Wireless Communications 2

Lecture course: 389.156 (VO)

SS 2020

Univ. Prof. Dr.-Ing. habil. Norbert Görtz
Institute of Telecommunications
Vienna University of Technology
norbert.goertz@nt.tuwien.ac.at
Contents: Wireless Communications 2

Part I: Joint Source-Channel Coding (JSCC)

- Motivation
- The General Problem: Transmission of Continuous-Valued Sources over Noisy Channels
- Theoretical limits
- Special Cases:
  - Gaussian Source over a Gaussian Channel
  - Channel-Optimised Vector Quantisation
- Practical Approaches / Overview
  - System Model for Multimedia Transmission
  - Separation of Source and Channel Coding: Theory and Practical Limitations (delay and complexity)
  - Approaches to Joint Source-Channel Decoding
  - Approaches to Joint Source-Channel Encoding
- Soft Information Processing
- Joint Source-Channel Decoding
- Iterative Source-Channel Decoding
- (Multiple Descriptions)

Part II: Cross-Layer Design

- Open Systems Interconnection Reference Model: basic concept and limitations for wireless systems
- Cross-Layer Design: principles
- Wireless multiuser systems: Information Theory
- Multiuser diversity gain
- Quality-of-Service vs. Throughput
- Resource allocation and user scheduling
- Practical aspects, delay in particular
- Scheduling concepts for wireless applications

Part III: Noisy Compressed Sensing (if time permits)

- JSCC from a different angle

Course Textbook:
(for joint source-channel coding; not required for course)
Summary:

The course introduces the basic principles of joint source-channel coding and cross-layer design. It is demonstrated by examples from speech transmission in mobile radio why the classical separation theorem motivated by information theory does not work satisfactory in practical applications with delay constraints; major performance gains can be achieved by joint source-channel coding. Similar arguments, albeit in a broader context, apply to cross-layer design of wireless multiuser systems, in which the classical layered design approach (OSI model) is demonstrated to fail to achieve good performance; cross-layer concepts are shown to be crucial for resource efficiency.

Technicalities:

- The course is taught in English.
- Exams are oral; dates mutually agreed
Course held as “flipped classroom” in SS2020

Lecture notes available for download, see link in TISS:
https://tiss.tuwien.ac.at/course/courseDetails.xhtml?courseNr=389156

Students are supposed to read the part of the lecture notes indicated in the table by the given dates

On the given dates, a **web-meeting will be held at 14:00**

- Link (permanent meeting room):
  https://global.gotomeeting.com/join/193274701
  Technical requirements: web-browser (notebook) or tablet or any modern smartphone (free app available)
  (Chrome-browser on Linux, but NOT Chromium)
  - Technical support: https://support.goto.com/meeting
  - The part of the lecture notes that is up for discussion will be briefly introduced by the lecturer
  - Questions by students will be answered (also off-line if required)
  - Students can email questions to lecturer before the web-meeting: norbert.goertz@tuwien.ac.at

Exams: oral web-meeting (same platform), dates by mutual agreement
Transmission is subject to noise and interference; applications have delay constraints.

Optimisation of network layers cannot be carried out independently.

Joint Source-Channel Coding and Cross-Layer Design are key concepts for efficient use of system resources in wireless multiuser systems.
The General Problem: Transmission of Continuous-Valued Sources over Noisy Channels

Input signal vector $X$

- $N$-dimensional vector with continuous-valued components,
  - e.g., block of $N$ speech samples (or codec parameters)
- block-length $N$ is limited by delay constraints
- pdf $p(x)$ (independent successive blocks assumed for simplicity)

Channel

- used $K$ times to transmit the source vector → inputs and outputs may be binary, real, or complex
- conditional pdf $p(\tilde{z} \mid z)$ (memoryless channel assumed for simplicity)
- input power may be limited, e.g., $E\{\|Z\|^2\}/K \leq P$ (average power per $K$ channel uses to transmit a whole vector $X$) or $E\{|Z_l|^2\} \leq P \quad \forall l$ (average power per channel use)

Goal:

find optimal deterministic encoder $\varepsilon$ /decoder $\varrho$: $\min_{\varrho, \varepsilon \text{: constraint}} E\{d(X, \tilde{X})\}$, $d(x, \tilde{x}) = \frac{1}{N} \sum_{l=0}^{N-1} (x_l - \tilde{x}_l)^2$
Illustration of the Design Problem for $N = 3$ and $K = 2$

**ENCODER**

$z = \varepsilon(x)$

**DECODER**

$\tilde{x} = \varrho(\tilde{z})$

Source signal vector

$N = 3$ real components

Channel input vector

$K = 2$ real components

Channel output vector

$K = 2$ real components

Receiver output vector

$N = 3$ real components

**Goal:**

find optimal deterministic encoder $\varepsilon$ / decoder $\varrho$: $\min_{\varrho, \varepsilon: \text{constraint}} \mathbb{E}\{d(X, \tilde{X})\}, \quad d(x, \tilde{x}) = \frac{1}{N} \sum_{l=0}^{N-1} (x_l - \tilde{x}_l)^2$
System Distortion

\[
\begin{array}{ccc}
\begin{array}{c}
X \\
N
\end{array} & \text{Encoder} & \begin{array}{c}
Z \\
K
\end{array} & \text{Channel} & \begin{array}{c}
\tilde{Z} \\
K
\end{array} & \text{Decoder} & \begin{array}{c}
\tilde{X} \\
N
\end{array}
\end{array}
\]

\[
D(\varepsilon, \varrho) \doteq \mathbb{E}\{d(X, \tilde{X})\} = \int \int d(x, \tilde{x} = \varrho(\tilde{z})) \cdot p(\tilde{z}, x) \, dx \, d\tilde{z}
\]  

(1)

\(X\) and \(\tilde{Z}\) denote the possible sets of “values” (vectors) of the random variables \(X\) and \(\tilde{Z}\) resp.

\(D()\) can be rewritten as

\[
D(\varepsilon, \varrho) = \int \int \int \int D_D(\tilde{z}, \varepsilon, \varrho) \cdot p(\tilde{z}) \, d\tilde{z}
\]  

(2)

with

\[
D_D(\tilde{z}, \varepsilon, \varrho) \doteq \int \int d(x, \varrho(\tilde{z})) \cdot p(x | \tilde{z}) \, dx
\]  

(3)

→ conditional expected distortion, given a particular channel output \(\tilde{z}\)

→ minimization of \(D_D()\) for each particular \(\tilde{z}\) also minimizes \(D()\)
Optimal Decoder $\varrho^\circ$ for a given Encoder $\varepsilon$

Minimization of $D_D(\cdot)$ for squared error distortion

$\rightarrow$ optimization problem:

$$\varrho^\circ(\tilde{z}, \varepsilon) = \arg\min_{\varrho} D_D(\tilde{z}, \varepsilon, \varrho)$$

(4)

For simplicity of notation, the following derivation of the optimal decoder is restricted to the scalar case, i.e., $N = 1$:

$$D_D(\tilde{z}, \varepsilon, \varrho) = \int_{\mathcal{X}} (x - \tilde{x})^2 \cdot p(x | \tilde{z}) \, dx$$

(5)

- We take the derivative

$$\frac{dD_D(\tilde{z}, \varepsilon, \varrho)}{d\tilde{x}} = \frac{d}{d\tilde{x}} \int_{\mathcal{X}} (x - \tilde{x})^2 \cdot p(x | \tilde{z}) \, dx = -\int_{\mathcal{X}} 2 \cdot (x - \tilde{x}) \cdot p(x | \tilde{z}) \, dx .$$

(6)

- Set to zero, we obtain

$$\int_{\mathcal{X}} x \cdot p(x | \tilde{z}) \, dx = \tilde{x} \int_{\mathcal{X}} p(x | \tilde{z}) \, dx = \tilde{x}$$

(7)

- Hence, the optimal receiver equals

$$\tilde{x} = \varrho^\circ(\tilde{z}, \varepsilon) = \int_{\mathcal{X}} x \cdot p(x | \tilde{z}) \, dx = \mathbb{E}\{X | \tilde{z}\} ,$$

(8)

i.e., it is the conditional expectation of the source, given the channel output. This solution can be extended to sources and channels with memory by introducing the past channel-outputs into the “conditioning.”
... Optimal Decoder $\varrho^\circ$ for a given Encoder $\varepsilon$

- The solution $\varrho^\circ(\tilde{z}, \varepsilon)$ for the optimal decoder is quite general; it is the optimal decoder for any deterministic choice of the encoder $\varepsilon$, as long as the mean squared error is used as a quality criterion.
- It should be noticed, however, that an optimal decoder does not guarantee good system performance, as the encoder might be a bad choice.

- The solution for the optimal decoder can be rewritten (vector notation re-introduced):

$$\varrho^\circ(\tilde{z}, \varepsilon) = \mathbb{E}\{X|\tilde{z}\} = \int_{\mathcal{X}} x \cdot p(x | \tilde{z}) \, dx$$

$$\text{(9)}$$

- The a-posteriori probability (density function) (APP) can be reformulated by use of the Bayes rule:

$$\varrho^\circ(\tilde{z}, \varepsilon) = A \int_{\mathcal{X}} x \cdot \underbrace{p(\tilde{z} | z = \varepsilon(x))}_{\text{conditional channel pdf}} \cdot \underbrace{p(x)}_{\text{source pdf}} \, dx$$

$$\text{(10)}$$

with

$$\frac{1}{A} = p(\tilde{z}) = \int_{\mathcal{X}} p(\tilde{z} | z = \varepsilon(x)) \cdot p(x) \, dx$$

$$\text{(11)}$$

- The new formulation of the decoder contains only quantities we know or which we at least can approximate or measure.
- Major problem in many cases: implementation complexity.
Optimal Encoder

Minimization of the system distortion by selection of the encoder:

\[ \varepsilon^\circ = \arg \min_{\varepsilon: \text{constraints}} D(\varepsilon, \varrho^\circ(\varepsilon)) \]  \hspace{1cm} (12)

with

\[ D(\varepsilon, \varrho^\circ(\varepsilon)) = \int \int_{\tilde{z} \times X} d(\tilde{x}, \tilde{\tilde{x}} = \varrho^\circ(\tilde{z}, \varepsilon)) \cdot p(\tilde{z}, x) \, d\tilde{x} \, d\tilde{z} \]  \hspace{1cm} (13)

→ use of the optimal decoder \( \varrho^\circ(\varepsilon, \tilde{z}) \)

- This general optimization task is usually intractable (analytically, but also numerically).
- Only in some very special cases solutions can be found.
Theoretical Limits for Transmission of Continuous-Amplitude Sources over Noisy Channels

• In principle: distortion-rate function $D(R)$ is evaluated at the channel capacity $C$
  → This best possible performance is often referred to as “OPTA” (optimal performance theoretically attainable)

• source-rate $R$ is given in bits/sample, but channel-capacity $C$ is given in bits/channel-use
  → system model: $N$ source samples are transmitted by $K$ channel uses
  → DRF is evaluated at $R = \frac{K}{N}C$

• example uncorrelated Gaussian source:

  \[ D(R) = 2^{-2R\sigma_x^2} \text{ with } R \text{ in bits/sample} \]  
  \[ D = 2^{-2\frac{K}{N}C\sigma_x^2} \Rightarrow \frac{SNR}{dB} = 10 \log_{10}(\frac{\sigma_x^2}{D}) = \frac{K}{N} \cdot \frac{20 \log_{10}(2)}{6.02} \cdot C \]  

  with channel capacity $C = \frac{1}{2} \log_2 \left( 1 + \frac{2E_s}{N_0} \right)$ we obtain

  \[ \frac{SNR}{dB} = \frac{K}{N} \cdot 10 \log_{10} \left( 1 + \frac{2E_s}{N_0} \right) \]  

  $K/N$: number of channel uses per source sample
  $E_s$: energy per transmitted bit; $N_0/2$ channel noise power spectral density
... examples:

- uncorrelated Gaussian source over a binary symmetric channel with error probability \( p \):

  \[
  \frac{SNR}{dB} = \frac{K}{N} \cdot \frac{20 \log_{10}(2)}{6.02} \cdot \left(1 - H_2(p)\right)
  \]
  with

  \[
  H_2(p) = -p \log_2(p) - (1 - p) \log_2(1 - p)
  \]  

  (17)

\( K/N \): number of channel uses per source sample

\( p \): channel bit error probability, “hard decided” binary input AWGN channel: 

\[
p = \frac{1}{2} \text{erfc} \left( \sqrt{E_s/N_0} \right)
\]

- SNR performance limits for a Gaussian source which is transmitted over an AWGN and a binary symmetric channel

  \( K/N \): channel uses per source sample

- Information theoretic results are “non-constructive,” i.e., theory does not say how to build practical systems that achieve the limits.

- optimal system known only for \( K/N = 1 \) and AWGN channel

- SNR-saturation for BSC but \textit{not} for AWGN
Special Case: Gaussian Source and Gaussian Channel

Assumptions:

- Independent discrete-time Gaussian source signal (power $\sigma_x^2$) to be transmitted over a Gaussian channel (power $\sigma_w^2$) with independent noise samples.
- Channel can be used exactly once per source sample, i.e., $N = K$.
- Average channel input power limited to $\frac{1}{K} \mathbb{E}_X \{ \| Z \|^2 \} = P$.

The perfect solution (to be proved):

\[
\mathcal{N}(0, \sigma^2_x) \xrightarrow{\sim} \mathcal{N}(0, \sigma^2_w)
\]

Note:

- NO coding
- NOTHING is quantised
- Blocksize is $N = K = 1$!
... Special Case: Gaussian Source and Gaussian Channel

- Pick encoder to be (how can I know.... ?)
  \[ \varepsilon(x) = \frac{\sqrt{P}}{\sigma_x} \cdot x \]  \hspace{1cm} (18)

- Optimal decoder mapping for a given encoder can be found by evaluating (9):
  \[ \tilde{x} = E_{X|\tilde{z}}\{X|\tilde{z}\} = \int_{-\infty}^{\infty} x \ p(x | \tilde{z}) \ dx = \frac{1}{p(\tilde{z})} \int_{-\infty}^{\infty} x \ p(\tilde{z} | x) \ p(x) \ dx . \]  \hspace{1cm} (19)

- Since the encoder mapping \( z = \alpha \cdot x \), with \( \alpha = \sqrt{P/\sigma_x} \), is deterministic, we can equivalently write
  \[ \tilde{x} = \frac{1}{p(\tilde{z})} \int_{-\infty}^{\infty} x \ p(\tilde{z} | z = \alpha \cdot x) \ p(x) \ dx . \]  \hspace{1cm} (20)

- Input signal and channel noise are independent Gaussian random variables, i.e., the pdfs are \( p(x) = \mathcal{N}(0, \sigma_x^2) \) and \( p(\tilde{z} | z) = p(w) = \mathcal{N}(0, \sigma_w^2) \). Since the channel output \( \tilde{z} \) is the sum of the two independent Gaussian random variables \( Z \) and \( W \), we have \( p(\tilde{z}) = \mathcal{N}(0, P + \sigma_w^2) \).

- By insertion of the probability distributions into (20) we obtain for the decoder mapping (after same computations)
  \[ \tilde{x}(\tilde{z}) = \varrho^\oplus(\tilde{z}, \varepsilon) = \frac{\sigma_x/\sqrt{P}}{1 + \sigma_w^2/P} \cdot \tilde{z} . \]  \hspace{1cm} (21)
• The total system distortion is given by

\[
D = \mathbb{E}_{X,W}\{\|X - \hat{X}\|^2\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left( x - \left( \frac{\sqrt{P}}{\sigma_x} \cdot x + w \right) \sqrt{P} \right)^2 p(x)p(w) dx dw ,
\]

which results in

\[
D = \frac{\sigma_x^2}{1 + P/\sigma_w^2} .
\]

In terms of signal-to-noise ratio we, therefore, obtain

\[
\frac{SNR}{dB} = 10 \log_{10} \frac{\sigma_x^2}{D} = 10 \log_{10} \left( 1 + \frac{P}{\sigma_w^2} \right) .
\]

All results stay the same, if \( N = K > 1 \) is selected, because the independence of the input signal and the channel noise decouples the vector components in the evaluation of (9), so each vector component is transmitted separately. Hence, the decoder mapping (20) is optimal for any choice of \( N \), even for \( N \to \infty \).

