0, a_1, a_2, ..., a_{n-1}) \quad \Rightarrow \quad A = (A_0, A_1, A_2, ..., A_{k-1}, 0, 0, ..., 0)
\]

The RS codes satisfy the Singleton bound with equality, since the following applies:

\[
n - k = d_{\min} - 1
\]

(4.36)

The best case is an odd \( d_{\min} \), so that applies: \( t = (d_{\min} - 1)/2 \).

And thus:

\[
n - k = 2 \ t
\]

(4.37)

I.e., the number of detectable and corrigible errors can be adjusted.
4.2.1 Construction of Reed-Solomon Codes

Example:
Selection of the Galois field $\mathbb{GF}(p^m)$ with $p = 2$ and $m = 4 \Rightarrow \mathbb{GF}(2^4)$.
The code word length is then: $n = p^m - 1 = 16 - 1 = 15$.
Let the number of corrigible symbols be $t = 3 \Rightarrow n - k = d_{\text{min}} - 1 = 2$ $t = 6$. Then the code word in the frequency domain reads in general:

$$
\begin{array}{cccccccccc}
A_0 & A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_7 & A_8 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$

$k = n - (d_{\text{min}} - 1) = 9$

The number of information digits / bits is: $k = 9 \ / \ k \cdot m = 9 \cdot 4 = 36$;
the number of code word digits / bits is: $n = 15 \ / \ n \cdot m = 15 \cdot 4 = 60$.

This code can be referred to as hexadecimal $(15,9)$-RS code and as binary $(60,36)$-RS code.
4.2.1 Construction of Reed-Solomon Codes

Possible error vectors in a (partial) binary notation with \(0 = (0000)\) are:

\[
\begin{align*}
\mathbf{f} &= (0001111111110000000000) \text{ is corrigible}; \\
\mathbf{f} &= (00011111110000000000000) \text{ is not corrigible}; \\
\mathbf{f} &= (0100000001000100000000100) \text{ is not corrigible}.
\end{align*}
\]

From the extreme cases it can be derived that definitely 3 single errors or 9 consecutive errors are corrigible.

Thus, burst errors are corrigible up to a length of:

\[
t_{b,Bündel} = m (t - 1) + 1 = m (d - 3) / 2 + 1
\]

(4.38)

For the code rate of the RS code applies:

\[
R_C = \frac{k}{n} = \frac{n - (d - 1)}{n} = 1 - \frac{d - 1}{n}.
\]

Example: For the (15,9)-RS code, \(R_C = k / n = 9 / 15 = 60\%\).
4.2.1 Construction of Reed-Solomon Codes

Generator polynomial:
Due to the restriction of the degree of the code word \( A(x) \), for which applies:
\[
A_j = 0 \quad \text{for} \quad j = k \ldots n - 1,
\]
and the DFT characteristic (4.26d), every code word polynomial \( a(x) \) has the roots:
\[
a(z^{-j}) = 0 \quad \text{for} \quad j = k \ldots n - 1.
\]
The product of all linear factors resulting from the roots forms the generator polynomial of the respective RS code:
\[
g(x) = \prod_{j=k}^{n-1} (x - z^{-j}) = \prod_{j=1}^{n-k} (x - z^j)
\]
(4.39)
Every code word polynomial can then be described as:
\[
a(x) = u(x) \cdot g(x).
\]
Since an RS code can be described by a generator polynomial, it is a cyclic code.
4.2.1 Construction of Reed-Solomon Codes

Check polynomial:
The check polynomial $h(x)$ is the complementary polynomial to the generator polynomial and contains as roots all those elements different from zero which are no roots of $g(x)$:

$$h(x) = \prod_{j=0}^{k-1} (x - z^{-j}) = \prod_{j=n-k+1}^{n} (x - z^j)$$

(4.40)

Proof:

$$a(x) \cdot h(x) = u(x) \cdot g(x) \cdot h(x)$$

$$= u(x) \cdot \prod_{i=0}^{n-1} (x - z^i) = u(x) \cdot (x^n - 1) = 0 \mod (x^n - 1)$$

For the polynomials $g(x)$, $u(x)$ and $h(x)$ the following relations apply:

- $grad(g(x)) = n - k$
- $grad(u(x)) = k - 1$
- $grad(g(x) \cdot u(x)) = n - 1$
- $grad(h(x)) = k$. 
Example:
For the (6,2)-RS code over $GF(7)$ using the primitive element $z = 5$, the generator polynomial $g(x)$ is:

$$g(x) = \prod_{i=2}^{5} (x - 5^{-i}) = \prod_{i=1}^{4} (x - 5^{i})$$

$$= (x - 5)(x - 4) \cdot (x - 6)(x - 2)$$

$$= (6 + 5x + x^2) \cdot (5 + 6x + x^2) = 2 + 5x + 6x^2 + 4x^3 + x^4$$

The check polynomial $h(x)$ is then calculated to be:

$$h(x) = \frac{x^n - 1}{g(x)}$$

$$h(x) = \frac{x^6 - 1}{2 + 5x + 6x^2 + 4x^3 + x^4} = x^2 + 3x + 3 = (x - 1)(x - 3)$$
4.2.1 Construction of Reed-Solomon Codes

Generator matrix:
The RS code is a cyclic code, the generator matrix $G$ according to equation 3.18 of which can be developed from the coefficients of the generator polynomial $g(x)$.

Example with $g(x) = 2 + 5x + 6x^2 + 4x^3 + x^4$ (compare page 203):

$$G = \begin{pmatrix} 2 & 5 & 6 & 4 & 1 & 0 \\ 0 & 2 & 5 & 6 & 4 & 1 \end{pmatrix}$$

Systematisation:
By transformation, every generator matrix can be systemised.

Example of the above generator matrix: The systematisation steps are here 1) substitution of the first row by the sum of the rows and 2) multiplication of all elements by 4.

$$G \Rightarrow G_1 = \begin{pmatrix} 2 & 0 & 4 & 3 & 5 & 1 \\ 0 & 2 & 5 & 6 & 4 & 1 \end{pmatrix} \Rightarrow G_{\text{syst}} = \begin{pmatrix} 1 & 0 & 2 & 5 & 6 & 4 \\ 0 & 1 & 6 & 3 & 2 & 4 \end{pmatrix}$$
4.2.1 Construction of Reed-Solomon Codes

Control matrix:
The control matrix $H$ can be determined in three ways:

1. Due to the cyclic characteristic of the RS codes using eq. 3.25b

Example with $h(x) = x^2 + 3x + 3$ (see page 203):

$$H_1 = \begin{pmatrix}
1 & 3 & 3 & 0 & 0 & 0 \\
0 & 1 & 3 & 3 & 0 & 0 \\
0 & 0 & 1 & 3 & 3 & 0 \\
0 & 0 & 0 & 1 & 3 & 3
\end{pmatrix}$$

2. Using the systematic generator matrix: $G = [I_k \ P] \Rightarrow H_2 = [-P^T \ I_{n-k}]$

Example with a systematic generator matrix $G_{syst}$ (compare page 204):

$$H_2 = \begin{pmatrix}
-2 & -6 & 1 & 0 & 0 & 0 \\
-5 & -3 & 0 & 1 & 0 & 0 \\
-6 & -2 & 0 & 0 & 1 & 0 \\
-4 & -4 & 0 & 0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
5 & 1 & 1 & 0 & 0 & 0 \\
2 & 4 & 0 & 1 & 0 & 0 \\
1 & 5 & 0 & 0 & 1 & 0 \\
3 & 3 & 0 & 0 & 0 & 1
\end{pmatrix}$$
3. Direct evaluation of the check equations:

\[ A_j = -a(z^{-j}) = 0 = -\sum_{i=0}^{n-1} a_i \cdot z^{-ij} \quad \text{for} \quad j = k \ldots n-1 \]

Inserting \( j = k \ldots n-1 \), the sums for all \( j \) yield as a matrix:

\[
\begin{pmatrix}
1 & z^{-k} & z^{-2k} & \ldots & z^{-(n-1)k} \\
1 & z^{-(k+1)} & z^{-2(k+1)} & \ldots & z^{-(n-1)(k+1)} \\
1 & z^{-(k+2)} & z^{-2(k+2)} & \ldots & z^{-(n-1)(k+2)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & z^{-(n-1)} & z^{-2(n-1)} & \ldots & z^{-(n-1)^2}
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_{n-1}
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
0 \\
0 \\
\vdots \\
0
\end{pmatrix}
\]

(4.41)

**Example**

with \( z = 5 \)

in \( GF(7) \):

\[
-H_3 = 
\begin{pmatrix}
1 & z^{-2} & z^{-4} & z^{0} & z^{-2} & z^{-4} \\
1 & z^{-3} & z^{0} & z^{-3} & z^{0} & z^{-3} \\
1 & z^{-4} & z^{-2} & z^{0} & z^{-4} & z^{-2} \\
1 & z^{-5} & z^{-4} & z^{-3} & z^{-2} & z^{-1}
\end{pmatrix}
= 
\begin{pmatrix}
1 & 2 & 4 & 1 & 2 & 4 \\
1 & 6 & 1 & 6 & 1 & 6 \\
1 & 4 & 2 & 1 & 4 & 2 \\
1 & 5 & 4 & 6 & 2 & 3
\end{pmatrix}
\]
4.2.1 Construction of Reed-Solomon Codes

Syndrome calculation:
The received signal \( r \) can be described as
\[
  r = a + f \quad \text{and accordingly} \quad r(x) = a(x) + f(x)
\]
and accordingly \( R = A + F \) and accordingly
\[
  R(x) = A(x) + F(x).
\]
From this, the following can be derived in the frequency range:
\[
  A(x) = A_0 x^0 + A_1 x^1 + \ldots + A_{k-1} x^{k-1}
\]
\[
  F(x) = F_0 x^0 + F_1 x^1 + \ldots + F_{k-1} x^{k-1} + F_k x^k + \ldots + F_{n-1} x^{n-1}
\]
\[
  R(x) = R_0 x^0 + R_1 x^1 + \ldots + R_{k-1} x^{k-1} + F_k x^k + \ldots + F_{n-1} x^{n-1}
\]
with \( R_j = A_j + F_j \) syndrome coefficients

In the frequency range, the syndrome is a part of the receive vector which is located at the test frequencies:
\[
  S(x) = S_0 x^0 + \ldots + S_{n-k-1} x^{n-k-1} = F_k x^0 + \ldots + F_{n-1} x^{n-k-1}
\]  (4.42)
4.2.1 Construction of Reed-Solomon Codes