The optimality of the decoder mapping, however, does not guarantee that the encoder is optimal.

Therefore, compare the performance of the system with the information theoretical bounds:

\[
\frac{SNR}{dB} = \frac{K}{N} \cdot 10 \log_{10} \left( 1 + \frac{2E_s}{N_0} \right)
\]

\( K/N \): number of channel uses per source sample \( \rightarrow \) “1” in our example!

\( E_s \): energy per transmitted bit; \( N_0/2 \) channel noise power spectral density

Relations: \( P = E_s/T_s \) and \( \sigma_w^2 = \frac{N_0}{2T_s} \), with \( T_s \) the symbol transmit period.

The simple system is indeed optimal!
Some comments:

- This is a very special case: “double matching of sources and channels”
- We don’t need any coding of long blocks – simple scalar scaling does the job!
- This special case is misleading: if only one of the conditions is not fulfilled ($N \neq K$ or the source is correlated or the channel has memory or the channel or the source are non-Gaussian) there is no such simple solution any more and we need complicated coding schemes that work on “long” blocks of source samples to achieve the theoretical limit.
- There is another pair of source and channel for which such a simple scheme exists: Bernoulli (binary) source and binary symmetric channel.
- Practically relevant cases usually do not allow for such simple systems; to use such a simple scheme does also not provide a good approximation of the theoretically achievable performance.
- However, in this special case we achieve perfect performance with a simple system by NOT separating into source and channel coding – this is “true” joint source-channel “coding”.
- In a real-world system, we will not give up separation completely (due to complexity of the fully joined approach) but we will try to pass information between the coding and decoding components to benefit from or compensate for non-ideal performance the components will have – details to follow.
Special Case: Channel-Optimized Vector Quantisation / Binary Symmetric Channel

- $N$-dimensional source vector $X$, independent real components $\rightarrow$ no further restrictions
- channel:
  - binary input, binary output; $K$ channel uses are allowed to transmit each source vector
  - the $K$-dimensional input bit vectors $Z$ and output bit vectors $\tilde{Z}$ may also be interpreted as $K$-bit indices
  - index transition probabilities $Pr(\tilde{Z} = j \mid Z = i), i, j \in \{0, 1, ..., 2^K - 1\}$, are known.
  For a binary symmetric channel (for details see next chapter) with bit error probability $p$ we obtain
  
  $$p_{j|i} = Pr(\tilde{Z} = j \mid Z = i) = \frac{d_H(j, i)}{2^K} \cdot (1 - p)^K$$

  where $d_H(j, i)$ is the Hamming distance between the bit-vector representations of transmitted index $i$ and the received index $j$, i.e., $d_H(j, i)$ equals the number of bit errors inserted by the channel.

Encoder:
- $2^K$ possible channel inputs
  - source vector must be quantised
  - partitioning of the source signal space $X$ into regions $X^{(i)}, i = 0, 1, ..., 2^K - 1$
  - encoding: $z = \varepsilon(x) = i$ if $x \in X^{(i)}$ (the encoder is completely defined by that!)
  - the binary representation of the number $i$ is transmitted over the channel
    $\rightarrow$ at the channel output some bits may be flipped due to bit errors
Source signal space $\mathcal{X}$

$N = 2$ real components

Encoder Mapping:

$z = \varepsilon(x) = i \quad \text{if} \quad x \in \mathcal{X}^{(i)}$

$i = 0, 1, ..., 7$

Partitioning

Channel input vector $K = 3$ bits

Centroids

For given regions $\mathcal{X}^{(i)}$ and index mapping we can find the optimal decoder by evaluation of

$\rho(i, \varepsilon) = \mathbb{E}\{X | \tilde{Z} = i\}$

follows next

The encoder, i.e., the partition regions $\mathcal{X}^{(i)}$ and the mapping, are later optimized by COVQ codebook training
Optimal Decoder for a Given Encoder:

- For the moment, assume that the encoder mapping $\varepsilon$ (the partition regions $\mathcal{X}^{(i)}$ and the index mapping) are known.
- At the channel output $\tilde{Z} = j$, $j = 0, 1, \ldots, 2^K - 1$, is received; possibly some bits are flipped.
- We first evaluate the general result $q^{\oplus}(\tilde{z}, \varepsilon) = E\{X|\tilde{z}\}$ to find the optimal decoder for squared error distortion measure. Note that the channel output $\tilde{z}$ we observe might be our binary index $j$, but it could also be a vector of real matched-filter outputs we receive for the transmitted $K$-bit vector representation of our quantiser index $i$.
  - We first write out the definition of the optimal decoder:
    \[
    \tilde{x}^{\oplus}(\tilde{z}) = q^{\oplus}(\tilde{z}, \varepsilon) = E\{X|\tilde{z}\} = \int_{\mathcal{X}} x \cdot p(x | \tilde{z}) \, dx = \frac{1}{p(\tilde{z})} \int_{\mathcal{X}} x \cdot \left( p(\tilde{z} | x) \cdot \underbrace{p(x)}_{\text{“channel term”}} \right) \, dx .
    \] (27)
  - The integration over the whole region $\mathcal{X}$ can be split into a sum over integrals over the partition regions $\mathcal{X}^{(i)}$.
  - Moreover, as the encoder is deterministic and the source is memoryless, $Z = i$ can be equivalently used in the channel term instead of $x$, if $x \in \mathcal{X}^{(i)}$:
    \[
    \tilde{x}^{\oplus}(\tilde{z}) = \frac{1}{p(\tilde{z})} \sum_{i=0}^{2^K-1} \int_{\mathcal{X}^{(i)}} x \cdot p(\tilde{z} | i) \cdot p(x) \, dx .
    \] (28)
  - This equals
    \[
    \tilde{x}^{\oplus}(\tilde{z}) = \frac{1}{\sum_{i'=0}^{2^K-1} p(\tilde{z}, i')} \sum_{i=0}^{2^K-1} p(\tilde{z} | i) \int_{\mathcal{X}^{(i)}} x \cdot p(x) \, dx
    \] (29)
  - where the denominator was expanded by a summation over all possible channel inputs.
Finally, we obtain for the optimal decoder:

\[
\tilde{x}^\circ(\tilde{z}) = \varrho^\circ(\tilde{z}, \varepsilon) = \frac{\sum_{i=0}^{2^{K-1}} p(\tilde{z} | i) \cdot p_i \cdot c_i}{\sum_{i' = 0}^{2^{K-1}} p(\tilde{z} | i') \cdot p_{i'}}
\]  

with

\[p_i \doteq \int_{\mathcal{X}^{(i)}} p(x) \, dx \quad \rightarrow \quad \text{probability, that a source vector is within } \mathcal{X}^{(i)} \quad (31)\]

and

\[c_i \doteq \frac{1}{p_i} \int_{\mathcal{X}^{(i)}} x \cdot p(x) \, dx \quad \rightarrow \quad \text{centroid of the partition region } \mathcal{X}^{(i)} \quad (32)\]

\[\rightarrow \quad \text{The quantities } c_i \quad \text{are known as the "centroids" of the partition regions } \mathcal{X}^{(i)}; \quad \text{these regions have not been discussed so far: for the moment, they are simply assumed to be given, which also includes their numbering (i.e., the choice of the bit-vectors for the binary transmission of the region numbers).} \]

\[\rightarrow \quad c_i \quad \text{and } p_i \quad \text{can be computed and stored in advance, because both depend only on the pdf of the source vectors and the partitioning of the source signal space; both are fixed when the transmission system is used.} \]

\[\rightarrow \quad \text{Hence, the partitioning of the source space and the definition of the quality measure completely determine the encoder and the decoder (30); the latter is optimal for any choice of the partition regions (encoder mapping).} \]
As the channel output in our case is also binary/discrete, only a limited number of $K$-bit channel outcomes $\tilde{z} = j$, $j = 0, 1, \ldots, 2^K - 1$, exists.

Thus, the optimal decoder will also generate only $2^K$ outputs; the latter are called COVQ code-vectors.

The COVQ code-vectors $y_{\tilde{z}}$, $j = 0, 1, \ldots, 2^K - 1$, can be pre-computed by (30), i.e.,

$$y_{\tilde{z}} \doteq \tilde{x}^\oplus(j) = \tilde{e}^\oplus(j, \varepsilon) = \frac{\sum_{i=0}^{2^K-1} \Pr(\tilde{Z} = j \mid Z = i) \cdot p_i \cdot c_i}{\sum_{i' = 0}^{2^K-1} \Pr(\tilde{Z} = j \mid Z = i') \cdot p_{i'}}$$

and they can be stored in a table—the COVQ codebook.

Thus, if some index $\tilde{Z} = j$ appears at the channel output it can be decoded by a simple table-lookup which will give $y_{\tilde{z}}$.

We still have the problem, however, that we need the partition regions $X^{(i)}$ to compute $c_i$ and $p_i$ by (32) and (31), respectively:

- as these regions define the encoder, we want to optimise them
- we will have the same problem as in conventional VQ: even if we had mathematical descriptions for the (optimised) regions, it would be very hard to use them for encoding; i.e., the encoding operation $\tilde{z} = \varepsilon(x) = i$ if $x \in X^{(i)}$ is extremely hard to implement in practice, especially for large vector dimensions.

→ to solve both problems, we will first define an equivalent alternative encoder
Alternative Implementation of the Encoder:

- Conventional encoding operation: $z = \varepsilon(x) = i \text{ if } x \in \mathcal{X}^{(i)}$
- Descriptions of the partition regions $\mathcal{X}^{(i)}$ are hard to handle, especially for high-dimensional source vectors.

We use an alternative equivalent realisation, that does not require the explicit knowledge of the partition regions $\mathcal{X}^{(i)}$:

- total expected system distortion ($d(\cdot, \cdot)$ denotes the MSE distortion measure):

$$D(\varepsilon, \varrho) = \int_{\mathcal{X}} \sum_{j=0}^{2K-1} d(x, \varrho(j, \varepsilon)) p(\tilde{Z} = j, x) d\tilde{Z}$$

$$= \int_{\mathcal{X}} \sum_{j=0}^{2K-1} d(x, \varrho(j, \varepsilon)) \cdot \Pr(\tilde{Z} = j \mid Z = \varepsilon(x)) \cdot p(x) d\tilde{Z}, \quad (34)$$

$$\equiv D_E(x, \varepsilon, \varrho) \quad (35)$$

- alternative encoding by search of the quantiser index $i \in \{0, 1, \ldots, 2^K - 1\}$ that minimizes the distortion $D_E()$:

$$z = \arg\min_i D_E(x, \varepsilon, \varrho, i) \quad (36)$$

with

$$D_E(x, \varepsilon, \varrho, i) = \sum_{j=0}^{2K-1} d(x, \varrho(j, \varepsilon)) \cdot \Pr(\tilde{Z} = j \mid Z = i) \quad (37)$$

As yet the partition regions $\mathcal{X}^{(i)}$ are not optimized and still implicitly required to compute $y_j$. Solution: →
Iterative Optimization of the Encoder: COVQ Codebook Training

- Variation of the LBG algorithm for conventional VQ codebook training:

  0. Assume that a large set \( T = \{x_0, x_1, x_2, \ldots \} \) of training source vectors \( x_n \) is given. Furthermore assume that an initial guess for the COVQ codebook \( y_j, j = 0, 1, \ldots, 2^K - 1 \) is given. One can, e.g., use a VQ codebook (optimized for bit error probability \( p = 0 \)).

  1. Encode all training vectors by use of (37) and create the sets \( S_X^{(i)} \), which contain all those training vectors that are encoded to the index \( i, i = 0, 1, \ldots, 2^K - 1 \) (\( K \) is the number of channel input bits).

  2. Approximately compute for \( i = 0, 1, \ldots, 2^K - 1 \) the probabilities and the centroids

\[
\begin{align*}
p_i &\approx \frac{|S_X^{(i)}|}{|T|}, \\
c_i &\approx \frac{1}{|S_X^{(i)}|} \sum_{x \in S_X^{(i)}} x
\end{align*}
\]

(38)

then, pre-compute

\[
y_j = \frac{\sum_{i=0}^{2^K-1} \text{channel transition probab.} \Pr(\tilde{Z} = j \mid Z = i) \cdot p_i \cdot c_i}{\sum_{i' = 0}^{2^K-1} \Pr(\tilde{Z} = j \mid Z = i') \cdot p_{i'}}
\]

Return to Step 1, until the total distortion does not decrease more than a pre-specified threshold (see conventional LBG algorithm).

→ partition regions \( X^{(i)} \) are not explicitly required for training and encoding
Low-Complexity Implementation of the Encoder for Squared Error Distance Measure

![Diagram of the encoder system](image)

- Encoding:
  \[
  d_{\text{covq}}(x, y_i) = D_E(x, \varepsilon, q^*, i) = \sum_{j=0}^{2^K-1} d(x, y_j) \cdot \Pr(\tilde{Z} = j \mid Z = i) \]
  \[
  = p_{j|i} \cdot \sum_{l=0}^{N-1} (x_l - y_{j,l})^2 \]
  \[
  = \frac{1}{N} \sum_{l=0}^{N-1} x_l^2 + \frac{2}{N} \left\{ \sum_{j=0}^{2^K-1} p_{j|i} \cdot \frac{1}{N} \sum_{l=0}^{N-1} y_{j,l}^2 - \sum_{l=0}^{N-1} x_l \sum_{j=0}^{2^K-1} p_{j|i} \cdot y_{j,l} \right\} \]

- Next we expand the encoder for squared error distance measure:

\[
q(y_i) \triangleq \sum_{j=0}^{2^K-1} p_{j|i} \cdot y_{j,l} \]

Wireless Communications 2

Special Case: Channel-Optimised Vector Quantisation
• Thus, similar as in the conventional VQ-case, it is sufficient to minimize the simplified COVQ distance measure
\[
d'_{\text{covq}}(x, y_i) = q(y_i) - \sum_{l=0}^{N-1} x_l \overline{y}_{i,l},
\]
with
\[
q(y_i) = \sum_{j=0}^{2K-1} p_{j|i} \left( \frac{1}{2} \sum_{l=0}^{N-1} y_{j,l}^2 \right)
\]
and
\[
\overline{y}_{i,l} = \sum_{j=0}^{2K-1} p_{j|i} \cdot y_{j,l}
\]

• We obtain “transformed” COVQ code-vectors \( \overline{y}_{i} = \{ \overline{y}_{i,l}, l = 0, ..., N-1 \} \), which can be interpreted as the expectation of the code-vector at the receiver conditioned on the possibly transmitted index \( i \).

• The transformed COVQ code-vectors and their “energies” \( q(y_i) \) can be stored at the encoder in place of the actual COVQ codebook.

• The actual COVQ codebook is only required at the decoder.

• As long as the channel properties don’t change (e.g., const. bit error probability) the memory requirements and the complexity of COVQ encoding by minimization of (41) over \( i \) are not larger than in the conventional VQ-case.

• Empty coding regions may occur at higher BERs, i.e., some indices \( i \) are never transmitted. Still, all indices are possibly received due to bit errors, and hence a pre-computed estimation \( y_{j} \) is required for each channel output index \( j \).
Simulation Results for COVQ

- uncorrelated Gaussian source
- binary symmetric channel
- $K$ channel uses per $N$ source samples, here with $K = N$, i.e., rate $R = 1$ bit per sample
- OPTA: optimal performance theoretically attainable for $N \to \infty$:
  \[
  \frac{SNR}{dB} = 6.02 \cdot (1 - H_2(p)) \tag{44}
  \]
  \[
  H_2(p) = -p \log_2(p) - (1 - p) \log_2(1 - p)
  \]

- COVQ is always optimally matched to the channel (new codebook for each $p$)
  \[
  \rightarrow \text{ large gap to optimal performance, mainly due to limited block-length!}
  \]
  \[
  \rightarrow \text{ COVQ is optimal for limited block-length, if the codebook training has found the global optimum}
  \]
  \[
  \rightarrow \text{ complexity and required memory both grow exponentially with } K, \text{ i.e., } K < 10 \text{ in practice}
  \]
... COVQ Simulation Results

Correlated Gaussian source, 5-bit codebooks with $N = 2$ trained for different bit error rates $p_e'$. 

Wireless Communications 2
Special Case: Channel-Optimised Vector Quantisation
Remarks

- COVQ has been extended to
  - joint optimization of the quantiser code-vectors, the index mapping, and the modulation signal set (Vaishampayan, Farvardin; Han, Kim)
  - soft-output channels/optimized decoders (Skoglund; Alajaji, Phamdo; Liu, Ho, Cupermann)
  - improved codebook training to avoid bad local optima (Miller, Rose)

- COVQ is a very interesting concept, because it is the only non-trivial but still “optimal” case, in which there is no separation between source and channel coding.

- Drawback: COVQ can only be used for small block-length $N$, due to memory and complexity limitations.

- In practical source-channel coding schemes, we may apply COVQ after channel decoding to cope with the residual bit error rate.

- We will need channel coding, as we cannot COVQ-encode several hundreds of source samples due to complexity (same problem as in conventional VQ).

- Hence, practical systems are still all based on the separation principle of information theory (tandem of source and channel coding), but with some major modifications that somewhat re-join source and channel coding.
S: vector with samples from a multimedia source signal; reproduction at the decoder denoted by $\tilde{S}$

Most source coding schemes extract parameters from the source vector; the parameters are quantised and coded, i.e., quantisation is usually NOT applied directly to the source signal

The parameters $x_1, \ldots, x_M$ are quantised (coded by a limited number of bits), channel coded and transmitted

In what follows we will interpret the parameter vectors as “the source signal” → approximation, but concepts can be applied generally: therefore we will only consider the part of the system between the points “A” and “B”

Often we will look for a min. MSE reproduction of the parameters and expect that this is a good choice with respect to the quality of the source signal reproduction

⇒ This “trick” allows to treat jointly systems for speech, audio, image, video transmission.
Separation of Source and Channel Coding

- Classical approach, motivated by Separation Theorem from information theory.
- One may replace the transmitter and the receiver mappings by cascades of mappings with binary interfaces, at which independent and uniformly distributed bits (of information) are exchanged – without any loss in performance compared with the best possible direct mapping (see discussion above).
- Assumptions: no limits on delay and complexity, i.e., \( N, N_B \) and \( K \) may be infinitely large, but they don’t always have to be (see special case “Gaussian source over a Gaussian channel”)
- Basic notion:
  - channel code: achieve error-free bits after channel decoding at a bit rate that equals the channel capacity.
    → The “channel code” includes modulation and detection!
  - source code: reduce the number of bits required to represent the source signal to the amount that can be transmitted error-free over the channel
    → reduction only possible at the price of some distortion if the entropy (rate) of the source is higher than the channel capacity. This is always the case for source with continuous amplitudes!
    → An optimal source code will produce independent and identically distributed (iid) bits \( b \)
... Separation of Source and Channel Coding

Remarks:

• The separation theorem does neither say that separation is the only way to build a system nor does it say that it is the “best” way to build a system (and what would “best” mean...?)

• The theorem just says that one may build a system by separation and that there won’t be any loss in performance compared to the best possible (usually unknown) system.

• For applications with delay constraints, i.e., $N, N_B, K < \infty$, the separation theorem does NOT hold!
  ⇒ This means, in fact, that the separation theorem never holds in practice in a strict sense!
  ⇒ Any kind of “coding” (source, channel) is necessarily non-ideal for limited blocksize (discussion below).

• Still, with some modifications, source coding and channel coding (and modulation) are separate processing steps in all practical systems:
  ⇒ separation allows to construct source and channel encoders/decoders, with moderate computational complexity!
  ⇒ for moderate-to-large block size, complexity of a direct source-channel mapping becomes impractical (see channel-optimised vector quantisation)

• Practical approaches to “joint source-channel coding” will try to partially join source and channel coding by adding extra information exchange to the coding and decoding algorithms.
  ⇒ Below we present a tour through common well-known and more recent concepts.
  ⇒ A crucial idea for advanced schemes is “Soft-In/Soft-Out' (SISO) processing (details to follow)
    (Note: SISO is sometimes also used for Single-Input/Single-Output!!)

But first we discuss the effects of limited block size on channel coding and source coding
... Separation of Source and Channel Coding

How limited block size (i.e., $N, N_B, K < \infty$) affects coding schemes:

1. It is impossible to realize channel codes that achieve a bit-error rate of "zero" after decoding.
2. The source encoder output bits are not independent and uniformly distributed, i.e., the coded bits contain residual redundancies.
3. The distortion at the source decoder output depends on which bit is in error, and this is true in an average sense as well as for a particular source input.

Interesting: while (1) is the reason why joint source-channel decoding is needed, (2) and (3) – which are also due to limited block size – form the basis for measures against the quality decrease that arises from residual bit errors.

We will discuss the items (1)...(3) below.
... Separation of Source and Channel Coding

(1) Why we have non-zero bit error rate after channel decoding for limited block size

Example:

- binary symmetric channel with a bit error probability of $p_e = 1/12 = 0.0833$.
- this means that “on average”(!) 1 in 12 bits is in error

⇒ Channel capacity: $C = 1 - H_2(p_e) = 1 - (-p_e \log_2(p_e) - (1-p_e) \log_2(1-p_e)) = 0.5862$ data-bits/channel-use

- We pick two $(K,N)$-channel codes with different block sizes (systematic encoders, block size $K$, number data bits $N$):
  - $(7,4)$-Hamming code, code rate $R_c = 4/7 = 0.571$, can correct all 1-bit errors
  - $(63,36)$-BCH code, code rate $R_c = 36/63 = 0.571$, can correct all 5-bit errors (and fewer errors)
  - One $(63,36)$-BCH code word contains 9 code words of the $(7,4)$-Hamming code

- As the code rate $R_c < C$, ALL errors can be corrected according to the channel coding theorem of information theory assuming infinite block size ⇒ NOT true in practice due to limited block size

- Due to the random nature of the channel, the number of bit errors that occur in a block of limited size does NOT (always) match the bit error probability:
  ⇒ there is a high risk that we will NOT observe 1 bit error in a block of 12 code bits
  ⇒ within a block of size 7 we can not observe 0.5833…. erroneous bits
  ⇒ for a block size of 63000 we are likely to observe “around” 5250 bits errors, but exactly 5250 will hardly ever happen
  ⇒ in a strict sense, convergence of the observed error probability to its “true” value will occur at infinite block size
  ⇒ for infinite block size we know the number of errors we will have to correct and we can design a code for exactly this case
  ⇒ More general comment: how would we know the “true” value of $p_e$ in practice?
... Separation of Source and Channel Coding

... (1) Why we have non-zero bit error rate after channel decoding for limited block size

Illustration:

- codewords (7,4)–code
  - (7,4)–code: decoding error
  - data bits
  - random bit errors
  - redundancy bits

- positions (but NOT values) of redundancy bits are marked by lines below the main horizontal line

- (7,4)-Hamming code (upper part of the figure): some code words do not contain any bit errors (waste of redundancy bits) while others contain two bit errors although only one can be corrected

- (63,36)-BCH code: one code word covers 9 of the (7,4)-Hamming code words.

- in the example, the total of 5 errors can be corrected.

- Hence, the Hamming code will fail in the 6th block while the BCH-code will correct all errors (in this example).

- Note that the Hamming code can correct 1 out of 7 bits while the BCH code can correct only 5 out of 63 bits. Relative to the block size, the Hamming code is actually the “better” code (as $\frac{1}{7} \approx 0.143 > \frac{5}{63} \approx 0.0793$)
  ⇒ Still the BCH code works better, because of its larger block size!
  ⇒ We prefer to use long blocks in channel coding!
... Separation of Source and Channel Coding

... (1) Why we have non-zero bit error rate after channel decoding for limited block size

A more formal view:

- For a binary symmetric channel with bit error probability $p_e > 0$, the probability $P_n$ of $n$ bit errors at any locations within a $K$-bit codeword is given by

$$P_n = \binom{K}{n} \cdot p_e^n \cdot (1 - p_e)^{K-n} \rightarrow P_n > 0 \text{ for } n = 1, \ldots, K \text{ for } p_e > 0.$$  \hspace{1cm} (45)

\Rightarrow For limited block size $K < \infty$, any number $n = 1, \ldots, K$ of bit errors has non-zero probability (for $p_e > 0$)

- Any channel code can only correct a limited number of errors. If we know the minimum distance $d_{\text{min}}$ between any two codewords, the error correction capability $t$ of the code equals ($\lfloor \cdot \rfloor$ denotes the nearest smaller integer, e.g. $\lfloor 1.8 \rfloor = 1$)

$$t = \left\lfloor \frac{d_{\text{min}} - 1}{2} \right\rfloor.$$  \hspace{1cm} (46)

- As the error probability (45) $P_{t+q} > 0$ for $q = 1, 2, 3, \ldots$ there can, no matter what the code rate is, NOT be reliable transmission in a strict sense (i.e. zero errors) for limited block size!

- Note that for infinite block size $K \to \infty$ the ratio $\frac{n_e}{K}$ of observed bit errors $n_e$ and the total number of bits $K$ will converge against the bit error probability $p_e$. This means we will observe $n_e = K \cdot p_e |_{K \to \infty}$ bit errors in the infinitely long codeword, i.e., $P_{n_e} |_{K \to \infty} = 1$ and $P_n |_{K \to \infty} = 0$ for $n \neq n_e$ (with $K$ such that $K \cdot p_e$ is an integer).

- If $t > n_e$ we can correct the error patterns in every block of infinite size and, hence, achieve reliable transmission.

\Rightarrow If $K < \infty$ this is fundamentally impossible – but we may still achieve an error probability that is “low-enough”
... Separation of Source and Channel Coding

(2) Why the source-encoder output bits contain residual redundancies for limited block size

- We consider vector quantisation: can be seen as THE general source coding scheme.
- VQ with infinite dimension will achieve theoretical limits of rate-distortion theory \( \Rightarrow \) nothing to prove here!
- The figure shows realizations of two successive samples (\( \cdot \)) of an uncorrelated Gaussian source (upper two plots) and a low-pass correlated source (lower two plots)

- left two plots: \( \cdot \) indicates the locations of the reproducer values of a source-optimised scalar quantiser in drawn in TWO dimensions
- right two plots: \( \times \) indicates the locations of the reproducer vectors of a source-optimised two-dimensional VQ
- SNR gain by VQ for correlated sources but also in the uncorrelated case
... Separation of Source and Channel Coding

... (2) Why the source-encoder output bits contain residual redundancies for limited block size

- Upper left plot: four reproduction levels of the scalar quantizer in the corners are almost never used, because almost no realizations of the source signal vectors are located in these areas ⇒ Redundancy!!
- Lower left plot: half of the reproducer “vectors” that the scalar quantiser generates in two dimensions are never used. This is because of the source’s low-pass correlation: a large positive value will hardly ever follow a large negative sample. ⇒ A lot of redundancy in the quantised data!!
- Improvement in quantisation SNR possible by source-optimised VQ (right two plots): now the locations of the vector quantizer code-vectors are better adapted to the two-dimensional pdf of the source signal.
- However, in four dimensions the pdf of the source will be spherical but the two-dimensional VQ will produce “corner points” similar to the scalar quantiser as discussed above: hence, there will again – albeit less – redundancy be left in the output bits of the quantiser.
  ⇒ The source correlations between two successive blocks of the VQ is ignored
  ⇒ This effect only goes away, as we increase the VQ dimension to infinity.
- Note that the “imperfectness” of source coding (i.e. residual redundancy) can be exploited to combat channel noise and, hence, to improve the system performance
  ⇒ major argument for joint sourc-channel coding.
Separation of Source and Channel Coding

(3) Why it matters, which bit is in error:

- Figures show the code-vectors of a two-dimensional three-bit vector quantizer and the associated bit-labels.
- Lines connect each pair of the code-vectors whose bit-vectors differ in exactly one bit position; the leftmost bit position in the leftmost plots, the rightmost bit position in the rightmost plots.

- Length of the lines indicate distortion at the decoder output that occurs when, due to a bit error during transmission, code-vectors are permuted.
... Separation of Source and Channel Coding

... (3) Why it matters, which bit is in error:

- Average length of the lines is a measure for distortion at the decoder output due to single bit errors.
- In the optimised bit mapping we achieve much lower distortion, but clearly the bits have very different “importance”: when the leftmost bit is flipped, much higher distortion results compared to the rightmost bit.
- The average distortion for the bit positions seems more balanced in the upper figure (non-optimised case), but the average of the distortion is much higher than in the optimised case.
- If we look at particular code vectors we see a large difference for each bit position for the non-optimised case (upper plot) and more balanced distortion for the optimised case.
- Overall, we aim to achieve low distortion due to bit errors, so we need to pick good optimised bit mappings.
- If we use optimised bit mappings we generate more balanced distortion for a particular bit position, but the “importance” of the bit positions differs significantly. Hence, it really matters which bit is in error.
  ⇒ We may distinguish between “service classes” for the different bit positions of a quantiser and tune the error protection channel coding according to the error sensitivity of the bit class (more details to be discussed later).
- For very large vector dimension, the difference in importance for the bit positions might go away – this is speculation, however, as no one is has managed to do analysis on the topic and, of course, no one has managed to do a simulation (due to complexity).
- In practical source codecs for speech, audio, images and video bits of different “importance” always exist, and the difference can be very large: some bits may code structural information about the coding method used (what would happen if they are flipped ...), while other bits just describe fine details of the signal. Hence, the effects described above are usually even more emphasised in advanced multimedia source codecs.
Separation of Source and Channel Coding

Remarks & Summary

- We have seen above that the assumptions of the separation theorem (infinite complexity and delay) are not fulfilled in practice and that, therefore, the “separated” coding components will necessarily not be optimal.
- This means we should at least partially give up separation in its “pure” form and look for approaches that widely keep the simplicity and moderate complexity of the separation approach but which improve on the performance by advanced information processing.
- The separation theorem suggests that we separate into source coding and channel coding and that “bits” are passed between encoders and decoders.
  - This is based on the idea, that the bits after channel decoding are correct and, hence, totally reliable.
  - As totally reliable transmission is impossible, we need a concept to pass the “reliability” of bits from the channel, to the channel decoder and further to the source decoder.
    ⇒ As “obvious” as this idea may seem, it was definitely NOT trivial in the early 1990s, and pioneering work (e.g. “Soft is better than hard” by J. Hagenauer) paved the way for many modern receiver concepts we know today.

⇒ We will not only give up the separation principle as such but also its implications on the data interfaces within the system: we will change from binary to real-valued data interfaces and use Soft-Input/Soft-Output (SISO) processing (instead of processing only hard bits) – details to follow!

⇒ Note: this has also implications on hardware design: where once “bits” were enough we now need real numbers ...
Practical Approaches to Joint Source-Channel Decoding: An Overview

- The encoding scheme is assumed to be given and we try to find a “good” joint decoder.
- Important: this does not mean that we necessarily get a good overall system! The encoder may be a bad choice and then an “optimal decoder” won’t help very much.
- Remember: the best possible decoder with MSE distance measure for a given encoder is

\[ \hat{x} = E\{X|\tilde{z}\} \]  

(47)

with \(X\) the transmitted parameter/source random variable and \(\tilde{z}\) the observed channel output (equations are modified below for correlated source signals).
- If MSE is our quality measure, a good joint decoding scheme will aim to closely approach \(\hat{x} = E\{X|\tilde{z}\}\)
  \[ \Rightarrow \text{The major drawback is complexity!!} \]
- Some classical and more recent practical approaches to joint source-channel decoding in order of increasing computational complexity
  - Unequal Error Protection and Error Concealment
  - Source-Controlled Channel Decoding
  - Soft-in/Soft-out (SISO) Channel Decoding and Estimation-Based Source Decoding
  - Joint Source-Channel Decoding
  - Iterative Source-Channel Decoding

\[ \Rightarrow \text{discussed in more detail below} \]
Unequal Error Protection (UEP)

- significance (importance) of source-encoder output bits measured → source significance information (SSI)
- measurement typically by listening tests (speech coding) or SNR change due to a bit error in a particular bit position
- more significant bits are given stronger forward error protection (punctured convolutional codes, different coding classes)

Error Concealment

- most significant bits are additionally encoded by an error detecting channel code (cyclic redundancy check CRC)
- if errors cannot be corrected, they may still be detectable by the CRC decoder → BFI-flag is set in this case
- bad frame handling (BFH) procedures are initiated at the source decoder if the BFI-flag is set
  ⇒ frame repetition, extrapolation, muting → simple way to exploit “signal correlation” to combat channel noise
  ⇒ Good BFH will also exploit human perception! (BFH for data signals does not exist – why?)
- the more sensitive bits are often also the stronger correlated ones at the same time (such as bits coding signal energy)
  ⇒ UEP and error concealment can complement each other: error concealment for most important bits only

UEP and Error Concealment are very efficient and both have very low complexity → very frequently used
Source-Controlled Channel Decoding (Hagenauer, 1995)

- **bit** statistics (probability distributions and correlations in time) are used as a-priori information for the channel decoder (*binary* convolutional code, Viterbi decoder)
- channel state info (CSI) must be available at the decoder to correctly weight the inputs $\tilde{Z}$ from the channel and the a-priori knowledge
- bit error rate after channel decoding is significantly reduced, if strong bit correlations exist

**drawback**: groups of bits (quantizer indexes) can contain strong redundancies, but the individual index bits may not
→ no gain!