In vectorial notation, the syndrome is

\[ S = (S_0, S_1, \ldots, S_{n-k-1}). \]

The syndrome can also be calculated by multiplication by the check matrix:

\[
\begin{align*}
S &= r \cdot H_3^T \\
S^T &= H_3 \cdot r^T
\end{align*}
\]  \hspace{1cm} (4.43a)

Written out, the following representation results:

\[
\begin{pmatrix}
S_0 \\
S_1 \\
S_2 \\
\vdots \\
S_{n-k-1}
\end{pmatrix}
= 
\begin{pmatrix}
1 & z^{-k} & z^{-2k} & \cdots & z^{-(n-1)k} \\
1 & z^{-(k+1)} & z^{-2(k+1)} & \cdots & z^{-(n-1)(k+1)} \\
1 & z^{-(k+2)} & z^{-2(k+2)} & \cdots & z^{-(n-1)(k+2)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & z^{-(n-1)} & z^{-2(n-1)} & \cdots & z^{-(n-1)^2}
\end{pmatrix}
\begin{pmatrix}
r_0 \\
r_1 \\
r_2 \\
\vdots \\
r_{n-1}
\end{pmatrix}
\]  \hspace{1cm} (4.43b)
4.2.1 Construction of Reed-Solomon Codes

Generic RS code:
The RS code is a cyclic code. According to eq. 4.32a, cyclic shifts in the time domain have no effect on the test frequencies:

\[ a(x) = x^k \cdot b(x) \mod (x^n-1) \]

\[ z^{-jk} \cdot B_j = A_j \rightarrow B_j = 0 \Rightarrow A_j = 0. \]

Otherwise, using equations 4.26e and 4.32b, it is certain that the cyclic shift of a polynomial in the frequency range does not have an effect on the weight and thus on the minimum distance of the appropriate code word in the time domain.

Definition: A generic RS code of the length \( n \), dimension \( k \) and minimum distance \( d \) is defined by:

\[ C = \{ a_i = A(z^i), \ A(x) = x^k \cdot B(x) \mod (x^n-1), \ \text{grad } B(x) \leq k - 1 = n - d \}, \]

with \( k \) being any number.

(4.44)
Weighting function of MDS codes:
The weighting function $A(z)$ for an $(n,k)$-MDS code with the minimum distance $d = n - k + 1$ over the $\mathbb{GF}(q = p^m)$ can be calculated as follows:

$$A(z) = \sum_{i=0}^{n} A_i z^i$$

with $A_i = \begin{cases} 1 & \text{for } i = 0 \\ 0 & \text{for } 1 \leq i < d \end{cases}$

and $A_i = \binom{n}{i}(q-1) \sum_{j=0}^{i-d} (-1)^j \binom{i-1}{j} q^{i-d-j}$ for $i \geq d$ \hspace{1cm} (4.45)

The RS codes belong to the MDS codes (compare page 100).
The proof is based on combinatorial considerations in terms of essential MDS characteristics of the RS codes.
4.2.1 Construction of Reed-Solomon Codes

Example: weighting distribution of the (7,3)-RS code over $GF(q = 2^3)$ with the minimum distance $d = (n - k) + 1 = 5$.

$A_0 = 1$
$A_1 = A_2 = A_3 = A_4 = 0$

$A_5 = \binom{7}{5} \cdot 7 \cdot 1 = \frac{7!}{5! \cdot (7 - 5)!} \cdot 7 = 21 \cdot 7 = 147$

$A_6 = \binom{7}{6} \cdot 7 \cdot (8 - \binom{5}{1} \cdot 1) = \frac{7!}{6! \cdot (7 - 6)!} \cdot (8 - 5) = 147$

$A_7 = \binom{7}{7} \cdot 7 \cdot (8^2 - \binom{6}{1} \cdot 8 + \binom{6}{2} \cdot 1) = 7 \cdot (64 - 48 + 15) = 217$

With this, the weighting function is: $A(z) = 1 + 147z^5 + 147z^6 + 217z^7$

Check: $\sum_{i=0}^{n} A_i = 1 + 147 + 147 + 217 = 512 = q^3 = 8^3$
4.2.2 Coding of Reed-Solomon Codes

The information $u = (u_0, u_1, u_2, ..., u_{k-1})$ and the information polynomial $u(x) = u_0x^0 + u_1x^1 + ... + u_{k-1}x^{k-1}$, respectively, is to be coded. In the following, three possible coding methods are identified:

1. **Coding in the frequency range**, that means inverse transform of $u$ by means of IDFT (eq. 4.25b):

   $$A = (u_0, u_1, u_2, ..., u_{k-1}, 0, 0, ..., 0) \quad \bullet \quad a = (a_0, a_1, a_2, ..., a_{n-1})$$  \hspace{1cm} (4.46a)

2. **Non-systematic coding in the time domain**, that means multiplication of $u(x)$ by generator polynomial:

   $$a(x) = u(x) \cdot g(x)$$  \hspace{1cm} (4.46b)

3. **Systematic coding in the time domain**, that means multiplication of $u(x)$ by $x^{n-k}$: $u(x) \cdot x^{n-k} = u_0x^{n-k} + u_1x^{n-k+1} + ... + u_{k-1}x^{n-1}$, division of $u(x) \cdot x^{n-k}$ by $g(x)$:

   $$u(x) \cdot x^{n-k} = q(x) \cdot g(x) + r(x)$$ with $r(x) = r_0 + r_1x + ... + r_{n-k-1}x^{n-k-1}$ and solving according to the code word: $-r(x) + u(x) \cdot x^{n-k} = q(x) \cdot g(x) = a(x)$.

   $$a = (-r_0, -r_1, -r_2, ..., -r_{n-k-1}, u_0, u_1, u_2, ..., u_{k-1})$$  \hspace{1cm} (4.46c)
4.2.2 Coding of Reed-Solomon Codes

Block diagram for coding in the frequency range (method 1):

\[ a_i = \sum_{j=0}^{n-1} A_j \cdot z^{ij} \]

Message word

\[ \begin{array}{c}
  u_0 \\
  u_1 \\
  u_2 \\
  \vdots \\
  u_{k-1} \\
  0 \\
  0 \\
  \vdots \\
  0
\end{array} \]

Code word

\[ \begin{array}{c}
  A_0 \\
  A_1 \\
  A_2 \\
  \vdots \\
  A_{k-1} \\
  A_k \\
  A_{k+1} \\
  \vdots \\
  A_{n-1}
\end{array} \]

IDFT

\[ \begin{array}{c}
  a_0 \\
  a_1 \\
  a_2 \\
  \vdots \\
  a_{n-1} \\
  c_0 \\
  c_1 \\
  c_2 \\
  \vdots \\
  c_{n-1}
\end{array} \]
4.2.2 Coding of Reed-Solomon Codes

Example of coding in the frequency range (method 1):
A (6,2)-RS code over $GF(7)$ with the primitive element $z = 5$ is considered. The information $u = (3,3)$ is to be coded.

$n = 6; k = 2$ and $A(x) = 3 + 3x$. From $a_i = A(x = z^i)$ (eq. 4.25b) follows:

- $a_0 = A(x = z^0 = 5^0 = 1) = A_0 + A_1z^0 = 3 + 3 \cdot 1 = 6$,
- $a_1 = A(x = z^1 = 5^1 = 5) = A_0 + A_1z^1 = 3 + 3 \cdot 5 = 4$,
- $a_2 = A(x = z^2 = 5^2 = 4) = A_0 + A_1z^2 = 3 + 3 \cdot 4 = 1$,
- $a_3 = A(x = z^3 = 5^3 = 6) = A_0 + A_1z^3 = 3 + 3 \cdot 6 = 0$,
- $a_4 = A(x = z^4 = 5^4 = 2) = A_0 + A_1z^4 = 3 + 3 \cdot 2 = 2$,
- $a_5 = A(x = z^5 = 5^5 = 3) = A_0 + A_1z^5 = 3 + 3 \cdot 3 = 5$.

Result: $A = (3,3,0,0,0,0)$  \[\bullet\bullet\bullet\]  $a = (6,4,1,0,2,5)$.

With $B(x) = 1 + 1x$, the code word $b$ results as follows

- $B = (1,1,0,0,0,0)$  \[\bullet\bullet\bullet\]  $b = (2,6,5,0,3,4)$.

The Hamming distance is: $d_H(a,b) = d_H((6,4,1,0,2,5),(2,6,5,0,3,4)) = 5$.

For comparison: $d_{\text{min}} = n - k + 1 = 6 - 2 + 1 = 5$. 

Example of the systematic coding in the time domain (method 3):
A (6,2)-RS code over $GF(7)$ with the primitive element $z = 5$ is considered. The information $u = (3,4)$ is to be coded systematically.

$u = (3,4) \Rightarrow u(x) = 3 + 4x \Rightarrow u(x) \cdot x^{n-k} = 3x^4 + 4x^5$.

Division of $u(x)$ by $g(x) = 2 + 5x + 6x^2 + 4x^3 + x^4$ (compare page 203):

$$
\begin{align*}
(4x^5 + 3x^4) & \div (1x^4 + 4x^3 + 6x^2 + 5x + 2) = 4x + 1 + \frac{r(x)}{g(x)} \\
-(4x^5 + 2x^4 + 3x^3 + 6x^2 + 1x) & = 4x + 1 + \frac{r(x)}{g(x)} \\
1x^4 + 4x^3 + 1x^2 + 6x & = 4x + 1 + \frac{r(x)}{g(x)} \\
-(1x^4 + 4x^3 + 6x^2 + 5x + 2) & = 4x + 1 + \frac{r(x)}{g(x)} \\
2x^2 + 1x + 5 & = r(x) \\
-r(x) + u(x) \cdot x^{n-k} & = 2 + 6x + 5x^2 + 3x^4 + 4x^5 \Rightarrow a = (2,6,5,0,3,4)
\end{align*}
$$

An alternative is given by coding with a systematic generator matrix:

$$
b = u \cdot G_{syst} = (3,4) \cdot \begin{pmatrix} 1 & 0 & 2 & 5 & 6 & 4 \\ 0 & 1 & 6 & 3 & 2 & 4 \end{pmatrix} = (3,4,2,6,5,0)$$
Example of the systematic coding in the time domain (method 3):
A (7,3)-RS code over $\mathbf{GF}(2^3)$ is considered. The primitive polynomial is $p(x) = 1 + x + x^3$.
The code word length is calculated with $n = p^m - 1 = 8 - 1 = 7$. The number of correctible symbols is $t = 2$, because $n - k = 4$ with $k = 3$. With $p(x)$, the following relation applies to the primitive element: $1 + z + z^3 = 0$.
The single elements in exponential and component representation of this expansion field are given in the table.

The generator polynomial is calculated according to eq. (4.39):
$$g(x) = (x - z) (x - z^2) (x - z^3) (x - z^4)$$
$$= (x^2 + xz^3 + z^3) (x^2 + xz^6 + 1)$$
$$= x^4 + x^3z^3 + x^2 + xz + z^3.$$  

The check polynomial is calculated according to eq. (4.40):
$$h(x) = (x - z^5) (x - z^6) (x - z^7)$$
$$= x^3z^3 + x^2z^3 + xz^2 + z^4.$$  

<table>
<thead>
<tr>
<th>$z^0$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z^1$</td>
<td>$z$</td>
</tr>
<tr>
<td>$z^2$</td>
<td>$z^2$</td>
</tr>
<tr>
<td>$z^3$</td>
<td>$1 + z$</td>
</tr>
<tr>
<td>$z^4$</td>
<td>$z + z^2$</td>
</tr>
<tr>
<td>$z^5$</td>
<td>$1 + z + z^2$</td>
</tr>
<tr>
<td>$z^6$</td>
<td>$1 + z^2$</td>
</tr>
</tbody>
</table>
Example (continuation):
The information \( u = (1, z, z^2) \) is to be coded systematically.
\[ u = (1, z, z^2) \Rightarrow u(x) = 1 + xz + x^2z^2 \Rightarrow u(x) \cdot x^{n-k} = x^4 + x^5z + x^6z^2. \]
Division of \( u(x) \cdot x^{n-k} \) by \( g(x) = x^4 + x^3z^3 + x^2 + xz + z^3 \):
\[
\begin{align*}
(x^6z^2 + x^5z + x^4) & \div (x^4 + x^3z^3 + x^2 + xz + z^3) = x^2z^2 + x + z^3 + \frac{r(x)}{g(x)} \\
+ (x^6z^2 + x^5z^5 + x^4z^2 + x^3z^3 + x^2z^5) & \div x^5z^6 + x^4z^6 + x^3z^6 + x^2z^6 \\
\frac{x^5z^6 + x^4z^6 + x^3z^6 + x^2z^5}{+ (x^5z^6 + x^4z^2 + x^3z^6 + x^2 + xz^2)} & \div x^4 + x^3z^4 + x^2z^4 + xz^2 \\
+ (x^4 + x^3z^3 + x^2 + xz + z^3) & \div x^3z^6 + x^2z^5 + xz^4 + z^3 = r(x)
\end{align*}
\]
\[ -r(x) + u(x) \cdot x^{n-k} = z^3+ xz^4 + x^2z^5 + x^3z^6 + x^4 + x^5z + x^6z^2 \Rightarrow a = (z^3, z^4, z^5, z^6, 1, z, z^2). \]
The respective generator matrix reads:
\[
G = \begin{pmatrix}
    z^3 & z & 1 & z^3 & 1 & 0 & 0 \\
    0 & z^3 & z & 1 & z^3 & 1 & 0 \\
    0 & 0 & z^3 & z & 1 & z^3 & 1
\end{pmatrix}.
\]
4.2.3 Decoding of Reed-Solomon Codes

The decoding of RS codes is carried out by means of special algorithms. The usage of a syndrome table is not possible due to the enormous disc space required.

A receive vector $\mathbf{r}$ is assumed, which has resulted from the covering of the originally sent code word $\mathbf{a}$ with an error vector $\mathbf{f}$: $\mathbf{r} = \mathbf{a} + \mathbf{f}$.

Let the RS code be able to correct $t$ errors.

The approach with the algebraic decoding can then be divided into the following steps:

1. Calculation of the syndrome $S(x)$ from $\mathbf{r}$
2. Calculation of the fault locations from $S(x)$ by means of the so-called code equation
3. Calculation of the error values of $\mathbf{f}$, as RS codes are no binary codes
4. Error correction using the relation $\mathbf{a} = \mathbf{r} - \mathbf{f}$.
4.2.3 Decoding of Reed-Solomon Codes

Calculation of the syndrome:
According to definition, a code word $A$ has $2t$ consecutive coefficients in the frequency range that are equal to zero. These positions of the vector $R$ are referred to as syndrome vector or simply syndrome. The following applies:

$$r = a + f \quad \Rightarrow \quad R = A + F = (A_0 + F_0, A_1 + F_1, \ldots, A_{k-1} + F_{k-1}, \underbrace{F_{n-2t}, \ldots, F_{n-1}}_{	ext{syndrome } S}).$$

Calculation of the fault locations:
For this, the so-called code equation is derived. The code equation establishes a relationship between the syndrome $S(x)$ and the fault location polynomial $c(x)$, the zeros of which mark the fault locations. $c(x)$ is defined that way that: $c_i = 0 \; \text{for} \; f_i \neq 0$, so that the following applies:

$$c_i \cdot f_i = 0 \; \text{for} \; i = 0, 1, \ldots, n - 1$$

Thus, every fault location produces a zero / a linear factor in $C(x)$:

$$C(x) = \prod_{i, f_i \neq 0} (x - z^i)$$
### 4.2.3 Decoding of Reed-Solomon Codes

According to equations 4.31b and 4.26e, for the DFT of the product $c_i \cdot f_i$ applies:

$$c_i \cdot f_i = 0 \quad \Rightarrow \quad C(x) \cdot F(x) = 0 \mod (x^n - 1) \quad (4.50)$$

The polynomial $C(x) \cdot F(x)$ has all kinds of zeros. The degree of $C(x)$ is equal to the number of fault locations $e$. This applies under the assumption:

$$e \leq t = \frac{n - k}{2}$$

With eq. (4.49), for the fault location polynomial $C(x)$ applies:

$$C(x) = C_0 + C_1 x + \ldots + C_e x^e \quad (4.51)$$

Thus, $C(x)$ has exactly $e + 1$ coefficients, however, only the $e$ zeros of the polynomial are of interest. Therefore, a coefficient $C_i$ is arbitrary and can be normalised to 1. According to eq. (4.49), $C_e = 1$ is chosen. If $C_0 = 1$ is chosen, $C(x)$ is defined by:

$$C(x) = \prod_{i, f_i \neq 0} (1 - z^{-i} \cdot x) = 1 + C_1 x + \ldots + C_e x^e \quad (4.52)$$
4.2.3 Decoding of Reed-Solomon Codes

Before deriving the code equation, the relations between the polynomials considered so far are to be summarised first. These are the code word \( a(x) \) sent, the error polynomial \( f(x) \), the receive word \( r(x) \) as well as the fault location polynomial \( c(x) \) and the appropriate representations in the frequency range:

\[
\begin{align*}
\text{a}(x) & \quad \text{A}(x) \\
\text{f}(x) & \quad \text{F}(x) \\
\text{r}(x) & \quad \text{R}(x) \\
\text{c}(x) & \quad \text{C}(x)
\end{align*}
\]

\[
c_i \cdot f_i = 0 \quad \text{C}(x) \cdot F(x) = 0 \mod (x^n - 1)
\]
4.2.3 Decoding of Reed-Solomon Codes

In order to determine the coefficients $C_i$ by setting up and subsequently solving the code equation, it is assumed at first that the following applies: $e = t$.

Eq. (4.50) in the form $C(x) \cdot F(x) = 0 \mod (x^n - 1)$ is written as a system of equations, where the coefficients are arranged according to index sums modulo-$n$ in ascending order, that means according to the powers $x^0$ to $x^{n-1}$ ($x^n = x^0$):

\[
\begin{align*}
    C_0 F_0 &+ C_1 F_{n-1} + C_2 F_{n-2} + \ldots + C_t F_{n-t} = 0 \\
    C_0 F_1 &+ C_1 F_0 + C_2 F_{n-1} + \ldots + C_t F_{n-t+1} = 0 \\
    \vdots & \quad \vdots \quad \vdots \quad \vdots \\
    C_0 F_{n-t-1} &+ C_1 F_{n-t-2} + C_2 F_{n-t-3} + \ldots + C_t F_{n-2t-1} = 0 \\
    C_0 F_{n-t} &+ C_1 F_{n-t-1} + C_2 F_{n-t-2} + \ldots + C_t F_{n-2t} = 0 \\
    C_0 F_{n-t+1} &+ C_1 F_{n-t} + C_2 F_{n-t-1} + \ldots + C_t F_{n-2t+1} = 0 \\
    \vdots & \quad \vdots \quad \vdots \quad \vdots \\
    C_0 F_{n-2} &+ C_1 F_{n-3} + C_2 F_{n-4} + \ldots + C_t F_{n-t-2} = 0 \\
    C_0 F_{n-1} &+ C_1 F_{n-2} + C_2 F_{n-3} + \ldots + C_t F_{n-t-1} = 0
\end{align*}
\] (4.53)
The framed part of eq. (4.53) has \( t \) equations with \( t \) unknowns. The coefficients of the fault location polynomial are known there and correspond to the syndrome coefficients of \( S(x) \):

\[
S_0 = F_{n-2t}, \quad S_1 = F_{n-2t+1}, \quad \ldots, \quad S_{2t-1} = F_{n-1}.
\]

Inserting this in the framed part of eq. (4.53), the following applies:

\[
\begin{align*}
C_0 S_t & + C_1 S_{t-1} + C_2 S_{t-2} + \ldots + C_t S_0 = 0 \\
C_0 S_{t+1} & + C_1 S_t + C_2 S_{t-1} + \ldots + C_t S_1 = 0 \\
\vdots & \quad \vdots \quad \vdots \\
C_0 S_{2t-1} & + C_1 S_{2t-2} + C_2 S_{2t-3} + \ldots + C_t S_{t-1} = 0
\end{align*}
\] (4.54)

The linear equation system is uniquely solvable. The relation between the wanted coefficients \( C_i \) and the known coefficients \( S_i \) obtained from this corresponds to a cyclic convolution and, using the normalisation \( C_0 = 1 \), forms the code equation (also called Newton identity):

\[
S_j + \sum_{i=1}^{t} C_i \cdot S_{j-i} = 0 \quad \text{for} \quad j = t, \ldots, 2t-1
\] (4.55)
In general, the number of errors is unknown, with errors of little importance being more likely than errors of great importance. For $e < t$, the wanted coefficients are $C_1, C_2, \ldots, C_e$. The number of variables is equal to $e$, however, the number of equations is equal to $t$. From this results that the equation system is over-determined and different approaches for $C(x)$ are required. For this, the not normalised code equation with a variable number of errors is considered:

$$
\sum_{i=0}^{e} C_i \cdot S_{j-i} = 0 \quad \text{for} \quad j = e, \ldots, 2t - 1 \quad \text{for} \quad e = 1, 2, \ldots, t
$$

(4.56)

In matrix notation, the following representation applies:

$$
\begin{pmatrix}
S_e & S_{e-1} & \cdots & S_1 & S_0 \\
S_{e+1} & S_e & \cdots & S_2 & S_1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
S_{2t-2} & S_{2t-3} & \cdots & S_{2t-e-1} & S_{2t-e-2} \\
S_{2t-1} & S_{2t-2} & \cdots & S_{2t-e} & S_{2t-e-1}
\end{pmatrix}
\begin{pmatrix}
C_0 \\
C_1 \\
\vdots \\
C_{e-1} \\
C_e
\end{pmatrix}
= \mathbf{0} \quad \text{for} \quad e = 1, 2, \ldots, t
$$

(4.57)
4.2.3 Decoding of Reed-Solomon Codes

The search for the coefficients of the fault location polynomial is started with the most likely error event $e = 1$. If for this number of errors no solution has been found, $e$ is increased gradually until it has reached the limit of $t$. If no fault location polynomial has been found, there is a decoding failure.