<table>
<thead>
<tr>
<th>$I = {i_1, i_0}$</th>
<th>$P(I)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${0, 0}$</td>
<td>$0.1$</td>
</tr>
<tr>
<td>${0, 1}$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>${1, 0}$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>${1, 1}$</td>
<td>$0.1$</td>
</tr>
</tbody>
</table>

Example:
index $I$ contains redundancy, but the individual bits don’t:
$P(i_0 = 0) = 0.5$
$P(i_1 = 0) = 0.5$
Soft-In/Soft-Out (SISO) Channel Decoding and Estimation-Based Source Decoding

- **basic idea**: use of the full index-based statistical information for source decoding
  (Phamdo, Farvardin, 1994; Fingscheidt, Vary, 1996; Skoglund 1997; Sayood, Borkenhagen 1991; Miller, Park 1998)
- **estimation of the output signal**, e.g., by the *minimum mean-square error estimator*

\[
\hat{X}_k = \mathbb{E}\{X_k | \tilde{i}_k, \tilde{i}_{k-1}, \ldots\} \tag{48}
\]

→ \(\tilde{i}_k\) contains the reliabilities of the bit decisions → SISO channel decoder needed!

→ index a-priori information, modeled by Markov chains (with limited order; typically order of “one”) is used

→ realizable decoder \(\tilde{x}_k = \sum I_k y_{I_k} \cdot \Pr(I_k | \tilde{i}_k, \tilde{i}_{k-1}, \ldots)\) is slightly suboptimal when \(I_k\) is modeled as a first-order Markov chain and \(X_k\) can be accurately modeled as a first-order Markov chain (details below)

- channel code improves the virtual channel “seen” by the source decoder
- **drawback**: channel decoding is *not* improved by the source a-priori information
- we may combine this scheme with the previous one (use bit-based statistics to improve channel decoding), but it is hard to control what information is used twice
Joint Source-Channel Decoding

- example: parameter coding by vector quantization; quantizer index $I_k$ with $N_b$ bits at each time $k$
  - quantizer codevectors: $y_\lambda$, $\lambda = 0, 1, \ldots, 2^{N_b} - 1$
  - if $X_k$ is correlated, redundancy remains in the indices $I_k$ after source encoding
- channel coding: codewords $Z_k = \text{CC}(I_k)$ (any code can be used)
- slightly suboptimal but feasible decoder:

$$
\tilde{X}_k = \sum_{\lambda=0}^{2^{N_b} - 1} y_\lambda \cdot P(I_k = \lambda \mid \tilde{z}_k, \tilde{z}_{k-1}, \ldots) \in \mathbb{E}\{X_k \mid \tilde{z}_k, \tilde{z}_{k-1}, \ldots\} 
$$

- The key to joint decoding is the computation of the index a-posteriori probabilities $P(I_k = \lambda \mid \tilde{z}_k, \tilde{z}_{k-1}, \ldots)$!
- Recursive solution using known quantities ⇒
... Joint Source-Channel Decoding: Computation of the Index A-Posteriori Probabilities

$Z_k = \text{CC}(I_k)$

\[ P(I_k = \lambda \mid \tilde{z}_k, \tilde{z}_{k-1}, \ldots) = M_k \cdot p(\tilde{z}_k \mid z_k = \text{CC}(\lambda)). \]

\[ 2^{N_b - 1} \sum_{\mu=0}^{2^{N_b} - 1} P(I_k = \lambda \mid I_{k-1} = \mu) \cdot P(I_{k-1} = \mu \mid \tilde{z}_{k-1}, \tilde{z}_{k-2}, \ldots) \]

- $M_k$ normalizes the left-hand side such that it sums up to 1 over all $\lambda$
- initialization by the unconditional probability distribution of the source-encoder indices $P(I_k = \lambda)$
- can be extended to several indices that are jointly channel-encoded: very high computational complexity!
- note that "channel-code redundancy" and "source-redundancy" appear as "factors" in the computation of the APPs: there is no structural difference between redundancies from the source and the channel code!
- main problem is computational complexity! – more details to follow
Iterative Source-Channel Decoding

- the implicit redundancies in the source encoder indexes are interpreted as a “channel code”
  - transmitter is a serially concatenated coding system
- the concatenated code is iteratively decoded (Görtz, 2000; Hindelang, 2000; Adrat, Vary, 2001)
  - APP-algorithm for (convolutional) channel codes is available (BCJR, 1974)
  - APP-algorithm for the source code can be derived from the estimation-based decoders; extrinsic information for the
    index bits is derived from the implicit index-based redundancies
  - usually, a few iterations are enough
- works best of all realizable schemes, but is also the most complex approach
- approximates the optimal joint decoder (but would not be able to compensate for a bad encoder design)
- details to follow
Approaches to Joint Source-Channel Encoding

- Less structured field than joint decoding!
- Reminder: joint decoding is essentially a trial to approach the “optimal” formula $\tilde{x}_k = \mathbb{E}\{X_k | \tilde{z}_k, \tilde{z}_{k-1}, \ldots\}$ (time index and set of channel outputs for times $k, k-1, \ldots$ introduced to account for correlated sources in a causal system)  
  ⇒ This is for any given encoder, even a bad one!
- Now: try to find optimal encoder – much harder problem with no structured approach.
- Decoder design is applied math (with approximations that require creativity) – encoder design is as such a creative task!
- Channel-Optimised Vector Quantisation (COVQ) merges the design problem of encoder and decoder into a numerical optimisation without any obvious “structure”.
- Some approaches to encoder design (some with constraints)
  - index optimisation
  - multiple descriptions for internet / erasure channels
  - (source) adaptive power allocation and modulation
  - ...
  
  We will discuss some ideas below.
Soft Information Processing

⇒ Instrumental for ANY Advanced Receiver Design

Basic concepts:

- Exploit all knowledge available at the receiver to characterise the likelihood of a “bit decision” to be correct (“reliability”).
- Avoid hard decisions on symbols or bits whenever possible and rather process the reliability information.
- The concept does NOT assume any particular channel model although for the Gaussian channel many of the results take simple forms.
- We start from the transmission of a single bit $I_l$ (random variable) and develop its “reliability” at the receiving end upon observation of a channel output $\tilde{z}_l$ (value).
- We then generalise to channel-coded bit sequences.
Log-Likelihood Ratio – L-Value: Basic Definitions

• Assume we have a random variable $I_k$ which takes values from the set $\{0, 1\}$, i.e., $I_k \in \{0, 1\}$.

• Further assume we denote by $p^0_k = \Pr\{I_k = 0\}$ the probability that the random variable $I_k$ takes the value "0" and by $p^1_k = \Pr\{I_k = 1\}$ the probability that it takes the value "1". Of course, we have $p^0_k = 1 - p^1_k$.

• In the framework of bit transmission in communications we would call $p^0_k$ or $p^1_k$ "a-priori information", because this is what is known "in advance" about the binary random variable $I_k$ before any transmission has at all taken place.

• It will turn out to be convenient to convert this probability information into another form we will call "L-value".

• The definition of the a-priori L-value is as follows:

$$L(I_k) \doteq \log \frac{\Pr\{I_k = 0\}}{\Pr\{I_k = 1\}} = \log \frac{p^0_k}{p^1_k} = \log \frac{p^0_k}{1 - p^0_k} = \log \frac{1 - p^1_k}{p^1_k}$$ (51)

• The relation is invertible, i.e., we can solve (51) for $p^0_k$ and $p^1_k$ and we obtain

$$p^0_k = \frac{1}{1 + e^{-L(I_k)}} \quad \text{and} \quad p^1_k = \frac{e^{-L(I_k)}}{1 + e^{-L(I_k)}}$$ (52)

This can be summarised in compact form as follows:

$$p^b_k \doteq \Pr\{I_k = b\} = \frac{e^{-L(I_k) \cdot b}}{1 + e^{-L(I_k)}} \quad \text{with} \quad b \in \{0, 1\}$$ (53)

• Such an (a-priori) L-value definition is possible for any binary random variable. If conditional probabilities are involved, the L-value will also be "conditional".
Log-Likelihood Ratio – L-Value: Basic Concepts for Transmission of Bits

- Assume we transmit a data bit $i_k \in \{0, 1\}$ and at the channel output we observe the value $\tilde{z}_k$, with $k$ the time or bit index. We do not specify what values $\tilde{z}_k$ may take, but we assume we know a channel model that describes the “probability” of a channel output given a channel input, i.e., we know a function (typically a conditional pdf)

$$p_{\tilde{Z}_k|I_k}(\tilde{z}_k|i_k) \text{ Abbr.} = p_{c}(\tilde{z}_k|i_k)$$  \hspace{1cm} (54)

- For the moment we also assume that the channel input is binary (e.g., binary phase shift keying for modulation).

- We further assume that the channel is memoryless, i.e.,

$$p_{c}(\tilde{z}_k|i_k, i_{k\pm 1}, i_{k\pm 2}, \ldots; \tilde{z}_{k\pm 1}, \tilde{z}_{k\pm 2}, \ldots) = p_{c}(\tilde{z}_k|i_k)$$  \hspace{1cm} (55)

- At the receiver we don’t know the transmitted bit itself – we rather know the (noisy) channel observation $\tilde{z}_k$. Therefore we treat the transmitted bit as a random variable $I_k$ (capital letter!; small letters correspond to a value a random variable will take, such as $\tilde{z}_k$).

- We may have a-priori knowledge about the bit $I_k$, e.g., we may know that it is more likely to take the value “0” than “1”. This would be expressed by a probability distribution $\Pr\{I_k = 0\} = p_{0}^k$ and $\Pr\{I_k = 0\} = p_{1}^k = 1 - p_{0}^k$. We assume that $p_{0}^k$ is known. In channel coding, we often assume $p_{0}^k = 0.5 = p_{1}^k$ for convenience.

- We define the a-posteriori $L$-value of the transmitted bit as follows

$$L(I_k|\tilde{z}_k) \doteq \log \frac{\Pr\{I_k = 0|\tilde{Z}_k = \tilde{z}_k\}}{\Pr\{I_k = 1|\tilde{Z}_k = \tilde{z}_k\}}$$  \hspace{1cm} (56)

with $\tilde{z}_k$ the observed channel output that results from the transmission of the unknown bit $I_k$ (random variable).
Log-Likelihood Ratio – L-Value: Basic Concepts for Transmission of Bits

- Note that $L(I_k|\tilde{z}_k)$ is the L-value of the data bit (random variable) $I_k$ given the channel observation $\tilde{z}_k$, i.e., the L-value is conditional on $\tilde{z}_k$.
- As a-posteriori probabilities (APPs) $\Pr\{I_k = 0|\tilde{Z}_k = \tilde{z}_k\}$ are used in the definition we call $L(I_k|\tilde{z}_k)$ an APP-L-value.
- Numerical properties of the L-value definition:
  - The L-value is a real number that characterises the statistical properties of a random variable!
  - When $\Pr\{I_k = 0|\tilde{Z}_k = \tilde{z}_k\} > \Pr\{I_k = 1|\tilde{Z}_k = \tilde{z}_k\}$, we have $\frac{\Pr\{I_k = 0|\tilde{Z}_k = \tilde{z}_k\}}{\Pr\{I_k = 1|\tilde{Z}_k = \tilde{z}_k\}} > 1$ and, therefore, $L(I_k|\tilde{z}_k)$ will be positive.
  - The opposite is true if $\Pr\{I_k = 0|\tilde{Z}_k = \tilde{z}_k\} < \Pr\{I_k = 1|\tilde{Z}_k = \tilde{z}_k\}$.
  - For $\Pr\{I_k = 0|\tilde{Z}_k = \tilde{z}_k\} = \Pr\{I_k = 1|\tilde{Z}_k = \tilde{z}_k\}$ we obtain $L(I_k|\tilde{z}_k) = 0$

This means that the sign of the L-value determines if a bit should be decided as a “zero” (if $L(I_k|\tilde{z}_k) > 0$) or a “one” (if $L(I_k|\tilde{z}_k) < 0$) and the magnitude $|L(I_k|\tilde{z}_k)|$ determines the reliability of such a sign-decision: a large magnitude corresponds to high reliability, i.e., the likelihood that the decision would be correct is large (see example below).

- Important: we will avoid to really take bit decisions whenever and wherever we can!!! This is the key concept of “Soft Information Processing”.

Wireless Communications 2
Soft Information Processing
Log-Likelihood Ratios – L-Value: Calculation Rules

- **Goal:** express L-values by quantities we know, i.e., channel model and a-priori probabilities.

- **We use Bayes’ rule**

\[
\Pr\{A | B\} = \frac{\Pr\{A, B\}}{\Pr\{B\}} = \frac{\Pr\{B | A\} \Pr\{A\}}{\Pr\{B\}}
\]

with the events \(A\) and \(B\). This rule also applies in mixed form, when one variable is continuous-valued and the other takes discrete values (this means we mix probabilities and probability density functions).

- **We equivalently obtain with** \(A. = \{I_k = 0/1\}\) and \(B. = \{\tilde{Z}_k = \tilde{z}_k\}\):

\[
L(I_k | \tilde{z}_k) = \log \frac{\Pr\{I_k = 0 | \tilde{Z}_k = \tilde{z}_k\}}{\Pr\{I_k = 1 | \tilde{Z}_k = \tilde{z}_k\}} = \log \frac{p_c(\tilde{z}_k | 0) \Pr\{I_k = 0\}}{p_c(\tilde{z}_k | 1) \Pr\{I_k = 1\}} = \log \frac{p_c(\tilde{z} | 0)}{p_c(\tilde{z} | 1)} + \log \frac{\Pr\{I_k = 0\}}{\Pr\{I_k = 1\}}
\]

\[
= L(\tilde{z}_k | I_k) + L(I_k)
\]

- **Note that the channel term**

\[
L(\tilde{z}_k | I_k) = \log \frac{p_c(\tilde{z}_k | 0)}{p_c(\tilde{z}_k | 1)}
\]

involves the observed channel output \(\tilde{z}_k\) (which is a number!) and that for the binary random variable \(I_k\) both possible values are inserted as “conditions” in the numerator and the denominator, respectively. This means, in fact, that the channel term \(L(\tilde{z}_k | I_k)\) only depends on the observed channel output and the channel model \(p_Z_k | I_k\).)

- **Note that the channel term is actually the only proper “likelihood ratio” in our definitions (in the sense of mathematical statistics); that is why we have introduced “a-priori” and “a-posteriori” L-values above.**

- **The channel term is NOT a “sufficient statistic” for the random variable (RV) \(I_k\) as probability information on \(I_k\) can NOT be computed: the a-priori information is missing!”
Example: Gaussian Channel with BPSK Modulation

- In the special Gaussian case with BPSK modulation, the channel model is as follows:

\[
p_{c}(\tilde{z}_{k}|i_{k}) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{1}{2\sigma^{2}}(\tilde{z}_{k} - z_{k})^{2} \right) \bigg|_{z_{k}=1-2i_{k}} \quad \text{with} \quad i_{k} \in \{0, 1\} \quad (60)
\]

- We use (60) in (59) and we obtain for the channel term

\[
L(\tilde{z}_{k}|I_{k}) = \log \frac{p_{c}(\tilde{z}_{k}|0)}{p_{c}(\tilde{z}_{k}|1)} = \log \frac{\exp \left( -\frac{1}{2\sigma^{2}}(\tilde{z}_{k} - 1)^{2} \right)}{\exp \left( -\frac{1}{2\sigma^{2}}(\tilde{z}_{k} + 1)^{2} \right)} = \log \left( \exp \left( -\frac{1}{2\sigma^{2}}(\tilde{z}_{k} - 1)^{2} + \frac{1}{2\sigma^{2}}(\tilde{z}_{k} + 1)^{2} \right) \right)
\]

\[
= \frac{1}{2\sigma^{2}} \left( -\tilde{z}_{k}^{2} + 2\tilde{z}_{k} - 1 + \tilde{z}_{k}^{2} + 2\tilde{z}_{k} + 1 \right) = \frac{1}{2\sigma^{2}} 4\tilde{z}_{k} = \frac{2}{\sigma^{2}} \tilde{z}_{k} \quad (61)
\]
Example: Gaussian Channel with BPSK Modulation

- Hence, for the Gaussian channel with BPSK modulation the channel-term in L-value notation takes the simple form

\[ L(\tilde{z}_k | I_k) = \log \frac{p_c(\tilde{z}_k | 0)}{p_c(\tilde{z}_k | 1)} = \frac{2}{\sigma^2} \tilde{z}_k \tag{62} \]

i.e., we only need to scale the observed channel output with the factor \(\frac{2}{\sigma^2}\) to obtain the L-value.

- We compute the probability \(\varepsilon_k\) of error for a sign decision of bit \(I_k\). We use the “BPSK-modulated” bit \(z_k = 1 - 2 \cdot i_k\), with \(i_k \in \{0, 1\}\), to simplify notation. For simplicity we further assume that the bits are uniformly distributed, i.e., \(\Pr\{z_k = 1\} = \Pr\{z_k = 0\} = 0.5\):

\[
\varepsilon_k \doteq \Pr\left( z_k \neq \text{sign}(\tilde{z}_k) \mid \tilde{z}_k \right) = \frac{\underbrace{p(\tilde{z}_k \mid (z_k \neq \text{sign}(\tilde{z}_k))) \cdot p(z_k \neq \text{sign}(\tilde{z}_k))}{p(\tilde{z}_k)} = \frac{p(\tilde{z}_k \mid z_k \neq \text{sign}(\tilde{z}_k))}{\sum_{z_k \in \{+1, -1\}} p(\tilde{z}_k \mid z_k)} \tag{63}
\]

With the Gaussian pdf

\[ p(\tilde{z}_k | z_k) = \frac{1}{\sqrt{2\pi \sigma}} e^{-\frac{1}{2\sigma^2}(\tilde{z}_k - z_k)^2} \text{ with } z_k \in \{+1, -1\} \tag{64} \]

we obtain

\[
\varepsilon_k = \frac{\exp\left(-\frac{1}{2\sigma^2}(\tilde{z}_k + \text{sign}(\tilde{z}_k))^2\right)}{\exp\left(-\frac{1}{2\sigma^2}(\tilde{z}_k - 1)^2\right) + \exp\left(-\frac{1}{2\sigma^2}(\tilde{z}_k + 1)^2\right)} = \ldots = \frac{1}{1 + \exp\left(\frac{2}{\sigma^2} \tilde{z}_k\right)} = \frac{1}{1 + \exp(|L(\tilde{z}_k | I_k)|)} \tag{65}
\]

- Note that, as the magnitude \(|\tilde{z}_k|\) gets larger, the probability of error for the sign-decision of bit \(I_k\) is gets smaller! This quantifies the reliability-interpretation of the magnitude \(|\tilde{z}_k|\) of the channel observation and of the L-value \(2|\tilde{z}_k|/\sigma^2\).
Example: Gaussian Channel with BPSK Modulation

- The result above is for the error probability of a particular bit decision at time $k$!
- The average probability of error for this example is given by the well-known result (AWGN channel BPSK modulation)

$$\varepsilon = \int_{0}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(\tilde{z}_k - (-1))^2\right) d\tilde{z}_k = \frac{1}{2} \text{erfc}\left(\sqrt{\frac{1}{2\sigma^2}}\right)$$

which follows from an integration over the error-region of the PDF of the Gaussian noise (e.g. range of positive numbers for a “-1” decision, assuming uniformly distributed input bits)

![Error Probability Diagram](image)

(error prob. $\varepsilon$)

($p(\tilde{z}_k | z_k = 1)$ and $p(\tilde{z}_k | z_k = 0)$)

(symmetric for $v_k = 1$ and $v_k = 0$)
L-Values: Calculations with Parity Bits

- We start from two data bits $I_1, I_2 \in \{0, 1\}$ that are channel-encoded by a single parity bit $I_3 \in \{0, 1\}$. The channel coding rule is

$$I_1 \oplus I_2 = I_3 \quad \text{or equivalently} \quad I_1 \oplus I_2 \oplus I_3 = 0$$

(66)

with $\oplus$ denoting “modulo-2” addition (which really is XOR):

<table>
<thead>
<tr>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3 = I_1 \oplus I_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- At the channel output we observe the receive values $\tilde{z}_1, \tilde{z}_2, \tilde{z}_3$ which we organise in the vector $\tilde{z} = \{\tilde{z}_1, \tilde{z}_2, \tilde{z}_3\}$.

- As there is now an underlying code that connects the three bits, not only the receive value $\tilde{z}_k$ ($k \in \{1, 2, 3\}$) (known deterministic value because it is an observation) will contain information about the transmitted bit $I_k$ (random variable) but also the other received values of the codeword.

- Therefore we now want to calculate the APP (a posteriori probability) of the random variable $I_j$ given the WHOLE receive vector $\tilde{z}$ of the channel codeword, not only the channel observation for the transmitted bit at position $j$.

Hence, we aim to calculate the APP

$$\Pr\{I_k = b | \tilde{z}\} \quad \text{with} \quad b \in \{0, 1\}$$
... L-Values: Calculations with Parity Bits

- We use the standard-trick to insert new random variables and compute the marginal probability:

\[
Pr\{I_k = b|\tilde{z}\} = \sum_{\forall i \in C : i_k = b} Pr\{I = i|\tilde{z}\}
\]  

(67)

Note that the summation is carried out over all codewords which have a bit value \(b\) at bit position \(k\). Furthermore note that the summation is taken over valid codewords only, i.e., any combination of bits \(i \subset \{0, 1\}^3\) MUST fulfil the code constraint \(i_1 \oplus i_2 \oplus i_3 = 0\). The short notation for the code \(C\) is, hence,

\[
C = \{i \in \{0, 1\}^3 : i_1 \oplus i_2 \oplus i_3 = 0\}
\]  

(68)

- We use Bayes’ rule in (67):

\[
Pr\{I_k = b|\tilde{z}\} = \sum_{\forall i \in C : i_k = b} \frac{p(\tilde{z}|I = i) Pr\{I = i\}}{p(\tilde{z})}
\]  

(69)

Note that now probability density functions \(p(\tilde{z}|I = i)\) and \(p(\tilde{z})\) are involved, as we, in general, assume a channel model with real outputs.

- We assume that the channel is memoryless, i.e., the bits are transmitted independently. This means we can decompose the term \(p(\tilde{z}|I = i)\) into factors, i.e.,

\[
p(\tilde{z}|I = i) = \prod_{l=1}^{3} p(\tilde{z}_l|I_l = i_l)
\]  

(70)

Note that \(p(\tilde{z}_l|I_l = i_l)\) is our channel model!

- The term \(p(\tilde{z})\) denotes the unconditional channel output. This quantity is actually not known, but it does NOT depend on the bit \(b\) we pick on the left-hand side of the equation. We will see that this term cancels out in the L-value notation below, so we leave it unaltered.
... L-Values: Calculations with Parity Bits

- We rewrite (69):
\[
\Pr\{I_k = b | \tilde{z}\} = \sum_{\forall i \in C: i_k = b} \frac{p(\tilde{z} | I = i) \Pr\{I = i\}}{p(\tilde{z})} = \frac{1}{p(\tilde{z})} \sum_{\forall i \in C: i_k = b} \prod_{l=1}^{3} p(\tilde{z}_l | I_l = i_l) \cdot \prod_{l'=1}^{3} \Pr\{I_{l'} = i_{l'}\} \quad (71)
\]

- The term \( \Pr\{I = i\} \) was also decomposed into factors, assuming that the a-priori information about the code bits is also independent!

- Note that all equations up to this point would have applied for an \( n \)-bit parity check as well, i.e., we could replace “3” by a general “\( n \)” and check \( n - 1 \) data bits by a single parity bit. In what follows we specialise to \( n = 3 \) (as in the equations above), but we generalise later back to an arbitrary number of data bits.

- Now we continue with (71) for the case \( n = 3 \) and consider all possible cases (i.e. codewords) with a bit value of \( b \) at position/bit number \( k \). For this we pick position number 1, i.e., we now ask for the probability \( \Pr\{I_1 = b | \tilde{z}\} \).

- We obtain from the parity-check equation: \( b = I_2 \oplus I_3 \). If we pick \( I_2 = 0 \), then \( I_3 = b \); if we pick \( I_2 = 1 \), then \( I_3 = \bar{b} \) (with \( \bar{0} = 1 \) and \( \bar{1} = 0 \)). We insert those results into (71) and obtain

\[
\Pr\{I_1 = b | \tilde{z}\} = \frac{1}{p(\tilde{z})} \left[ p(\tilde{z}_1 | I_1 = b) \Pr\{I_1 = b\} \cdot \frac{I_2=0 \text{ picked value}}{p(\tilde{z}_2 | I_2 = 0) \Pr\{I_2 = 0\}} \cdot \frac{I_3=b \oplus I_2=b}{p(\tilde{z}_3 | I_3 = b) \Pr\{I_3 = b\}} +
\right.

\[
\left. p(\tilde{z}_1 | I_1 = b) \Pr\{I_1 = b\} \cdot \frac{I_2=1 \text{ picked value}}{p(\tilde{z}_2 | I_2 = 1) \Pr\{I_2 = 1\}} \cdot \frac{I_3=\bar{b} \oplus I_2=\bar{b}}{p(\tilde{z}_3 | I_3 = \bar{b}) \Pr\{I_3 = \bar{b}\}} \right] \quad (72)
\]
... L-Values: Calculations with Parity Bits

- Now we form the L-value

\[
L(I_1|\check{z}) \doteq \log \frac{\Pr\{I_1 = 0|\check{z}\}}{\Pr\{I_1 = 1|\check{z}\}}
\]

\[
= p(\check{z}_1|I_1 = 0) \Pr\{I_1 = 0\} \cdot p(\check{z}_2|I_2 = 0) \Pr\{I_2 = 0\} \cdot p(\check{z}_3|I_3 = 0) \Pr\{I_3 = 0\} + \ldots
\]

\[
\cdot p(\check{z}_1|I_1 = 1) \Pr\{I_1 = 1\} \cdot p(\check{z}_2|I_2 = 0) \Pr\{I_2 = 0\} \cdot p(\check{z}_3|I_3 = 1) \Pr\{I_3 = 1\} + \ldots
\]

\[
\ldots + p(\check{z}_1|I_1 = 0) \Pr\{I_1 = 0\} \cdot p(\check{z}_2|I_2 = 1) \Pr\{I_2 = 1\} \cdot p(\check{z}_3|I_3 = 1) \Pr\{I_3 = 1\}
\]

\[
\ldots + p(\check{z}_1|I_1 = 1) \Pr\{I_1 = 1\} \cdot p(\check{z}_2|I_2 = 1) \Pr\{I_2 = 1\} \cdot p(\check{z}_3|I_3 = 0) \Pr\{I_3 = 0\}
\]

\[
= \log \left( \frac{p(\check{z}_1|I_1 = 0) \Pr\{I_1 = 0\}}{p(\check{z}_1|I_1 = 1) \Pr\{I_1 = 1\}} \times \ldots \right)
\]

\[
\ldots \times \frac{p(\check{z}_2|I_2 = 0) \Pr\{I_2 = 0\} \cdot p(\check{z}_3|I_3 = 0) \Pr\{I_3 = 0\} + p(\check{z}_2|I_2 = 1) \Pr\{I_2 = 1\} \cdot p(\check{z}_3|I_3 = 1) \Pr\{I_3 = 1\}}{p(\check{z}_2|I_2 = 0) \Pr\{I_2 = 0\} \cdot p(\check{z}_3|I_3 = 1) \Pr\{I_3 = 1\} + p(\check{z}_2|I_2 = 1) \Pr\{I_2 = 1\} \cdot p(\check{z}_3|I_3 = 0) \Pr\{I_3 = 0\}}
\]

\[
= \log \frac{p(\check{z}_1|I_1 = 0)}{p(\check{z}_1|I_1 = 1)} + \log \frac{\Pr\{I_1 = 0\}}{\Pr\{I_1 = 1\}} + \log \frac{p(\check{z}_2|I_2 = 0) \Pr\{I_2 = 0\} p(\check{z}_3|I_3 = 0) \Pr\{I_3 = 0\}}{p(\check{z}_2|I_2 = 1) \Pr\{I_2 = 1\} p(\check{z}_3|I_3 = 1) \Pr\{I_3 = 1\}} + 1
\]

(73)

- Now we use the following L-value definitions (as above):

\[
L(\check{z}_k|I_k) \doteq \log \frac{p(\check{z}_k|I_k = 0)}{p(\check{z}_k|I_k = 1)} \quad \text{and} \quad L(I_k) \doteq \log \frac{\Pr\{I_k = 0\}}{\Pr\{I_k = 1\}}
\]

(74)

Moreover, we write \( \frac{\Pr\{I_k = 0\}}{\Pr\{I_k = 1\}} = e^{\cdot L(I_k)} \) and \( \frac{p(\check{z}_k|I_k = 0)}{p(\check{z}_k|I_k = 1)} = e^{\cdot L(\check{z}_k|I_k)} \) which we insert into the last term in (73).
... L-Values: Calculations with Parity Bits

- We obtain for the APP L-value:

\[
L(I_1 | \tilde{z}) = \underbrace{L(\tilde{z}_1 | I_1)}_{\text{channel information}} + \underbrace{L(I_1)}_{\text{a-priori information}} + \log \frac{e^{L(\tilde{z}_2 | I_2) + L(I_2)}}{e^{L(\tilde{z}_2 | I_2)} + e^{L(\tilde{z}_3 | I_3) + L(I_3)}} + 1
\]  

(75)

and similarly for the other bits:

\[
L(I_2 | \tilde{z}) = \underbrace{L(\tilde{z}_2 | I_2) + L(I_2)}_{\text{extrinsic information}} + \log \frac{e^{L(\tilde{z}_1 | I_1) + L(I_1)}}{e^{L(\tilde{z}_1 | I_1)} + e^{L(\tilde{z}_3 | I_3) + L(I_3)}} + 1
\]  

(76)

and

\[
L(I_3 | \tilde{z}) = \underbrace{L(\tilde{z}_3 | I_3) + L(I_3)}_{\text{extrinsic information}} + \log \frac{e^{L(\tilde{z}_2 | I_2) + L(I_2)}}{e^{L(\tilde{z}_2 | I_2)} + e^{L(\tilde{z}_1 | I_1) + L(I_1)}} + 1
\]  

(77)

- L-values that correspond to statistically independent information about a bit simply add up (simple implementation)!!
- **Extrinsic information** is the knowledge about a bit that OTHER bits of a codeword contribute: the concept of extrinsic information is absolutely fundamental to advanced iterative receivers!!!!
- Equations can be extended to \(n\)-bit parity checks, but for this it is convenient to re-formulate the equations (see below).
- Important:
  - There is NO assumption about a particular channel model in the derivation of the equations above.
  - But we have assumed that the channel is memoryless, i.e., the channel noise at some time does not depend on the noise at any other time.
  - We have assumed in the derivation that the a-priori information of the bits are all mutually independent.
... L-Values: Calculations with Parity Bits

- Definition of the new “boxplus” operator “⊞” (for convenience of notation): we take the formula for extrinsic information and define

\[ l_1 ⊞ l_2 \equiv \log \frac{e^{l_1} \cdot e^{l_2} + 1}{e^{l_1} + e^{l_2}} = \log \frac{1 + \tanh(l_1/2) \cdot \tanh(l_2/2)}{1 - \tanh(l_1/2) \cdot \tanh(l_2/2)} = 2 \tanh^{-1} \left( \tanh(l_1/2) \cdot \tanh(l_2/2) \right) \]  

with \( l_1, l_2 \in \mathbb{R} \) (i.e., any two real numbers, which may be L-values).