Using two well-known algorithms, the solution to the code equation can be calculated with low computation effort. These are:

- the Berlekamp-Massey algorithm and
- Euclid‘s division algorithm.

If a solution for the fault location polynomial has been found, the fault locations are searched by simple testing of all kinds of zeros (Chien search): $C(x = z^i)$ for $0 \leq i \leq n-1$.

If for any $j$ from this codomain $C(x = z^j) = 0$ applies, (4.58)

then $j$ is a fault location.
4.2.3 Decoding of Reed-Solomon Codes

Calculation of the error values:

Eq. (4.50) corresponds to a cyclic convolution of the coefficients, which can be described as

\[ \sum_{i=0}^{e} C_i \cdot F_{j-i} = 0 \quad \text{for} \quad j = 0,1,\ldots,n-1 \quad \text{(Index mod n)} \]  

(4.59a)

or

\[ C_0 F_j + \sum_{i=1}^{e} C_i \cdot F_{j-i} = 0 \quad \text{for} \quad j = 0,1,\ldots,n-1 \quad \text{(Index mod n)} \]  

(4.59b)

Using

\[ C(x) = C_0 + C_1 x + \ldots + C_e x^e \]

and

\[ F(x) = F_0 + F_1 x + \ldots + F_{k-1} x^{k-1} + F_k x^k + F_{k+1} x^{k+1} + \ldots + T_{n-1} x^{n-1} \]

unknown part  

known part \( (S) \)

Resolved to \( F_j \), the error values can be calculated recursively:

\[ F_j = -C_0^{-1} \cdot \sum_{i=1}^{e} C_i \cdot F_{j-i} \quad \text{for} \quad j = 0,1,\ldots,n-1 \quad \text{(Index mod n)} \]  

(4.60)
4.2.3 Decoding of Reed-Solomon Codes

Example:
The (6,2)-RS code over $GF(7)$ with the primitive element $z = 5$ is considered. Then the parameters are: $n = 6$, $k = 2$, $d = d_{\text{H.min}} = 5$ and $t = 2$. The receive vector $r = (1,2,3,1,1,1)$ has to be decoded.

1. Calculation of the syndrome (eq. 4.47):

   \[
   r = (r_0, r_1, r_2, r_3, r_4, r_5) = (1, 2, 3, 1, 1, 1)
   \]

   \[
   R = (R_0, R_1, R_2, R_3, R_4, R_5) = (R_0, R_1, F_2, F_3, F_4, F_5) = (5, 0, 4, 6, 6, 1)
   \]

   \[
   S = (F_2, F_3, F_4, F_5) = (S_0, S_1, S_2, S_3) = (4, 6, 6, 1).
   \]

2. Calculation of the number of errors using eq. (4.57):

   Assumption of the most likely number of errors: $e = 1$ \(\Rightarrow\) $C(x) = C_0 + C_1x$.

   Normalisation (arbitrary): $C_1 = 1$ \(\Rightarrow\)

   \[
   \begin{pmatrix}
   S_1 & S_0 \\
   S_2 & S_1 \\
   S_3 & S_2 \\
   \end{pmatrix}
   \begin{pmatrix}
   C_0 \\
   C_1 \\
   \end{pmatrix}
   =
   \begin{pmatrix}
   6 & 4 \\
   6 & 6 \\
   1 & 6 \\
   \end{pmatrix}
   \begin{pmatrix}
   C_0 \\
   1 \\
   \end{pmatrix}
   =
   \begin{pmatrix}
   0 \\
   0 \\
   \end{pmatrix}
   \]
The single equations have the following solutions:

\[
6 \ C_0 + 4 = 0 \quad \Rightarrow \quad C_0 = 4, \\
6 \ C_0 + 6 = 0 \quad \Rightarrow \quad C_0 = 6 \quad \text{and} \\
1 \ C_0 + 6 = 0 \quad \Rightarrow \quad C_0 = 1.
\]

This is a contradiction, thus \( e \) has to be increased:

\[ e = 2 \quad \Rightarrow \quad C(x) = C_0 + C_1 x + C_2 x^2. \]

Normalisation: \( C_2 = 1 \implies \begin{pmatrix} S_2 & S_1 & S_0 \\ S_3 & S_2 & S_1 \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 6 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 6 \\ 6 \\ 4 \end{pmatrix} \cdot \begin{pmatrix} C_0 \\ C_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \]

Solving the equation system leads to:

\[
\begin{align*}
\text{I:} & \quad 6 \ C_0 + 6 \ C_1 + 4 = 0 \\
\text{II:} & \quad 1 \ C_0 + 6 \ C_1 + 6 = 0 \\
\text{I–II:} & \quad 5 \ C_0 - 2 = 0 \quad \Rightarrow \quad C_0 = 2 \cdot 5^{-1} = 6 \\
\text{II:} & \quad 6 \ C_1 + 5 = 0 \quad \Rightarrow \quad C_1 = -5 \cdot 6^{-1} = 5
\end{align*}
\]

Thus, the error location polynomial reads: \( C(x) = 6 + 5 \ x + x^2 \).
4.2.3 Decoding of Reed-Solomon Codes

The search for the roots (Chien search) results in:

\[ C(x = z^0 = 1) = 6 + 5 + 1 = 5, \]
\[ C(x = z^1 = 5) = 6 + 4 + 4 = 0 \Rightarrow \text{root at } z^1, \]
\[ C(x = z^2 = 4) = 6 + 6 + 2 = 0 \Rightarrow \text{root at } z^2, \]
\[ C(x = z^3 = 6) = 6 + 2 + 1 = 2, \]
\[ C(x = z^4 = 2) = 6 + 3 + 4 = 6 \text{ and} \]
\[ C(x = z^5 = 3) = 6 + 1 + 2 = 2. \]

Errors were detected at the 1st and the 2nd position of the receive vector (index beginning with 0!).

Proof: \( C(x) = (x - z^1) \cdot (x - z^2) = 6 + 5 \cdot x + x^2 \)

3. Calculation of the error values (eq. 4.60):

With \( R = (R_0, R_1, F_2, F_3, F_4, F_5) = (5, 0, 4, 6, 6, 1) \) and \( C(x) = 6 + 5 \cdot x + x^2 \) the following results:

\[ F_0 = -C_0^{-1} \cdot (C_1 \cdot F_{-1} + C_2 \cdot F_{-2}) \]
\[ = -C_0^{-1} \cdot (C_1 \cdot F_5 + C_2 \cdot F_4) \]
\[ = -6^{-1} \cdot (5 \cdot 1 + 1 \cdot 6) = 1 \cdot (5 + 6) = 4 \]
and

\[
F_1 = -C_0^{-1} \cdot (C_1 \cdot F_0 + C_2 \cdot F_{-1})
= -C_0^{-1} \cdot (C_1 \cdot F_0 + C_2 \cdot F_{-5})
= -6^{-1} \cdot (5 \cdot 4 + 1 \cdot 1) = 1 \cdot (6 + 1) = 0.
\]

With this, the complete error vector in the frequency domain reads:

\[
F = (F_0, F_1, F_2, F_3, F_4, F_5) = (4, 0, 4, 6, 6, 1).
\]

The inverse transform in the time domain results in:

\[
f = (f_0, f_1, f_2, f_3, f_4, f_5) = (0, 1, 2, 0, 0, 0).
\]

This position provides a good checking opportunity, since in \( f \) only those positions should not be equal to zero, which have been determined with the search for fault locations.

Finally, the decoding result reads:

\[
\hat{a} = r - f = (1, 2, 3, 1, 1, 1) - (0, 1, 2, 0, 0, 0) = (1, 1, 1, 1, 1, 1).
\]
4.3 Bose-Chaudhuri-Hocquenghem Codes

Like the RS codes, Bose-Chaudhuri-Hocquenghem codes (BCH codes) belong to the class of the cyclic codes and represent an important extension of the Hamming codes. Especially, they are suitable for the correction of multiple, statistically independent (single) errors. Regarding this, a differentiation between binary and non-binary BCH codes is made, where the binary BCH codes can be considered as a special case of the RS codes, whereas in turn the RS codes can be described as a special case of the non-binary BCH codes. Only the binary BCH codes, however, will be considered here.

A binary BCH code of the code word length \( n = 2^m - 1 \) is characterised that way that every code word \( \mathbf{a} \) in the time domain consists of components \( a_i \in \mathbb{GF}(2) \), whereas the code word in the frequency domain consists of components \( A_i \in \mathbb{GF}(2^m) \).

In analogy to the RS code, a binary BCH code can be defined in the frequency domain by disappearing adjacent control bit or, in the time domain, by a generator polynomial.
4.3 Bose-Chaudhuri-Hocquenghem Codes

4.3.1 Derivation of BCH Codes in the Frequency Domain

For a binary BCH code with \( a_i \in GF(2) \), the DFT transform equations 4.25a/b are simplified to

\[
A_i = a(x = z^{-i}) \quad \text{and} \quad a_j = A(x = z^j),
\]

although \( A_i \in GF(2^m) \).