- Proof of the second equality:
  We use the definition of the \( \tanh \)-function

\[ y = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \]  

which we insert into the middle term of (78):

\[ \log \frac{1 + \tanh(l_1/2) \cdot \tanh(l_2/2)}{1 - \tanh(l_1/2) \cdot \tanh(l_2/2)} = \log \frac{1 + \frac{e^{l_1-1}}{e^{l_1+1}} \cdot \frac{e^{l_2-1}}{e^{l_2+1}}}{1 - \frac{e^{l_1-1}}{e^{l_1+1}} \cdot \frac{e^{l_2-1}}{e^{l_2+1}}} = \log \frac{(e^{l_1} + 1)(e^{l_2} + 1) + (e^{l_1} - 1)(e^{l_2} - 1)}{(e^{l_1} + 1)(e^{l_2} + 1) - (e^{l_1} - 1)(e^{l_2} - 1)} = \frac{2e^{l_1}e^{l_2} + 2}{2e^{l_1} + 2e^{l_2}} \]  

- Moreover, note that the inversion of (79) leads to

\[ 2x = 2 \tanh^{-1}(y) = \log \frac{1 + y}{1 - y} \]

If we set \( y = \tanh(l_1/2) \cdot \tanh(l_2/2) \) we immediately find the rightmost equality in (78).
... L-Values: Calculations with Parity Bits

- With the “⊞”-operator we can write the decoding equations for the parity-check as follows:

\[
L(I_1 | \tilde{z}) = L(\tilde{z}_1 | I_1) + L(I_1) + \log \frac{e^{L(\tilde{z}_2 | I_2) + L(I_2)}}{e^{L(\tilde{z}_2 | I_2)} + L(I_2)} + \frac{e^{L(\tilde{z}_3 | I_3) + L(I_3)}}{e^{L(\tilde{z}_3 | I_3)} + L(I_3)}
\]

\[= (L(\tilde{z}_1 | I_1) + L(I_1)) + (L(\tilde{z}_2 | I_2) + L(I_2)) \Box (L(\tilde{z}_3 | I_3) + L(I_3)) \quad (81)\]

- Generalised and in compact form we write equivalently:

\[
L(I_k | \tilde{z}) = L(\tilde{z}_k | I_k) + L(I_k) + \left( \sum_{\forall j \neq k} L(\tilde{z}_j | I_j) + L(I_j) \right) \quad k = 1, 2, ..., n \quad (82)
\]

(83) is also valid for \( n > 3 \). The proof follows the same lines as the one for \( n = 3 \) but due to the many factors it gets tedious to write it down.

- The “⊞-sum” in (83) is defined as follows:

\[
\sum_{j=1}^{m} l_j \triangleq l_1 \oplus l_2 \oplus ... \oplus l_m \quad (84)
\]

- We still need to discuss calculation rules for the “⊞” operator.
... L-Values: Calculations with Parity Bits

Calculation rules for the "⊞" operator:

- **Definition:**
  \[ a ⊟ b = \log \frac{e^a \cdot e^b + 1}{e^a + e^b} , \quad a, b \in \mathbb{R} \]  \(85\)

- **Property:** \( a ⊟ b = b ⊟ a \) (commutative property; obvious, due to symmetry of the definition)

- **Property:** \( a ⊟ +\infty = a \)
  Proof:
  \[
  \lim_{b \to +\infty} a \bowtie b = \lim_{b \to +\infty} \log \frac{e^a \cdot e^b + 1}{e^a + e^b} = \lim_{b \to +\infty} \log \frac{e^a + e^{-b}}{e^a \cdot e^{-b} + 1} = \log \frac{e^a}{1} = a
  \]  \(86\)

- **Property:** \( a ⊟ -\infty = -a \)
  Proof:
  \[
  \lim_{b \to -\infty} a \bowtie b = \lim_{b \to -\infty} \log \frac{e^a \cdot e^b + 1}{e^a + e^b} = \lim_{b \to -\infty} \log \frac{1}{e^a} = \log e^{-a} = -a
  \]  \(87\)

- **Property:** \( a \bowtie 0 = 0 \)
  Proof:
  \[
  \log \left| \frac{e^a \cdot e^b + 1}{e^a + e^b} \right|_{b=0} = \log \frac{e^a \cdot 1 + 1}{e^a + 1} = \log(1) = 0
  \]  \(88\)
... L-Values: Calculations with Parity Bits

Calculation rules for the “⊞” operator:

- \( a \boxplus b \boxplus c = (a \boxplus b) \boxplus c = a \boxplus (b \boxplus c) \)

  Proof of associativity:

  \[
  (a \boxplus b) \boxplus c = \log \frac{e^{a\boxplus b} \cdot e^c + 1}{e^{a\boxplus b} + e^c} \quad \text{with} \quad a \boxplus b = \log \frac{e^a \cdot e^b + 1}{e^a + e^b} \quad \text{(89)}
  \]

  We insert \( a \boxplus b \) and obtain

  \[
  (a \boxplus b) \boxplus c = \log \frac{e^a \cdot e^b + 1 \cdot e^c + 1}{e^a + e^b + 1 + e^c} = \log \frac{e^a e^b e^c + e^a + e^b + e^c}{e^a e^b + 1 + e^a e^c + e^b e^c} \quad \text{(90)}
  \]

  Similarly, we obtain for \( a \boxplus (b \boxplus c) \):

  \[
  a \boxplus (b \boxplus c) = \log \frac{e^a \cdot e^b \cdot e^c + 1}{e^a + e^b + e^c + 1} = \log \frac{e^a e^b e^c + e^a + e^b + e^c}{e^a e^b + 1 + e^a e^c + e^b e^c} \quad \text{(91)}
  \]

  As (90) and (91) are equal, associativity has been shown. □

  This also means that a “chain” of \( \boxplus \)-operators can be evaluated in any arbitrary order, e.g.,

  \[
  \sum_{j=1}^{m} l_j \boxplus l_1 \boxplus l_2 \boxplus \ldots \boxplus l_m = (...)((l_1 \boxplus l_2) \boxplus l_3) \boxplus l_4) \boxplus l_5) \ldots \boxplus l_m \]

  \[
  = (...)((l_2 \boxplus (l_3 \boxplus l_4)) \boxplus l_5)) \ldots \boxplus l_m \quad \text{(92)}
  \]
... L-Values: Calculations with Parity Bits

- Using the tanh-notation one can also state a more compact version of a chain of \( m \) \( \ominus \)-operations:

\[
\sum_{j=1}^{m} l_j = \log \frac{1 + \prod_{j=1}^{m} \tanh(l_j/2)}{1 - \prod_{j=1}^{m} \tanh(l_j/2)} = 2 \tanh^{-1} \left( \prod_{j=1}^{m} \tanh(l_j/2) \right)
\]  \( (93) \)

Proof:
We start from \( (78) \), where, for two L-values, the \( \ominus \)-operation was found to equal

\[
l_1 \ominus l_2 = \log \frac{e^{l_1} \cdot e^{l_2} + 1}{e^{l_1} + e^{l_2}} = \log \frac{1 + \tanh(l_1/2) \cdot \tanh(l_2/2)}{1 - \tanh(l_1/2) \cdot \tanh(l_2/2)}
\]  \( (94) \)

For brevity of notation we define

\[
t_j = \tanh(l_j/2),
\]  \( (95) \)

so

\[
l_1 \ominus l_2 = \log \frac{1 + t_1 \cdot t_2}{1 - t_1 \cdot t_2} = \log \frac{1 + \prod_{j=1}^{2} t_j}{1 - \prod_{j=1}^{2} t_j}
\]  \( (96) \)

We extend this to three L-values, i.e.,

\[
(l_1 \ominus l_2) \ominus l_3 = \log \frac{e^{l_1} \ominus l_2 \cdot e^{l_3} + 1}{e^{l_1} \ominus l_2 + e^{l_3}}
\]  \( (97) \)
and we insert (96):

$$
(l_1 \oplus l_2) \oplus l_3 = \log \frac{\frac{1+\prod_{j=1}^{2} t_j}{1-\prod_{j=1}^{2} t_j} \cdot e^{l_3} + 1}{\frac{1+\prod_{j=1}^{2} t_j}{1-\prod_{j=1}^{2} t_j} + e^{l_3}}
$$

(98)

Moreover, we write $e^{l_3}$ in terms of the $t_3 = \tanh(l_3/2)$ (we use the same relation as in (79)):

$$
t_3 = \tanh(l_3/2) = \frac{e^{l_3} - 1}{e^{l_3} + 1} \Rightarrow e^{l_3} = \frac{1 + t_3}{1 - t_3}
$$

(99)

We use (99) to replace $e^{l_3}$ in (98) and obtain

$$
(l_1 \oplus l_2) \oplus l_3 = \log \frac{\frac{1+\prod_{j=1}^{2} t_j}{1-\prod_{j=1}^{2} t_j} \cdot \frac{1+t_3}{1-t_3} + 1}{\frac{1+\prod_{j=1}^{2} t_j}{1-\prod_{j=1}^{2} t_j} + \frac{1+t_3}{1-t_3}} = \log \frac{(1 + \prod_{j=1}^{2} t_j)(1 + t_3) + (1 - \prod_{j=1}^{2} t_j)(1 - t_3)}{(1 + \prod_{j=1}^{2} t_j)(1 - t_3) + (1 - \prod_{j=1}^{2} t_j)(1 + t_3)}
$$

(100)

We expand both the numerator and the denominator:

$$
(l_1 \oplus l_2) \oplus l_3 = \log \frac{(1 + \prod_{j=1}^{2} t_j + t_3 + \prod_{j=1}^{3} t_j) + (1 - \prod_{j=1}^{2} t_j - t_3 + \prod_{j=1}^{3} t_j)}{(1 + \prod_{j=1}^{2} t_j - t_3 - \prod_{j=1}^{3} t_j) + (1 - \prod_{j=1}^{2} t_j + t_3 - \prod_{j=1}^{3} t_j)} = \log \frac{1 + \prod_{j=1}^{3} t_j}{1 - \prod_{j=1}^{3} t_j}
$$

(101)

By re-labelling we can extend this equation to any number $m$ of $\oplus$-sum-terms, so we have proved (93) by induction. ■
... L-Values: Calculations with Parity Bits

Relations and Approximations for the □-operator:

- \(|a □ b| \leq \min(|a|, |b|)\)

This means the magnitude of the result of a □-operation is never larger than the smallest magnitude among the operands.

Proof:
We use the tanh-version of the □-operator:

\[
a □ b = 2 \tanh^{-1}(\tanh(a/2) \cdot \tanh(b/2))
\]

Properties of the \(\tanh\)-function:
- for any \(x \in \mathbb{R}\) we have \(-1 \leq \tanh(x) \leq 1\)
- strictly monotonically increasing
- the function values \(\pm 1\) are only achieved for \(x \to \pm \infty\)
- the function is odd
- the function is uniquely invertible

Those properties imply, that the magnitude of the product \(\tanh(a/2) \cdot \tanh(b/2)\) can never be larger than the magnitude of \(\tanh(a/2)\) and \(\tanh(b/2)\) alone. This, in turn, means that the inverse \(2 \tanh^{-1}\) of the product can never have a magnitude larger than \(|a|\) and \(|b|\). Equality applies, if at least one out of \(a\) and \(b\) has an infinitely large magnitude. ■
... L-Values: Calculations with Parity Bits

Approximations for the $\boxplus$-operator:

- For an implementation it is often desired to have very low computational complexity. In such a situation is common practice to use the following approximation:

$$a \boxplus b \approx a \bar{\oplus} b = \text{sign}(a) \cdot \text{sign}(b) \cdot \min(|a|, |b|)$$  \hfill (103)

Motivation for this approximation: We start again from the $\tanh$-version of the $\boxplus$-operator:

$$a \boxplus b = 2 \tanh^{-1}(\tanh(a/2) \cdot \tanh(b/2)) = \text{sign}(a) \cdot \text{sign}(b) \cdot 2 \tanh^{-1}(\tanh(|a|/2) \cdot \tanh(|b|/2))$$  \hfill (104)

with the second equality due to the fact that $\tanh$ and its inverse are both odd functions.

Now let us assume that $|a| \ll |b|$. Then $\tanh(|a|/2)$ will dominate the product $\tanh(|a|/2) \cdot \tanh(|b|/2)$ as the maximum possible value for $\tanh(|b|/2)$ is “1”. In order to simplify the computation, we assume that indeed $\tanh(|b|/2) = 1$, so we obtain

$$a \boxplus b \approx \text{sign}(a) \cdot \text{sign}(b) \cdot 2 \tanh^{-1}(\tanh(|a|/2) \cdot 1) = \text{sign}(a) \cdot \text{sign}(b) \cdot |a| \quad \text{for } |a| \ll |b|$$  \hfill (105)

For $|a| \gg |b|$ we similarly obtain

$$a \boxplus b \approx \text{sign}(a) \cdot \text{sign}(b) \cdot 2 \tanh^{-1}(1 \cdot \tanh(|b|/2)) = \text{sign}(a) \cdot \text{sign}(b) \cdot |b| \quad \text{for } |a| \gg |b|$$  \hfill (106)

In compact form we can therefore write the equation given in (103). □

- This approximation will be accurate, when $|a|$ and $|b|$ differ significantly. As we also know that $|a \boxplus b| \leq \min(|a|, |b|)$ we conclude that the approximation will usually OVERestimate but never underestimate the reliability of $a \boxplus b$. 
... L-Values: Calculations with Parity Bits

Relations and Approximations for the ⊕-operator:

- We derive the relation between \( a \oplus b \) and its approximation \( a ✲ b \). We start again from (104):

\[
a \oplus b = 2 \tanh^{-1}(\tanh(a/2) \cdot \tanh(b/2)) = \text{sign}(a) \cdot \text{sign}(b) \cdot 2 \tanh^{-1}(\tanh(|a|/2) \cdot \tanh(|b|/2))
\] (107)

We insert the approximation \( a \oplus b \approx a ✲ b = \text{sign}(a) \cdot \text{sign}(b) \min(|a|, |b|) \):

\[
a \oplus b = \text{sign}(a) \cdot \text{sign}(b) \cdot \min(|a|, |b|) \cdot \frac{2 \tanh^{-1}(\tanh(|a|/2) \cdot \tanh(|b|/2))}{\min(|a|, |b|)}
\] (108)

As \( x \oplus y = \log \frac{e^x e^y + 1}{e^x + e^y} = 2 \tanh^{-1}(\tanh(x/2) \cdot \tanh(y/2)) \) we write with \( x := |a| \) and \( y := |b| \):

\[
a \oplus b = \text{sign}(a) \text{sign}(b) \cdot \min(|a|, |b|) \cdot \frac{\log \frac{e^{|a| e^{|b|}} + 1}{e^{|a|} + e^{|b|}}}{\min(|a|, |b|)}
\] (109)

Correction factor: Case 1: \(|a| < |b|\)

\[
\log \frac{e^{|a| e^{|b|}} + 1}{e^{|a|} + e^{|b|}} = \log \frac{e^{|a| e^{|b|}} + e^{-|a|}}{e^{|a|} + e^{|b|}} = \log \frac{e^{|b| + e^{-|a|}}}{e^{|a|} + e^{|b|}} = 1 + \frac{1}{|a|} \log \frac{e^{|b|} + e^{-|a|}}{1 + e^{|a| + |b|}} = 1 + \frac{1}{|a|} \log \frac{1 + e^{-|a| - |b|}}{1 + e^{|a| - |b|}}
\] (110)
Correction factor: Case 2: $|a| > |b|

\[
\frac{\log e^{\frac{|a|+|b|}{|a|+|b|}} + 1}{\min(|a|, |b|)} = \frac{\log e^{\frac{|a|+|b|}{|a|+|b|}} + 1}{|b|} = \frac{\log e^{\frac{|a|+|b|}{e^{|a|+|b|}+1}}}{|b|} = 1 + \frac{1}{|b|} \log e^{\frac{|a|+|b|}{e^{|a|+|b|}+1}} = 1 + \frac{1}{|b|} \log \frac{1 + e^{-|a|-|b|}}{1 + e^{-|a|-|b|}}
\]
\[
= 1 + \frac{1}{|b|} \log \frac{1 + e^{-(|a|+|b|)}}{1 + e^{-|b|-|a|}} = 1 + \frac{1}{\min(|a|, |b|)} \log \frac{1 + e^{-|a|+|b|}}{1 + e^{-|a|-|b|}}
\]