If \( A_i = 0 \) applies, the polynomial \( a(x) \) has a root \( z^{-i} \) in the time domain. That means, in \( a(x) \) the coefficient \( (x - z^{-i}) \) is included:

\[
A_i = 0 = a(x = z^{-i}).
\]  
(4.62)

The theorem applies that a polynomial \( f(x) \) in \( GF(2) \) has the following important characteristic:

\[
[f(x)]^2 = f(x^2)
\]  
(4.63)

Proof via the binomial formula:

\[
(a + b)^2 = a^2 + 2ab + b^2 = a^2 + b^2, \text{ since } 2ab = 0 \text{ mod } 2.
\]

\( a \) and \( b \) themselves can be again a sum term, the single components of which are sum terms as well etc. The result is an indefinitely long sum term with the characteristic to be proven.
4.3.1 Derivation of the BCH Codes in the Frequency Domain

With eq. (4.63) applies:

\[ A_{2i} = a(x = z^{-2i}) = a(x = (z^{-i})^2) = [a(x = z^{-i})]^2 = (A_i)^2 \]
\[ A_{4i} = a(x = z^{-4i}) = a(x = (z^{-2i})^2) = [a(x = z^{-2i})]^2 = (A_{2i})^2 = (A_i)^4 \]
\[ A_{8i} = a(x = z^{-8i}) = a(x = (z^{-4i})^2) = [a(x = z^{-4i})]^2 = (A_{4i})^2 = (A_i)^8 \text{ etc.} \]

Thus, the general relation that the condition \( a_i \in GF(2) \) is fulfilled, reads:

\[
A_{(2^j \cdot i)} = (A_i)^{(2^j)}
\] (4.64)

With this, the definition reads as follows:

The binary (primitive) BCH code \( \mathbf{C} \) with the code word length \( n = 2^m - 1 \) and the built minimum distance \( d \) is defined by

\[
\mathbf{C} = \{ \mathbf{a} \mid a_i = A(z^i), \quad \text{grad } A(x) \leq n - d, \quad A_{(2^j \cdot i)} = (A_i)^{(2^j)}, \quad A_i \in GF(2^m) \} \]

with the primitive element \( z \in GF(2^m) \).

(4.65)
Example:
A BCH code over $GF(2^3)$ with $n = 2^3 - 1 = 7$ and $t = 1$, that means $d = 3$, is considered. For the components of the code word vector

$$A = (A_0, A_1, A_2, A_3, A_4, A_5, A_6) = (A_0, A_1, A_2, A_3, A_4, 0, 0)$$

applies:

for $A_0$: $A_{(2^j \cdot 0)} = A_0 = (A_0)^{(2^j)} \Rightarrow A_0 \in GF(2)$

for $A_1$: $A_{(2^j \cdot 1)} = A_{(2^j)} = (A_1)^{(2^j)} \Rightarrow A_2 = (A_1)^2, A_4 = (A_1)^4, A_{8 \mod 7} = A_1 = j(A_1)^8 \Rightarrow A_1 \in GF(2^3)$

for $A_3$: $A_{(2^j \cdot 3)} = (A_3)^{(2^j)} \Rightarrow A_6 = (A_3)^2, A_{12} = A_5 = (A_3)^4, A_{24} = A_3 = (A_3)^8 \Rightarrow A_3 \in GF(2^3)$

Using $A_5 = A_6 = 0$, however, also $A_3 = 0$. That means, although 5 digits are available in the frequency domain, only two symbols and 4 bits ($1 + 3$) of information per code word, respectively, can be transferred. Coding example:

$A_0 = 1, A_1 = z \Rightarrow A = (1, z, z^2, 0, z^4, 0, 0), A(x) = 1 + z x + z^2 x^2 + z^4 x^4$

$a_0 = A(z^0) = 1 + z + z^2 + z^2 + z^4 = 1 + z + z^2 + z^2 + z^4 = 1$

$a_1 = A(z^1) = 1 + zz + z^2z^2 + z^4z = 1 + z^2 + z^4 + z^8 = 1 + z^2 + z^2 + z = 1 \ldots$

$\Rightarrow a = (1, 1, 0, 1, 0, 0, 0)$ (comp. respective RS coding on p. 214f.)
The aforementioned example can be realised using the adjoining circuit.
4.3.2 Derivation of the BCH Codes in the Time Domain

In case that the component $A_i$ of a vector $A$ as well as their conjugated components are equal to zero, the product of the respective roots forms in the time domain a minimum polynomial with coefficients from $GF(2)$ according to eq. (4.17) and (4.22). If $A$ has consecutive components $A_i, A_{i-1}, \ldots$, that are equal to zero, the product of the minimum polynomials generates the generator polynomial of the binary BCH code. Then the definition reads:

\[ K_j \text{ are the cyclotomic fields in terms of a number } n = 2^m - 1, \text{ } z \text{ is a primitive element in } GF(2^m) \text{ and } M \text{ the union of sets of the cyclotomic fields: } M = \{K_1 \cup K_2 \cup \ldots\}. \text{ A primitive BCH code of the length } n = 2^m - 1 \text{ is defined by the generator polynomial:} \]

\[ g(x) = \prod_{i \in M} (x - z^i) = m_{j_1}(x) \cdot m_{j_2}(x) \cdot \ldots \]  
\hspace{1cm} (4.66)

In case that $d - 1$ consecutive numbers in $M$ exist, the built minimum distance is $d$. The minimum distance actually achieved can be longer.

The BCH code has the dimension $k = n - \text{grad (g(x))}$. 
4.3.2 Derivation of the BCH Codes in the Time Domain

Example:
On the basis of the $GF(2^4)$, that means $n = 2^4 - 1 = 15$, structures for different minimum distances to be built are considered (compare p. 184).

I: $g(x) = \prod_{i \in \{K_1, K_2\}} (x - z^i) = m_1(x) = x^4 + x + 1$

With $M = \{K_1, K_2\} = K_1 = \{1, 2, 4, 8\}$ the built minimum distance is $d = 3$, since $d - 1 = 2$ consecutive numbers are contained in $M$.
The dimension is $k = n - \text{grad } (g(x)) = 15 - 4 = 11$.

II: $g(x) = \prod_{i \in \{K_1, K_2, K_3, K_4\}} (x - z^i) = m_1(x) \cdot m_3(x)$

$= (x^4 + x + 1) \cdot (x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1$

With $M = \{K_1 \cup K_3\} = \{1, 2, 3, 4, 6, 8, 9, 12\}$ the built minimum distance is $d = 5$, since $d - 1 = 4$ consecutive numbers are contained in $M$.
The dimension is $k = n - \text{grad } (g(x)) = 15 - 8 = 7$. 
III:  \[ g(x) = \prod_{i \in \{K_1, K_2, \ldots, K_6\}} (x - z^i) = m_1(x) \cdot m_3(x) \cdot m_5(x) \]

\[ = (x^4 + x + 1) \cdot (x^4 + x^3 + x^2 + x + 1) \cdot (x^2 + x + 1) \]

\[ = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \]

With \( M = \{K_1 \cup K_3 \cup K_5\} = \{1, 2, 3, 4, 5, 6, 8, 9, 10, 12\} \) the built minimum distance is \( d = 7 \), since \( d - 1 = 6 \) consecutive numbers are contained in \( M \).

The dimension is \( k = n - \text{grad} (g(x)) = 15 - 10 = 5 \).

Combining the cyclotomic fields, it has to be taken into account that it is about the union of sets in which the same element cannot be found repeatedly. Therefore, e.g. for the first example (I) the following applies:

\[ M = \{K_1 \cup K_2\} = K_1, \text{ da } K_1 = K_2. \]
4.3.3 Aspects of Coding / Decoding of BCH Codes

The BCH code is a linear cyclic code that, in analogy to section 3.3, can be coded in two ways using the generator polynomial:
– unsystematically by multiplication by the generator polynomial and
– systematically by division by the generator polynomial.

Unlike the RS codes, the shift of the test frequencies the BCH codes. The decoding is similar to the decoding with the RS codes. It is the advantage of this case that with binary codes no calculation of the error values is required.
5 Convolutional Codes

Beside the block codes, the convolutional codes are the second important group of error-correction codes. Instead of the generation of code words block by block, the code words are generated by convolution of an information sequence with a set of generator coefficients. Though they have a simpler mathematical structure compared to the block codes, no analytical methods for construction exist for them; they are found by computer simulations. However, convolutional codes can process reliability information of code symbols much better than block codes. In return, they are highly sensitive to burst errors. Normally, the codes are binary.

First publications regarding convolutional codes were launched in 1955 by Elias. However, they could only be used in practice after decoding methods had been developed. Among those are the methods of the sequential decoding by Fano (1963) and the maximum Likelihood decoding algorithm by Viterbi (1967) which can be realised in practice.
Soft and hard decision:
With decoding, convolutional codes are able to use hard decision as well as soft decision symbols. Hard decision means that the continuous received signal is converted by a threshold decision into a sequence of possible transmit symbols after demodulation. With the soft decision, either the analog signal is passed on or the quantisation values (interim values) are applied (see figure). Therefore, a more reliable decoding is possible.
5.1 Definition of Convolutional Codes

In analogy to the block codes, information and code sequences are classified into blocks of the length $k$ and $n$, respectively. Here, however, they are indexed by the block number $r$: $u_r = (u_{r,1}, u_{r,2}, \ldots, u_{r,k})$, $a_r = (a_{r,1}, a_{r,2}, \ldots, a_{r,n})$. (5.1a/b)

Usually, the symbols are binary: $a_{ij}, u_{ij} \in GF(2) = \{0,1\}$.

The assignment of $a_r$ to $u_r$ is, in contrast to the block codes, not memoryless. The active code block is defined by the current information block and by $m$ preceding ones:

$$a_r = a_r(u_r, u_{r-1}, \ldots, u_{r-m}) \quad (5.2)$$

The coder is linear (comp. section 5.4), that means the code bits result in linear combinations of the information bits. The formal description is made by using the generator coefficients $g_{\kappa,\mu,\nu} \in GF(2)$ with $1 \leq \kappa \leq k$, $0 \leq \mu \leq m$ and $1 \leq \nu \leq n$, so that the code bit subsequences result in convolutions of the information bit sequences with the generator coefficients:

$$a_{r,\nu} = \sum_{\mu=0}^{m} \sum_{\kappa=1}^{k} g_{\kappa,\mu,\nu} \cdot u_{r-\mu,\kappa} \quad (5.3)$$
The code rate of the convolutional code is: \[
R_C = \frac{k}{n}.
\] (5.4)

The size \(m\) is referred to as memory and \(L = m + 1\) as the constraint length. In addition to the code rate, it influences the performance as well as the decoding effort. Formally, block codes are a special case of the convolutional codes with \(m = 0\).

Whereas with block codes \(k\) and \(n\) are usually large, typical values with convolutional codes are: \(k = 1, 2;\) \(n = 2, 3, 4;\) \(m \leq 8\).

For the realisation of the memory, \(k \cdot (m + 1)\) locations are required (comp. figure on the next page). In literature, representations also exist putting the information sequence \(u_r\) directly to the adder without intermediate storage. In this case, the number of locations only amounts to \(k \cdot m\).
5.1 Definition of Convolutional Codes

Block diagram of a general \((n,k,m)\) convolutional coder:

A connection corresponds to \(g_{\kappa,\mu,\nu}\).
Example: (2,1,2) convolutional coder

The parameters are:
\[ n = 2, \quad k = 1, \quad m = 2, \]
\[ R_C = 1/2. \]

The code bits are calculated from:
\[ a_{r,1} = u_r + u_{r-1} + u_{r-2}; \]
\[ a_{r,2} = u_r + u_{r-2}. \]

Example of a coding sequence:
\[ u = (1,1,0,1,0,0,...); \]
\[ a = (11,01,01,00,10,11,...). \]

This example will be often referred to in this section.
5.1 Definition of Convolutional Codes

Example: (3,2,2) convolutional coder

The parameters are:
\[ n = 3, \quad k = 2, \quad m = 2, \quad R_C = 2/3. \]

The code bits are calculated from:
\[
\begin{align*}
  a_{r,1} &= u_{r,2} + u_{r-1,1} + u_{r-2,2}; \\
  a_{r,2} &= u_{r,1} + u_{r-1,1} + u_{r-1,2}; \\
  a_{r,3} &= u_{r,2}.
\end{align*}
\]

Example of a code sequence:
\[
\begin{align*}
  u &= (11,10,00,11,01,...); \\
  a &= (111,110,010,111,001,...).
\end{align*}
\]
5.1 Definition of Convolutional Codes

Description of the polynomial:
The restriction $k = 1$ is to be applied here. From this, tremendous simplifications in the description result.
To the generator coefficients $g_{k,v}$ the generator polynomials

$$g_v(x) = \sum_{\mu=0}^{m} g_{\mu,v} \cdot x^\mu$$

(5.5)

are assigned. The information bit sequence is characterised by a power series and the code block sequence by a power series vector:

$$u(x) = \sum_{r=0}^{\infty} u_r x^r$$

and

$$a(x) = (a_1(x), a_2(x), \ldots, a_n(x)) \quad \text{with} \quad a_v(x) = \sum_{r=0}^{\infty} a_{r,v} x^r$$

(5.6)

(5.7a/b)
5.1 Definition of Convolutional Codes

The convolutional coding corresponds to a polynomial multiplication according to:
\[ a_\nu(x) = u(x) \cdot g_\nu(x) \quad \text{for} \quad \nu = 1, \ldots, n \]  (5.8a)

\[ (a_1(x), a_2(x), \ldots, a_n(x)) = u(x) \cdot (g_1(x), g_2(x), \ldots, g_n(x)) \]  (5.8b)

\[ \Leftrightarrow a(x) = u(x) \cdot G(x) \]  (5.8c)

with the generator matrix:
\[ G(x) = (g_1(x), g_2(x), \ldots, g_n(x)) \]  (5.9)

Then the definition of the convolutional codes with the polynomial multiplication reads:
\[
\mathbf{C} = \left\{ u(x) \cdot G(x) \mid u(x) = \sum_{r=0}^{\infty} u_r x^r, \ u_r \in \{0,1\} \right\} \]  (5.10)

For the memory applies:
\[ m = \max_{1 \leq \nu \leq n} \text{grad}(g_\nu(x)) \]  (5.11)
Example:

For the (2,1,2) convolutional coder on page 245 applies:
\[ G(x) = (g_1(x), g_2(x)) = (1 + x + x^2, 1 + x^2). \]

The coding example can be described as:
\[
\begin{align*}
u &= (1, 1, 0, 1, 0, 0, ...) 
\iff u(x) = 1 + x + x^3 + ...
\end{align*}
\]
and
\[
\begin{align*}a(x) &= (a_1(x), a_2(x)) = u(x) G(x) \\
&= ((1 + x + x^3)(1 + x + x^2), (1 + x + x^3)(1 + x^2)) \\
&= (1 + x^4 + x^5, 1 + x + x^2 + x^5) \\
&= (a_{0,1}x^0 + a_{1,1}x^1 + a_{2,1}x^2 + a_{3,1}x^3 + a_{4,1}x^4 + a_{5,1}x^5, \\
&\quad a_{0,2}x^0 + a_{1,2}x^1 + a_{2,2}x^2 + a_{3,2}x^3 + a_{4,2}x^4 + a_{5,2}x^5).
\end{align*}
\]

From this, the code sequence in vector notation results by sorting the coefficients:

\[
\begin{align*}a &= (a_{0,1}a_{0,2}, a_{1,1}a_{1,2}, a_{2,1}a_{2,2}, a_{3,1}a_{3,2}, a_{4,1}a_{4,2}, a_{5,1}a_{5,2}).
\end{align*}
\]

Result: \(a = (11, 01, 01, 00, 10, 11)\).
5.2 Special Code Classes

Systematic convolutional codes:
The information bits are adopted directly to the code bits.

Block diagram of a general systematic convolutional coder:

Non-systematic convolutional codes normally have more favourable features.
5.2 Special Code Classes

Time-phased convolutional codes:
After $L$ information blocks, $m$ known blocks (tail bits – normally zeros) are inserted. Thus, the code word is closed (scheduling) and the memory is set to the zero state. This corresponds to a block structure (block code). Then the code rate is:

$$R_{C,\text{terminiert}} = \frac{L \cdot k}{(L + m) \cdot n}$$  \hspace{1cm} (5.12)

Note: Do not confuse the $L$ combined blocks for the scheduling with the constraint length (see page 243)!
5.2 Special Code Classes

Punctured convolutional codes:

$P$ code blocks ($P$ is called puncturation length) are combined. From this, $l$ code bits are deleted according to a puncturation scheme (puncturation matrix $P$). Then the code rate is given by:

$$R_{C, \text{punktier}} = \frac{P \cdot k}{P \cdot n - l} \quad (5.13)$$

It must apply: $R_{C, \text{punktier}} < 1 \Rightarrow l < P \cdot (n - k)$. \quad (5.14)

Normally, they are applied with information blocks of the length $k = 1$. With the same code rate and $k > 1$, they are only slightly less efficient than non-punctured convolutional codes. The decoding effort hardly changes compared to the non-punctured convolutional codes. Switch is possible without high additional effort.

Especially the RCPC codes (rate compatible punctured convolutional codes) are of great practical importance. Their code families are derived from a mother code that way (non-punctured convolutional code), that the higher-rate codes are contained in the low-rate codes.
5.2 Special Code Classes

Example:
It is \( l = 1 \).

The code example reads as follows:
\[ u = (1, 1, 0, 1, 0, 0, \ldots) \]
\[ a = (11, 01, 01, 00, 10, 11, \ldots) \]
\[ a_{\text{punkt}} = (11, 0x, 01, 0x, 10, 1x, \ldots). \]
'\( x \)' identifies the digits that were deleted:
\[ a_{\text{punkt}} = (11, 0, 01, 0, 10, 1, \ldots). \]

Code rate: \( R_{C, \text{punktiert}} = \frac{2 \cdot 1}{2 \cdot 2 - 1} = \frac{2}{3} \)

This is NOT a matter of matrix multiplication!
5.3 Representation of Convolutional Codes

Besides the representation via the shift register circuit or via the generator polynomials and the polynomial multiplication as convolution, the following three representations are familiar:

– code tree
– state diagram
– netword diagram (Trellis diagram)

**Code tree:**
The code is spanned as a tree, with the nodes corresponding to the possible memory states. From every node, $2^k$ branches go off, where the information and respective code bits are depicted.
The disadvantage of this representation is that with every coding step the number of nodes increases exponentially.
The number of possible paths is:

$$N_{Pfade} = \left(2^k\right)^{N_{\text{Block}}} = 2^k \cdot N_{\text{Block}}$$  \hspace{1cm} (5.15)

where $N_{\text{Block}}$ is equal to the number of processed blocks and thus to the depth of the tree.
Example: Code tree for the (2,1,2) convolutional coder (see page 245). A path corresponds to an information and code sequence, respectively. It is marked here: \( u = (1\ 1\ 0\ 1\ \ldots), \quad a = (11\ 01\ 01\ 00\ \ldots). \)
5.3 Representation of Convolutional Codes

State diagram:
In this case, it is about (in contrast to the code tree) a repetition-free description of the convolutional code. The time information is missing in the state diagram. Thereby, the output block depends on the input block and the memory content, whereas the new memory content depends on the input block and the old memory content.
Regarding this, the minimum number of coding steps required to get from the state A to the state B can be read off. Thereby, the number of states is:

\[ N_{\text{Zustand}} = (2^k)^m = 2^{k \cdot m} \quad (5.16) \]

At every state, \(2^k\) transitions exist so that all in all

\[ N_{\text{Zustand}} \cdot 2^k = 2^{k \cdot m} \cdot 2^k = 2^{k \cdot (m+1)} \quad (5.17) \]

transitions exist. With large parameters, this representation can become unclear again.
5.3 Representation of Convolutional Codes

Example: (2,1,2) convolutional code (see page 245).

The parameters are:
\[ n = 2, \quad k = 1, \quad m = 2, \]
\[ R_C = 1/2. \]

The used notations are:
- memory content = state = \[ u_{r-1} u_{r-2} \]
- input bits (output bits) = \[ u_r (a_{r,1} a_{r,2}) \]
5.3 Representation of Convolutional Codes

Example: (3,2,2) convolutional coder (see page 246).

The parameters are:
\[ n = 3, \quad k = 2, \quad m = 2, \]
\[ R_C = \frac{2}{3}. \]

This example shows the problems with unfavourable parameter values. Though all arrows are included in the diagram, the information about the input and output bits was renounced.
Network diagram – Trellis diagram:
It corresponds to a state diagram with a time component and represents the basis for decoding. A receive sequence can be compared with all possible paths block by block and thus the most likely code sequence of a receive word can be determined.

Example:
Trellis segment with all possible transitions for the (2,1,2) convolutional coder (see page 245).
5.3 Representation of Convolutional Codes

Example:
Trellis diagram for the (2,1,2) convolutional coder (see page 245) with a coding example for the scheduling.
The time-phased code word reads: \( a = (11,01,01,00,10,11,00,00) \).
5.4 Features of Convolutional Codes

Linearity:
The convolution is a linear operation. Thus, the convolutional code is **linear**.