We observe that the rightmost term in (110) and (111) are the same. Moreover, we note that $1 + e^{-|a|+|b|} < 1 + e^{-|a|-|b|}$, i.e., we flip the denominator and the numerator and obtain

\[
\log \frac{1 + e^{-|a|+|b|}}{1 + e^{-|a|-|b|}} = -\log \frac{1 + e^{-|a|-|b|}}{1 + e^{-|a|+|b|}} < 0 \quad \forall |a|, |b| \in \mathbb{R}_+
\]

Finally we obtain

\[
\eta (a^\otimes b) = \text{sign}(a)\text{sign}(b) \cdot \min(|a|, |b|) \cdot \left(1 - \frac{1}{\min(|a|, |b|)} \cdot \log \frac{1 + e^{-|a|-|b|}}{1 + e^{-|a|+|b|}}\right)
\]

\[
\begin{array}{c}
0 \leq \text{multiplicative correction} \leq 1
\end{array}
\]

Note that for a low-complexity implementation of $a^\otimes b$ we may well use the approximation $a^\otimes b$ and perform a multiplicative correction by the factor within the brackets in (113). This factor only depends on the sums and the differences of the absolute values of the arguments $a$ and $b$, i.e., the correction factor could be stored in a two-dimensional table (in fact a matrix) and a “corrected” version of $a^\otimes b$ with very high accuracy could be implemented with low complexity and memory requirements.
... L-Values: Calculations with Parity Bits

Approximations related to the $\ominus$-operator:

- Frequently used in literature:

\[
\log(e^x + e^y) = \max(x, y) + \log(1 + e^{-|x-y|}), \quad x, y \in \mathbb{R} \tag{114}
\]

Proof:

Assume $x > y$. Then

\[
\log(e^x + e^y) = \log(e^x \cdot (1 + e^{y-x})) = x + \log(1 + e^{-|y-x|}) = x + \log(1 + e^{-|x-y|}) \tag{115}
\]

On the other hand assume $x < y$. Then

\[
\log(e^x + e^y) = \log(e^y \cdot (e^{x-y} + 1)) = y + \log(1 + e^{-|x-y|}) \tag{116}
\]

As the term $\log(1 + e^{-|x-y|})$ is the same in both cases, we obtain (114). ■

- (114) can be used to efficiently implement log-sums of exponential functions, as the additive correction term only depends on the absolute difference of the arguments. Such sums appear frequently in decoding algorithms for communication receivers.

- For the computation of $\log(e^x + e^y + e^z)$ (and any other extended log-sum of exponentials) we can use a nested version of (114):

\[
\log(e^x + e^y + e^z) = \log(e^c + e^z) \quad \text{with} \quad c = \log(e^x + e^y) \tag{117}
\]

- Note that the approximation (114) always UNDERestimates the true result as opposed to the approximation of the $\ominus$-operator.
... L-Values: Calculations with Parity Bits: Numerical examples

- Definitions:

\[ a \oplus b = \frac{e^a \cdot e^b + 1}{e^a + e^b} \quad \text{with} \quad a, b \in \mathbb{R} \]  \hspace{1cm} (118)

\[ a \ominus b = \text{sign}(a) \cdot \text{sign}(b) \cdot \min(|a|, |b|) \quad \text{with} \quad a, b \in \mathbb{R} \]  \hspace{1cm} (119)

- Examples:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>a \oplus b</th>
<th>a \ominus b</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>0.5</td>
<td>0.4813</td>
<td>0.5</td>
</tr>
<tr>
<td>-3.0</td>
<td>-1.0</td>
<td>0.8912</td>
<td>1.0</td>
</tr>
<tr>
<td>0.3</td>
<td>-6.0</td>
<td>-0.2985</td>
<td>-0.3</td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>0.7353</td>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
<td>-1.0</td>
<td>-0.4338</td>
<td>-1.0</td>
</tr>
</tbody>
</table>
Iterative Decoding

- Assume a coding scheme for 4 data bits \( I_1, \ldots, I_4 \) arranged in two rows and two columns.
- Each row and each column of the data bits is channel-coded by a single-bit parity check \( R_{ij} \).
- Coding scheme:

\[
\begin{array}{ccc}
I_1 & I_2 & R_{12} = I_1 \oplus I_2 \\
I_3 & I_4 & R_{34} = I_3 \oplus I_4 \\
R_{13} = I_1 \oplus I_3 & R_{24} = I_2 \oplus I_4
\end{array}
\]

General coding rule: \( R_{ij} = I_i \oplus I_j \) or equivalently \( R_{ij} \oplus I_i \oplus I_j = 0 \)

- Note: this is a rate-1/2 coding scheme, as 4 data bits are coded by 8 code bits (4 redundancy bits: \( R_{12}, R_{34}, R_{13}, R_{24} \)).
- Template of received real values at the channel output, converted into L-values:

\[
\begin{array}{ccc}
L(\tilde{z}_1|I_1) & L(\tilde{z}_2|I_2) & L(\tilde{z}_{12}|R_{12}) \\
L(\tilde{z}_3|I_3) & L(\tilde{z}_4|I_4) & L(\tilde{z}_{34}|R_{34}) \\
L(\tilde{z}_{13}|R_{13}) & L(\tilde{z}_{24}|R_{24})
\end{array}
\]

The bits \( I_j \) and the parity bits \( R_{ij} \) are transmitted and the resulting channel-outputs ("observations") at the receiving end are denoted by \( \tilde{z}_j \) and \( \tilde{z}_{ij} \) respectively.

- The conversion of channel observations \( \tilde{z}_j \) and \( \tilde{z}_{ij} \) into L-values \( L(\tilde{z}_j|I_j) \) and \( L(\tilde{z}_{ij}|R_{ij}) \) depends on the channel model; for an Additive White Gaussian Noise (AWGN) channel with BPSK modulation the receive values \( \tilde{z}_j \) and \( \tilde{z}_{ij} \) would be converted into an L-value according to \( L(\tilde{z}_j|I_j) = \frac{2}{\sigma^2} \cdot \tilde{z}_j \) and \( L(\tilde{z}_{ij}|R_{ij}) = \frac{2}{\sigma^2} \cdot \tilde{z}_{ij} \), with \( \sigma^2 \) the channel noise variance (which must be known and that comes at the price of channel estimation...).
... Iterative Decoding

- Template of receive values converted into L-values:

| $L(\tilde{z}_1|I_1)$ | $L(\tilde{z}_2|I_2)$ | $L(\tilde{z}_{12}|R_{12})$ |
|----------------------|----------------------|------------------------|
| $L(\tilde{z}_3|I_3)$ | $L(\tilde{z}_4|I_4)$ | $L(\tilde{z}_{34}|R_{34})$ |
| $L(\tilde{z}_{13}|R_{13})$ | $L(\tilde{z}_{24}|R_{24})$ |

- We consider all of the parity-checks separately and we know the decoding rules from above (see (83)):

- We initially assume “zero” a-priori information for all bits (data bits and parity bits).

- “Horizontal Decoding”:

First horizontal check:

\[
L(I_1|\tilde{z}_1, \tilde{z}_2, \tilde{z}_{12}) = (L(\tilde{z}_1|I_1) + L(I_1)) + \left((L(\tilde{z}_2|I_2) + L(I_2)) \oplus (L(\tilde{z}_{12}|R_{12}) + L(R_{12}))\right)
\]

\[
L(I_2|\tilde{z}_1, \tilde{z}_2, \tilde{z}_{12}) = (L(\tilde{z}_2|I_2) + L(I_2)) + \left((L(\tilde{z}_1|I_1) + L(I_1)) \oplus (L(\tilde{z}_{12}|R_{12}) + L(R_{12}))\right)
\]

\[
L(R_{12}|\tilde{z}_1, \tilde{z}_2, \tilde{z}_{12}) = (L(\tilde{z}_{12}|R_{12}) + L(R_{12})) + \left((L(\tilde{z}_1|I_1) + L(I_1)) \oplus (L(\tilde{z}_2|I_2) + L(I_2))\right)
\]

Second horizontal check:

\[
L(I_3|\tilde{z}_3, \tilde{z}_4, \tilde{z}_{34}) = (L(\tilde{z}_3|I_3) + L(I_3)) + \left((L(\tilde{z}_4|I_4) + L(I_4)) \oplus (L(\tilde{z}_{34}|R_{34}) + L(R_{34}))\right)
\]

\[
L(I_4|\tilde{z}_3, \tilde{z}_4, \tilde{z}_{34}) = (L(\tilde{z}_4|I_4) + L(I_4)) + \left((L(\tilde{z}_3|I_3) + L(I_3)) \oplus (L(\tilde{z}_{34}|R_{34}) + L(R_{34}))\right)
\]

\[
L(R_{34}|\tilde{z}_3, \tilde{z}_4, \tilde{z}_{34}) = (L(\tilde{z}_{34}|R_{34}) + L(R_{34})) + \left((L(\tilde{z}_3|I_3) + L(I_3)) \oplus (L(\tilde{z}_4|I_4) + L(I_4))\right)
\]
... Iterative Decoding

- Template of receive values converted into L-values:

\[
\begin{array}{c|c|c}
L(\tilde{z}_1|I_1) & L(\tilde{z}_2|I_2) & L(\tilde{z}_{12}|R_{12}) \\
L(\tilde{z}_3|I_3) & L(\tilde{z}_4|I_4) & L(\tilde{z}_{34}|R_{34}) \\
L(\tilde{z}_{13}|R_{13}) & L(\tilde{z}_{24}|R_{24}) & \\
\end{array}
\]

- “Vertical Decoding”:

First vertical check (see (83)):

\[
\begin{align*}
L(I_1|\tilde{z}_1, \tilde{z}_3, \tilde{z}_{13}) &= (L(\tilde{z}_1|I_1) + L(I_1)) + \left( (L(\tilde{z}_3|I_3) + L(I_3)) \oplus (L(\tilde{z}_{13}|R_{13}) + L(R_{13})) \right) \\
L(I_3|\tilde{z}_1, \tilde{z}_3, \tilde{z}_{13}) &= (L(\tilde{z}_3|I_3) + L(I_3)) + \left( (L(\tilde{z}_1|I_1) + L(I_1)) \oplus (L(\tilde{z}_{13}|R_{13}) + L(R_{13})) \right) \\
L(R_{13}|\tilde{z}_1, \tilde{z}_3, \tilde{z}_{13}) &= (L(\tilde{z}_{13}|R_{13}) + L(R_{13})) + \left( (L(\tilde{z}_1|I_1) + L(I_1)) \oplus (L(\tilde{z}_3|I_3) + L(I_3)) \right)
\end{align*}
\]

Second vertical check (see (83)):

\[
\begin{align*}
L(I_2|\tilde{z}_2, \tilde{z}_4, \tilde{z}_{24}) &= (L(\tilde{z}_2|I_2) + L(I_2)) + \left( (L(\tilde{z}_4|I_4) + L(I_4)) \oplus (L(\tilde{z}_{24}|R_{24}) + L(R_{24})) \right) \\
L(I_4|\tilde{z}_2, \tilde{z}_4, \tilde{z}_{24}) &= (L(\tilde{z}_4|I_4) + L(I_4)) + \left( (L(\tilde{z}_2|I_2) + L(I_2)) \oplus (L(\tilde{z}_{24}|R_{24}) + L(R_{24})) \right) \\
L(R_{24}|\tilde{z}_2, \tilde{z}_4, \tilde{z}_{24}) &= (L(\tilde{z}_{24}|R_{24}) + L(R_{24})) + \left( (L(\tilde{z}_2|I_2) + L(I_2)) \oplus (L(\tilde{z}_4|I_4) + L(I_4)) \right)
\end{align*}
\]
**Iterative Decoding**

- **Key idea of iterative decoding:**
  - alternate "horizontal" and "vertical" decoding
  - pass extrinsic information as new “a-priori” information to the other decoding step
- **Example:**
  Extrinsic information for bit $I_1$ can be generated from both the vertical and the horizontal parity check!
  - Extrinsic information for bit $I_1$ from *horizontal* decoder ...
    \[
    L_e^-(I_1) = (L(\bar{z}_2|I_2) + L(I_2)) \oplus (L(\bar{z}_{12}|R_{12}) + L(R_{12})) \quad \text{a-priori information} \quad L^1(I_1)
    \]
    ... used as additional a-priori information $L^1(I_1)$ for the vertical decoder.
  - Extrinsic information for bit $I_1$ from *vertical* decoder ...
    \[
    L_e^1(I_1) = (L(\bar{z}_3|I_3) + L(I_3)) \oplus (L(\bar{z}_{13}|R_{13}) + L(R_{13})) \quad \text{a-priori information} \quad L^-(I_1)
    \]
    ... used as additional a-priori information $L^-(I_1)$ for the horizontal decoder!
  - Same principle applies to all other bits (both data bits and parity bits).
  - Note that the “static” a-priori $L(I_1)$ describes the constant “bias” of a source and it just adds constantly to the channel information $L(\bar{z}_1|I_1)$. Therefore, static (non-zero) a-priori information creates a “better” channel, and there is no real need to distinguish between channel information and “static” source a-priori information.
  - To formulate the iterative decoding process in compact form we, therefore, introduce the abbreviations
    \[
    l_j = L(\bar{z}_j|I_j) + L(I_j) \quad \text{and} \quad l_{ij} = L(\bar{z}_{ij}|R_{ij}) + L(R_{ij}) \quad .
    \]
Iterative Decoding

- Template of “receive L-values plus static a-priori L values”:

  \[
  \begin{array}{ccc}
  l_1 & l_2 & l_{12} \\
  l_3 & l_4 & l_{34} \\
  l_{13} & l_{24} & \text{(##)}
  \end{array}
  \]

- Principle of iterative decoding:

  Step 1: Horizontal decoding / generate extrinsic L-values from horizontal checks:

  \[
  \begin{array}{ccc}
  L_e^{-}(I_1) \doteq l_2 \oplus l_{12} & L_e^{-}(I_2) \doteq l_1 \oplus l_{12} & L_e^{-}(R_{12}) \doteq l_1 \oplus l_2 \\
  L_e^{-}(I_3) \doteq l_4 \oplus l_{34} & L_e^{-}(I_4) \doteq l_3 \oplus l_{34} & L_e^{-}(R_{34}) \doteq l_3 \oplus l_4 \\
  \end{array}
  \]

  Step 2: Update L-value matrix by adding extrinsic L-values from Step 1 to the appropriate matrix cells:

  \[
  \begin{array}{ccc}
  l_1^- = l_1 + L_e^{-}(I_1) & l_2^- = l_2 + L_e^{-}(I_2) & l_{12}^- = l_{12} + L_e^{-}(R_{12}) \\
  l_3^- = l_3 + L_e^{-}(I_3) & l_4^- = l_4 + L_e^{-}(I_4) & l_{34}^- = l_{34} + L_e^{-}(R_{34}) \\
  l_{13} & l_{24} & \text{##}
  \end{array}
  \]

  Step 3: Vertical decoding / generate extrinsic L-values from vertical checks using the updated L-value matrix from Step 2:

  \[
  \begin{array}{ccc}
  L_e(I_1)^\dagger = l_3^- \oplus l_{13} & L_e(I_2)^\dagger = l_4^- \oplus l_{24} & \text{##} \\
  L_e(I_3)^\dagger = l_1^- \oplus l_{13} & L_e(I_4)^\dagger = l_2^- \oplus l_{24} & \text{##} \\
  L_e(R_{13}) = l_1^- \oplus l_3^- & L_e(R_{24}) = l_2^- \oplus l_4^- & \text{##}
  \end{array}
  \]
... Iterative Decoding

Step 4: Update L-value matrix by adding extrinsic L-values from Step 3 to the appropriate matrix cells:

\[
\begin{array}{ccc}
  l_1^- &=& l_1^- + L_e(I_1) \\
  l_2^- &=& l_2^- + L_e(I_2) \\
  l_3^- &=& l_3^- + L_e(I_3) \\
  l_4^- &=& l_4^- + L_e(I_4) \\
  l_{13}^- &=& l_{13}^- + L_e(R_{13}) \\
  l_{24}^- &=& l_{24}^- + L_e(R_{24}) \\
\end{array}
\]

Step 5: Check if the signs of the L-values in the matrix in Step 4 form a codeword. If so, STOP: the decoding result is given by the numbers in the table in Step 4.