Minimum distance and free distance:
According to the evaluation of block codes, the minimum distance is unsuitable for convolutional codes, because code words of convolutional codes do not have a defined length. Instead of that, a so-called **free distance** $d_f$ is defined. It is to represent the minimum Hamming distance between arbitrary code sequences with different information sequences:

$$
\begin{align*}
    d_f &= \min \{d_H(a_1,a_2) \mid u_1 \neq u_2\} \\
    \text{(5.18a)}
\end{align*}
$$

Since the convolutional code is a linear code, the distance to the zero sequence is sufficient for the calculation. Thus, $d_f$ corresponds to the minimum weight of all possible code sequences:

$$
\begin{align*}
    d_f &= \min \left\{ w_H(u(x) \cdot G(x)) \mid u(x) = \sum_{i=0}^{\infty} u_i x^i, u(x) \neq 0 \right\} \\
    \text{(5.18b)}
\end{align*}
$$
Fundamental path, distance function and distance profile:
A path leaving and returning to the zero state without touching it in the meantime is called fundamental path. At least one path exists here which forms a code sequence with a minimum weight.
The paths can be determined via the Trellis or the state diagram. For the latter, separation at the zero state is recommended.

Example (see next page): the shortest path from 00 to 00 is:
\[ \begin{array}{c}
00 \rightarrow 10 \rightarrow 01 \rightarrow 00 \\
\Rightarrow w_H = 5.
\end{array} \]  

Two paths can be found for \( w_H = 6 \):
\[ \begin{array}{c}
00 \rightarrow 10 \rightarrow 11 \rightarrow 01 \rightarrow 00 \\
\Rightarrow w_H = 6;
\end{array} \]  
\[ \begin{array}{c}
00 \rightarrow 10 \rightarrow 01 \rightarrow 10 \rightarrow 01 \rightarrow 00 \\
\Rightarrow w_H = 6.
\end{array} \]
State diagram separated at the zero state for the (2,1,2) convolutional coder (see page 245):
For further approach, two parameters are introduced.

The states are described by the formal parameter $Z_i$.

For the description of the distance, the formal parameter $D$ is introduced, the power of which states the number of bits different from zero.
By means of the introduced parameters, the following equation system can be set up:

\[ Z_b = D^2 Z_a + D^0 Z_c; \quad Z_c = D^1 Z_b + D^1 Z_d; \]
\[ Z_d = D^1 Z_b + D^1 Z_d; \quad Z_e = D^2 Z_c. \]

The solution of the equation system provides the distance function:

\[ T(D) = \frac{Z_e}{Z_a} = \frac{D^5}{1 - 2D} \]

From the power series expansion

\[ \frac{1}{1 - x} = 1 + x + x^2 + x^3 + x^4 + \ldots \quad \text{for} \quad |x| < 1 \quad (5.19) \]

the following results:

\[ T(D) = D^5 \cdot \frac{1}{1 - 2D} = D^5 + 2D^6 + 4D^7 + 8D^8 + 16D^9 + \ldots \]

The result shows the path with the minimum weight 5, so that the free distance \( d_f = 5 \). Furthermore, the two paths with the weight 6 can be read off (see page 262). There are 4 more paths with the weight 7, 8 paths with the weight 8, 16 paths with the weight 9 etc.
The minimum distance $d_{\text{min}}$ is dependent on the number of coding steps $i$. Considering $d_{\text{min}}$ as a function of $i$, the distance profile of a convolutional code is obtained. In this case, the statement is of interest how many coding steps are required for a code word to reach the free distance by all means.

Example: (2,1,2) convolutional code (see page 245)

After $i = 6$ steps at most, the free distance is reached (comp. fundamental way with the lowest weight, where only three steps form the free distance; page 262).
Optimal convolutional code:
A code with the maximum free distance at a given code rate and memory is referred to as optimal code.

Table of optimal convolutional codes:

<table>
<thead>
<tr>
<th>$m$</th>
<th>$d_f$</th>
<th>$g_1(x)$</th>
<th>$g_2(x)$</th>
<th>$g_3(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>$1 + x + x^2$</td>
<td>$1 + x^2$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>$1 + x + x^3$</td>
<td>$1 + x + x^2 + x^3$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>$1 + x^3 + x^4$</td>
<td>$1 + x + x^2 + x^4$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>$1 + x^2 + x^4 + x^5$</td>
<td>$1 + x + x^2 + x^3 + x^5$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>$1 + x^2 + x^3 + x^5 + x^6$</td>
<td>$1 + x + x^2 + x^3 + x^6$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>$1 + x + x^2$</td>
<td>$1 + x^2$</td>
<td>$1 + x + x^2$</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>$1 + x + x^3$</td>
<td>$1 + x + x^2 + x^3$</td>
<td>$1 + x^2 + x^3$</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>$1 + x^2 + x^4$</td>
<td>$1 + x + x^3 + x^4$</td>
<td>$1 + x + x^2 + x^3 + x^4$</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>$1 + x^2 + x^4 + x^5$</td>
<td>$1 + x + x^2 + x^3 + x^5$</td>
<td>$1 + x^3 + x^4 + x^5$</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>$1 + x^2 + x^3 + x^5 + x^6$</td>
<td>$1 + x + x^4 + x^6$</td>
<td>$1 + x + x^2 + x^3 + x^4 + x^6$</td>
</tr>
</tbody>
</table>
Catastrophic coder:
If a finite number of transmission errors results in an infinite number of decoding errors, this is called catastrophic error propagation. This occurs if two output sequences of a coder differ only in few bits, although the input sequences are completely different. Thus, a small number of transmission errors in these positions can lead to a falsification of one code word into the other.

This is always possible if at least two memory states exist which are not abandoned if the respective information does not change and which thereby generate identical code sequences. At the same time, this is a criterion for the detection of a catastrophic coder and can be read off easily from a state diagram (see example next page).

For the case $k = 1$, a catastrophic coder can be avoided, if the generator polynomials do not have a common divisor (comp. example):

$$\text{ggT}(g_1(x), g_2(x), \ldots, g_n(x)) = 1.$$  \hfill (5.20)
5.4 Features of Convolutional Codes

Example:

The two states (00) and (11) are not abandoned in case of consistent information and generate the same code sequences. Thus, this is about a catastrophic coder.

From the generator polynomials, the same statement can be derived, since $g_1(x)$ is a divisor of $g_2(x)$:

\[
g_1(x) = 1 + x; \\
g_2(x) = 1 + x^2 = (1 + x) \cdot (1 + x).
\]
5.5 Decoding Using the Viterbi Algorithm

Viterbi decoding is a method that requires small effort and with which, technically, memory depths of up to \( m \approx 10 \) can be realised. It provides a Maximum-Likelihood decoding and thus is optimal.

For the derivation of the method, a terminated code sequence \( a \) of the length \( N \) and a receive sequence \( r \) is to be known. The objective of the decoder is to estimate a code sequence complying with the transmit sequence \( a \) as much as possible, that means the Likelihood function \( p(r|a) \) maximises:

\[
p(r|a) = \prod_{i=0}^{N-1} p(r_i|a_i) \quad (5.21)
\]

Alternatively, the maximum of the Log-Likelihood function \( \log p(r|a) \) is searched for, so that for the memoryless channel the transition probability \( p(r|a) \) can be replaced by a sum of the transition probabilities of the single symbols \( p(r_i|a_i) \):

\[
\log p(r|a) = \sum_{i=0}^{N-1} \log p(r_i|a_i) \quad (5.22)
\]
Thus, the Maximum-Likelihood detection of the maximisation corresponds to metrics with the general form:

\[
M(r \mid a) = \alpha \cdot \sum_{i=0}^{N-1} \log p(r_i \mid a_i) + \beta
\]

\[
= \sum_{i=0}^{N-1} \left( \alpha \cdot \log p(r_i \mid a_i) + \beta_i \right) = \sum_{i=0}^{N-1} \mu(r_i \mid a_i) \quad (5.23)
\]

using the metrics increment \( \mu(r_i \mid a_i) \).

Example: Using \( a_i \in \{x_1, x_2\} = \{-1,+1\} \) and \( r_i \in \{y_1, y_2\} = \{-1,+1\} \) for the symmetric binary channel applies:

\[
p(r_i \mid a_i) = \begin{cases} 
1 - p_{\text{err}} & \text{for } a_i = r_i \\
p_{\text{err}} & \text{for } a_i \neq r_i 
\end{cases}
\]

\[
= \begin{cases} 
1 - p_{\text{err}} & \text{for } a_ir_i = +1 \\
p_{\text{err}} & \text{for } a_ir_i = -1 \quad (5.24)
\end{cases}
\]
Choosing for the free parameters $\alpha$ and $\beta_i$ the values

$$\alpha = 2 / \log \frac{1 - p_{\text{err}}}{p_{\text{err}}} \quad \text{and} \quad \beta_i = -1 - \alpha \log p_{\text{err}} \quad (5.25a/b)$$

then the metrics increment just reads:

$$\mu(r_i | a_i) = \begin{cases} +1 & \text{für } a_ir_i = +1 \\ -1 & \text{für } a_ir_i = -1 \end{cases} = a_ir_i \quad (5.26a)$$

The maximisation of the Viterbi metrics

$$\mu(r_i | a_i) = a_ir_i \quad (5.26b)$$

is equivalent to a minimisation of the Hamming distance or to the maximisation of the number of agreements.

This, however, is only one possible approach. Other definitions of metrics are possible.
5.5 Decoding Using the Viterbi Algorithm

Viterbi algorithm:
Decoding is carried out by means of the Trellis diagram. The objective is to compare all possible code sequences with the receive sequence. With this algorithm, this takes place calculating the Viterbi metrics.

Approach: With the initial state, the metrics is set to 0. Then, in every step, to each of the states reached via different paths a metrics $z$ is assigned, that is calculated from the addition of the metrics of the previous state and a metrics increment.

From all $2^k$ paths arriving at a state only the path is retained that shows the largest metrics. All other $2^k - 1$ paths are discarded, since they can never achieve a larger metrics. If several paths with the same metrics exist, a random sample has to be effected. Eventually, the decoding result is the path with the largest metrics.
In principle, the rule for the metrics increment is arbitrary. With the hard-decision decoding, eq. (5.26b) is normally used or simply the number of agreements between the code sequence of a part of a path and the respective receiver is selected. With the soft-decision decoding, e.g., the distance between a receive value and the code symbols can be selected, with this method being more reliable than the hard decision decoding (comp. page 241). In this case, it is advantageous that the probability of equal metrics at a state is lower.

During decoding, a definite number of paths has to be saved. Since the number of paths to be saved can become very large (before one path remains only; see the following example), the length of the paths to be saved is restricted to $5 \cdot k \cdot m$. This restriction is based on experience. Thereby, the need on storage space is approx. the product of the number of states and the path length ($2^{k \cdot m} \cdot 5 \cdot k \cdot m$ bit).
Example: (2,1,2) convolutional code (see page 245)

Let the information vector be
\[ u = (1,1,0,1,0,0,0,1,0,...). \]

From this, a code sequence vector results
\[ a = (11,01,01,00,10,11,00,11,10,...). \]

With the transition, the error vector
\[ f = (00,10,00,00,01,00,00,00,00,...) \]
occurs so that the receive vector reads:
\[ r = (11,11,01,00,11,11,00,11,10,...). \]

The receive vector is to be decoded using the Viterbi algorithm. Thereby, a hard-decision decoding is to be carried out and the sum of agreements is to be used as metrics.

In the following, the single decoding steps of the 9 known sequences all in all are represented in detail. The paths with the largest metrics are highlighted in purple.
1st step:
receive sequence  11

input 1: ----- input 0: ---

At the beginning the metrics is equal to 0 and increases per step with the number of agreements of the receive sequence with the possible code sequences.
5.5 Decoding Using the Viterbi Algorithm

2nd step:

receive sequence 11 11

input 1: ----- input 0: ---

After \( m = 2 \) steps, all states are achieved. With the next step, every state will be achieved over more than one path (see next page).
3rd step:

receive sequence 11 11 01

Per arriving path a metrics exist, with the upper metrics belonging to the upper path and the lower metrics belonging to the lower path. In each case, the path with the lower metrics can be deleted.
5.5 Decoding Using the Viterbi Algorithm

4th step:
receive sequence  11  11  01  00

Deleting the path, the representation becomes clearer. In this step, the two lower states each have the same metrics so that in each case one path is deleted randomly (comp. next page).

input 1: ------ input 0: ——

Deleting the path, the representation becomes clearer. In this step, the two lower states each have the same metrics so that in each case one path is deleted randomly (comp. next page).
5.5 Decoding Using the Viterbi Algorithm

5th step:
receive sequence 11 11 01 00 11

The previous path cancellations cause that only one single transition from the 0th state to the 1st state remains, that means is the most likely one. Thus, this single sequence is already decoded.
6th step:

Virtually, the code sequence (11) and the appropriate information sequence (1), respectively, can be output by the decoder and the representation can move by one state. In this example, it is still retained.
5.5 Decoding Using the Viterbi Algorithm

7th step:
receive sequence 11 11 01 00 11 11 00

From the maximum metrics it can be seen how many errors must at least have occurred so far (in this case, maximum 14 agreements are possible, 12 do exist at most, thus: at least 2 errors; comp. error vector).
8th step:
receive sequence: 11 11 01 00 11 11 00 11

In case of a time-phased code, the number of states would become smaller to the end (final phase), until after the last step only the initial state remains (comp. page 260).
5.5 Decoding Using the Viterbi Algorithm

9th step:
receive sequence 11 11 01 00 11 11 00 11 10

input 1: ------ input 0: —

The end of the known sequence is achieved. Up to the 4th state, the result can be displayed; up to the last state this would only be possible if the receive sequence was finished. However, in practice it goes on!
6 Advanced Coding Techniques

In the course of this lecture, fundamentals and basic codes in the field of channel coding have been dealt with in detail. These are widely spread also in practice, however, often as complex implementations. Particularly, among those are concatenated codes with interleavers, the principles of which are to be shortly introduced here.

These „basic codes“ are, of course, not the only codes developed in the past 40 years. Since the beginning of the 1990s, first and foremost two codes have been investigated, which are able to approach the Shannon limit very closely (less than 1dB) with regard to their correction capability. On the one hand, these are the turbo codes virtually derived from experiments and, on the other hand, the Low-Density-Parity-Check codes (LDPC codes), emerged in the 1960s. The latter, however, had been disregarded until the beginning of the 1990s since they were very complex compared to the best available technology at that time. The principle of these two codes is to be outlined.
Concatenation of codes:
With the concatenation of codes, 2 or several codes are connected in series (see figure next page). The motivation for the use of this construction consists in the exploitation of different complementary code characteristics, e. g. for error correction and detection or for single error and burst error correction.

One classical example consists in the correction of single errors by the inner code, whereas burst errors are corrected by the outer code. A method of this kind is applied e.g. in European mobile radio systems such as GSM. By concatenation, very efficient codes can be built. In many examples of use, the inner code is implemented using a convolutional code for single error correction and the outer code is implemented using a Reed-Solomon code for burst error correction.
Principle of concatenated codes:

The principle of the interleaver is explained on the following pages.
Interleaving:
On a transmission channel, burst errors can occur that are beyond the correction capability of the applied channel codes. It is the task of the interleaver to re-sort the code word sequence to be sent, so that at the receiver after having passed the deinterleaver several quasi-single errors result from one burst error, which are then spread to several code words and can be corrected.

The example (see next page) shows this method as a matrix, in which the code words are arranged line by line, however, are transmitted column by column. Thus, the burst error occurring in the 4th column is distributed to several code words as single error.

The disadvantage of the interleaver is that, due to the intermediate storage of the code words, an additional time delay emerges, which increases depending on the size of the interleaver. This causes the most problems with time-critical systems such as voice transmission.
6 Advanced Coding Techniques

Block interleaver:

- code word 1
- code word 2
- code word 3
- code word 4
- \ldots
- code word \( k \)

Blocks sent: 1, 2, 3, 4, 5, \ldots, \( n \)
Coding of the signaling information with GSM (BCCH channel):
The outer coder consists of a (224,184) block code, where 4 tail bits for the
terminated (2,1,4) convolutional coding (inner coder) are attached. The tail
bits are interpreted as information by the convolutional code; therefore a
code rate of 1/2. The 456 bits are cut into bursts and transmitted after interleaving.
Turbo codes:
The first paper was published in 1993 by the two French electrical engineers Claude Berrou and Alain Glavieux. The original idea resulting in the turbo codes was the use of feedback of signals commonly used in electrical engineering (e.g. with amplifiers). Turbo codes are characterised by their error correction property, which is close to the Shannon limit (approx. 0.5 – 0.7 dB).

The turbo codes were developed experimentally and not by mathematical-theoretical approaches. Therefore, their performance was checked critically for years, before they were taken into consideration for application in modern systems. Today, they are e.g. a part of UMTS with data transmission. For voice transmission, however, turbo codes are non-applicable due to the delay time. In this field, convolutional codes are used. In space communication, however, turbo codes are now applied.
Principle of coding with turbo codes:

The information bits are transmitted threefoldly over the channel: once without error protection \((i)\) and twice with the error protection of a convolutional code \((c_1\) and \(c_2)\), with one of the coding blocks receiving an information sequence split by an interleaver as input. The multiplexer then assembles the three signals to one signal.
Principle of decoding with turbo codes:

The receive signal consisting of the three parts sent is splitted and transmitted to the decoding blocks, with these receiving their signals to be decoded as soft-decision information as well as a copy of the uncoded signal. From this, suggestions for the correctly decoded bits are derived, which are passed on over the signals $e_1$ and $e_2$ to the respective other block. The decoding is not completed until the suggestions of both blocks are equal (iterative method). From experience, the number of iteration steps is between 4 and 10.
Low-Density-Parity-Check codes (LDPC codes):
In 1960, Robert G. Gallager published the idea of LDPC codes within the scope of his doctoral thesis. Due to its complexity (among other things, iterative decoding was only possible using mainframe computers at that time), they had been neglected with regard to practical applications for a long time. In terms of technology, the commercial application of highly integrated coder circuits to obtain efficient codes has only been possible for a few years. – In 1995, LDPC codes were rediscovered by MacKay and Neal. With a code rate of 0.5 and a block length of 10,000,000, the code approaches the Shannon limit up to 0.04 dB. This example rather to be considered as academic shows at least the capability of the LDPC codes. Therefore, finding suitable design methods for efficient codes (especially of short block lengths) is currently a major research topic. Nevertheless, these codes have already been practically included. In the ETSI standard for DVB-S2, for example, the concatenation of LDPC codes with a BCH code is used for channel coding.
Data rate and Shannon limit for digital satellite television: variants of DVB-S2

- **0.5** Ru [bit/s] per unit Rs
- **4.5** Shannon limit
- **18 dB**
- **QPSK**
- **8PSK**
- **16APSK**
- **32APSK**

**Modulation constrained Shannon limit**
Definition of LDPC codes over an arbitrary alphabet \( A \):
An LDPC code is a linear block code of the block length \( n \), the check matrix \( H \) of which has the following characteristics:

1. The check matrix \( H \) is sparsely populated, that means a large part of the elements \( h_{ml} \) is equal to 0.
2. Each column \( l \) of the matrix \( H \) has a small number of \( N_l \) symbols from \( A \) with \( N_l > 1 \).
3. Each line \( m \) of the matrix \( H \) has a small number of \( N_m \) symbols from \( A \) with \( N_m > 1 \).

LDPC codes over \( A = GF(2^b) \) are of particular practical interest. Since LDPC codes assigned to the class of linear block codes, they also have all their characteristics. Among those are the presentation by means of a generator matrix, coding by means of a matrix-vector multiplication, the possibility to convert non-systematic into systematic LDPC codes, the determination of the minimum distance by means of the minimum Hamming weight etc. (see chapter 3).
The figure shows the sparsely populated (900×1200) check matrix of a binary LDPC code with 3,600 ones. The (300×1200) generator matrix of the equivalent systematic LDPC codes, however, has 135,000 ones!
Although a code word can be calculated by means of the relation \( c = u \cdot G \), use of this approach is not preferred in practice. In contrast to the check matrix, with LDPC codes the generator matrix is not sparsely populated in general. The coding according to this method is complex, since currently LDPC codes only show their strengths for large block lengths. Therefore, in most cases special LDPC codes (e.g. cyclic codes with linear coding complexity) are applied for implementation.

Everything that is not easy in case of coding is, less than ever, easy for decoding. A maximum likelihood decoding cannot be realised since the dependency of the complexity on the block length \( n \) is exponential. A set of sub-optimal methods exists, with the iterative methods providing the best performance. Due to their complexity, however, boundaries are set to their implementation.

It can be summarised that the LDPC codes are the largest competitor of the turbo codes. Due to their complexity, however, they also show problems in terms of the time delay.
... there is nothing else but to say:

Good luck for the examination!