If NO codeword has been found: add vertical extrinsic information from Step 3 to the original channel values \(l_1, \ldots, l_4\) and, this way, form the matrix

\[
\begin{array}{ccc}
  l_1' &=& l_1 + L_e(I_1) \\
  l_2' &=& l_2 + L_e(I_2) \\
  l_3' &=& l_3 + L_e(I_3) \\
  l_4' &=& l_4 + L_e(I_4) \\
  l_{13}' &=& l_{13} + L_e(R_{13}) \\
  l_{24}' &=& l_{24} + L_e(R_{24}) \\
\end{array}
\]

Note: horizontal extrinsic information NOT included!

Then go to Step 1 and perform horizontal decoding with this new matrix instead of (###).

Remarks:
- The iterative decoder described above is NOT an optimal decoder for the joint channel code (discussion to follow).
- There is more than one possible iterative decoding scheme for a code. For the given example, one could define a scheme which does only calculate extrinsic information for the data bits but not for the code bits, etc...
- For a numerical illustration of iterative decoding, we will use the approximation “\(\tilde{\odot}\)” to simplify calculations below.
... Iterative Decoding: Numerical Example

<table>
<thead>
<tr>
<th>Data bits and code bits:</th>
<th>Receive values (incl. static a-priori):</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1  +1  +1</td>
<td>+0.5  +1.5  +1.0</td>
</tr>
<tr>
<td>+1  −1  −1</td>
<td>+4.0  +1.0  −1.5</td>
</tr>
<tr>
<td>+1  −1  −1</td>
<td>+2.0  −2.5  −1.5</td>
</tr>
</tbody>
</table>

Extrinsic information from horizontal decoding:

<table>
<thead>
<tr>
<th>1st iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1.0  +0.5  +0.5</td>
</tr>
<tr>
<td>−1.0  −1.5  +1.0</td>
</tr>
<tr>
<td>-    -</td>
</tr>
</tbody>
</table>

Vertical receive values PLUS horizontal extrinsic data inf:

<table>
<thead>
<tr>
<th>1st iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1.5  +2.0  -</td>
</tr>
<tr>
<td>+3.0  −0.5  -</td>
</tr>
<tr>
<td>+2.0  −2.5  -</td>
</tr>
<tr>
<td>+2.0  +0.5  -</td>
</tr>
<tr>
<td>+1.5  −2.0  -</td>
</tr>
<tr>
<td>+1.5  −0.5  -</td>
</tr>
</tbody>
</table>

Horizontal receive values PLUS vertical extrinsic data inf:

<table>
<thead>
<tr>
<th>1st iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>+2.5  +2.0  +1.0</td>
</tr>
<tr>
<td>+5.5  −1.0  −1.5</td>
</tr>
<tr>
<td>-    -</td>
</tr>
</tbody>
</table>

Horizontal extrinsic information second decoding iteration

<table>
<thead>
<tr>
<th>1st iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1.0  +1.0  +2.0</td>
</tr>
<tr>
<td>+1.0  −1.5  −1.0</td>
</tr>
<tr>
<td>-    -</td>
</tr>
</tbody>
</table>

Iterations may continue and the L-values obviously keep growing.... so what would they actually mean in terms of reliability?
But the bit error was **corrected**!
Iterative Decoding: Correlation of L-Values

- Extrinsic L-values are simply added to the other L-values. This assumes that all the sources of information are independent of each other (see derivation of the \( \oplus \) -formula above!)
- This independence is, however, not guaranteed in the iterative decoder and that is why it is not optimal and why L-values keep growing with the number of iterations.
- Example: Consider bit \( I_1 \). Independence of L-values is violated, when an extrinsic L-value which involves \( I_1 \) “travels” back to bit \( I_1 \) as a-priori information from “elsewhere”. We show that this indeed will happen.
- We represent the coding scheme by a table and by an equivalent code graph:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_1 )</td>
<td>( I_2 )</td>
<td>( R_{12} )</td>
</tr>
<tr>
<td>( I_3 )</td>
<td>( I_4 )</td>
<td>( R_{34} )</td>
</tr>
<tr>
<td>( R_{13} )</td>
<td>( R_{24} )</td>
<td></td>
</tr>
</tbody>
</table>

with the coding rule \( R_{ij} = I_i \oplus I_j \) or, equivalently, \( R_{ij} \oplus I_i \oplus I_j = 0 \)

- Assume that \( I_2 = a \) and \( R_{12} = b \) are the receive values at the channel output. Due to the parity check, we form extrinsic information about bit \( I_1 \) by the standard operation \( a \oplus b \). This information is added to the receive value for bit \( I_1 \) and the passed to the evaluation of the next parity check involving \( I_3 \) and \( R_{13} \) (see arrows in code graph).
- After several “iterations” part of the initial extrinsic information \( a \oplus b \) travels back to bit node \( I_2 \), where now a modified value \( a + f(a \oplus b, \ldots) \) is used in another boxplus operation to compute extrinsic information for bit \( I_1 \).
  \( \Rightarrow \) this means that the same information is contained more than once in the extrinsic information for bit \( I_1 \)
  \( \Rightarrow \) this violates the independence assumption that is necessary to add L-values!
Iterative Decoding on Code Graphs

- In our iterative decoding example we have made a particular choice for decoding. We have, however, actually the freedom to update the bit nodes (in “codes-on-graphs-speak”) in any order and schedule we like: as none of the schemes will lead to an optimal decoder, it is at least in part a matter of trial-and-error what scheme performs best.

- The code performance depends, of course, on the choice of the ENcoding rule (i.e., the choice of the code graph). If we structure the code such, that it takes many decoding iterations before correlated information loops back to its origin, then we can expect better performance of the decoder; this does, however, not necessarily mean that the code itself is “good” when it is optimally decoded by maximum likelihood decoder (which we usually can’t try, due to complexity).

- In graph-speak, the latter issue boils down to avoid, by code construction, short cycles in the graph of a code. A cycle is essentially a closed path in a code graph, and this closed path never travels the same node twice (it just “comes and goes” but never comes back). In the figure we illustrate such a cycle (8 edges involved, therefore called “8-cycle”) which is at the same time the shortest possible cycle: this is called the “girth” of the code.
Iterative Decoding on Graphs: Numerical Example Revisited

- We start from the code graph and add the receive values to the bit nodes:

\[
\begin{array}{cccccccc}
I_1 & I_2 & I_3 & I_4 & R_{12} & R_{34} & R_{13} & R_{24} \\
+0.5 & +1.5 & +4.0 & +1.0 & +1.0 & -1.5 & +2.0 & -2.5 \\
\end{array}
\]

- In our example above we separated into “horizontal” and “vertical” decoding. This means we updated the bit nodes \(I_1, \ldots, I_4\) (data bits) by evaluating extrinsic information from the parity checks \(R_{12}\) and \(R_{34}\).

- After updating the L-values of the bits, we used their new values to evaluate the vertical checks, which are \(R_{13}\) and \(R_{24}\). After updating the bit nodes once again we would again evaluate the “horizontal” checks.

- In the code graph there is actually no notion of “horizontal” and “vertical”: this separation was actually only a convenient way to describe the coding scheme by a table on a piece of paper.

- The code graph above describes the very same code and allows for a much more generalised view on decoding. We could, for instance, calculate extrinisic information for all nodes in parallel and then also update them all with the newly computed extrinsic information.

- In the literature this generalised concept of decoding appears with Low-Density Parity-Check codes and the corresponding decoding algorithm is often called “sum-product algorithm” or “message passing decoding”.

- Low-Density Parity-Check Codes (LDPC) have only few bits involved in each parity check: this leads to a small number of “one”-entries in the parity check matrices (“low density”) and also to long cycles....
First Iteration: bit error!

- **Extrinsic L-values** are computed
- For simplicity we use the approximation $a \boxplus b = \text{sign}(a) \cdot \text{sign}(b) \cdot \min(|a|, |b|)$.
  (in practice we may use the accurate “⊞”)
- Arrows at the edges from the check nodes indicate **extrinsic L-values** that are added to the received channel information (see second iteration).

Second Iteration: bit error corrected!

- Similar to the first iteration but with different “channel values” that now contain the sum of the extrinsic information for a bit from the previous iteration.
- BUT: **new extrinsic information** for an edge is calculated from the new channel-values minus the extrinsic information that was passed over this edge towards the bit-node in the previous (first) iteration!
Third iteration:

- Updated channel values computed from original channel values (see first iteration) plus the sum of extrinsic information for a bit node from the previous iteration.
- New information (numbers in "{· , · }") passed back over the edges from a bit-node to check nodes is calculated from the new channel-values in the top row minus the extrinsic information that was passed over an edge towards the bit-node in the previous (second) iteration!
- Observation: L-values grow, due to cycles in the graph: this will never stop!

In LDPC decoding a code word check is performed after each decoding iteration. If a “valid” code word has been found (then, parity checks on hard decisions from the top-row of L-values all lead to “zero”) the decoding iterations are terminated; otherwise they continue until a code word is found or a maximum number of iterations has been carried out.
Iterative Decoding: Remarks

- It has become common to describe the whole receiver by a graph, including components such as demodulator, demapper, equaliser, channel decoder, source decoder. The concept of receiver signal processing can be seen as one of “message passing on graphs”, and the messages are usually L-values.

- The concept is nice and simple as long as “bits” are considered. Things get more complicated when correlated groups of bits such as quantiser indices are considered, as in joint source-channel decoding.

- L-values are not fundamental for the theory of such “receiver graphs”: they are however very convenient representations of the messages passed and simple operations can be used to process “soft” information in the form of L-values.

- The idea of “codes on graphs” forms a rather general framework with which receiver algorithms can be described and implemented!

- Iterative receiver concepts beat all other receiver structures “hands-down”; that’s why the initial idea (decoding of Turbo codes) has caused a revolution in communication theory since 1993 (when Turbo codes were first published: “Near Shannon limit error-correcting coding and decoding: Turbo-codes”, C. Berrou, A. Glavieux and P. Thitimajshima, IEEE International Conference on Communications ICC, Geneva, May 1993, pp. 1064–1070)
Iterative Source-Channel Decoding (ICSD)

⇒ How to extend the (very successful) iterative decoding principle to joint source and channel decoding.

Basic Concept:

- The residual redundancies in the, say, quantiser indices of a source codec can be interpreted as a “channel code”.
- A channel code makes code bits depend on each other; the dependencies are generated by the coding rule such as a matrix multiplication by a generator matrix (simple example: \( r = i_1 \oplus i_2 \)).
- Residual source-correlation from the source is usually linked to quantiser indices (i.e., “groups” of bits) which depend on each other e.g. in “time direction” but also due to a “non-flat” probability distribution of quantiser indices (e.g. optimal scalar fixed-rate quantiser).
- Hence, we need a means to exploit correlation on index level and convert it to bit level when we use a binary channel code (as usual) to channel-encode the output indices of a source codec.
- Note that we assume the encoder is given, i.e., we try to find a better receiver for a given transmitter. This approach is very much motivated by wireless standards (such as mobile radio) which have standardised transmitters with a fixed air interface: we can try to build better receivers and still stick to the standard!
... Iterative Source-Channel Decoding (ICSD): Illustration of Inter-Bit Dependencies

\[
p(x) = \begin{cases} 
\frac{1}{x_{\text{max}}} (1 - |x|/x_{\text{max}}), & |x| < x_{\text{max}} \\
0, & \text{else} 
\end{cases}
\]

- For single-bit parity check we assume equiprobable data bits (for simplicity).
- Note: in BOTH examples, the probability for each single bit to be “zero” is 0.5, i.e., no redundancy visible in the individual bits!
- Source correlation and channel-code constraints BOTH produce non-uniform probability distributions on groups of bits and non-uniform conditional probabilities.

Scalar Quantiser

<table>
<thead>
<tr>
<th>index (i)</th>
<th>repr. value (\hat{x}_i)</th>
<th>prob. (p_i)</th>
<th>Bitcode (b_1, b_2, b_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-7/8</td>
<td>1/32</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>-5/8</td>
<td>3/32</td>
<td>0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>-3/8</td>
<td>5/32</td>
<td>0 1 0</td>
</tr>
<tr>
<td>3</td>
<td>-1/8</td>
<td>7/32</td>
<td>0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1/8</td>
<td>7/32</td>
<td>1 0 0</td>
</tr>
<tr>
<td>5</td>
<td>3/8</td>
<td>5/32</td>
<td>1 0 1</td>
</tr>
<tr>
<td>6</td>
<td>5/8</td>
<td>3/32</td>
<td>1 1 0</td>
</tr>
<tr>
<td>7</td>
<td>7/8</td>
<td>1/32</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

Example: \(P\{b_3 = 0|b_1 = 1 = b_2\} = \frac{P\{b_3 = 0, b_1 = 1 = b_2\}}{P\{b_1 = 1 = b_2\}} = \frac{3/32}{4/32} = 3/4\)

Single-Bit Parity Check (SPC)

<table>
<thead>
<tr>
<th>Bits</th>
<th>SPC Codewords: (b_1 \oplus b_2 = b_3)</th>
<th>probability (to be sent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>1/4</td>
</tr>
<tr>
<td>0 0 1</td>
<td>invalid</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>invalid</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
<td>1/4</td>
</tr>
<tr>
<td>1 0 0</td>
<td>invalid</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 0 1</td>
<td>1/4</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 0</td>
<td>1/4</td>
</tr>
<tr>
<td>1 1 1</td>
<td>invalid</td>
<td>0</td>
</tr>
</tbody>
</table>

For \(b_3 = 0|b_1 = 1 = b_2\) we have \(P\{b_3 = 0|b_1 = 1 = b_2\} = \frac{P\{b_3 = 0, b_1 = 1 = b_2\}}{P\{b_1 = 1 = b_2\}} = 1\)
Iterative Source-Channel Decoding (ICSD): System Model

- autocorrelated input signal samples $X_k \Rightarrow$ quantized by $N$-bit vector $I_k$
- several mutually independent signals are transmitted in parallel at each time $k$
  $\Rightarrow$ model for multimedia source codecs

**Goal:** minimum distortion at the output, i.e.,

$$\tilde{x}_k = \arg \min \tilde{x}_k \ E_{X_k} \{ (\tilde{x}_k - X_k)^2 | \tilde{z}_k \}$$

$\Rightarrow$ standard result (see first lectures): minimum mean-square error estimator (with slight approximation):

$$\tilde{x}_k = E_{X_k} \{ X_k | \tilde{z}_k \} \approx \sum_{i=0}^{2N-1} y_i \cdot P(I_k = i | \tilde{z}_k)$$

(121)

- $y_i$: quantizer reproducer vectors
- $\tilde{z}_k$: all observed channel outputs up to time $k$ (for desired causal system), i.e., $\tilde{z}_k = \{ \tilde{z}_0, \tilde{z}_1, ..., \tilde{z}_k \}$
- we assume a binary channel code and binary modulation for simplicity
- key quantity for joint decoding: bit vector APP $P(I_k | \tilde{z}_k)$
... ICSD: Source-Redundancies: Example

\[ \mathcal{N}(0, 1) \]

\[ X_k \]

\[ Q \]

\[ \gamma() \]

\[ I_k \]

\[ z^{-1} \]

\[ a = 0.9 \]

- Gaussian source, first-order recursive filter
- optimal scalar quantizer, 5 bits per sample

→ Markov model, transition probabilities \( P(J_k | J_{k-1}) \)

→ bit mapping \( I_k = \gamma(J_k) \) must be included, so \( P(I_k | I_{k-1}) \) effective in system (just a re-mapping)

Note: due to required complexity limitation model assumption that \( I_k \) forms a 1st-order Markov random process. Not really true as \( P(I_k | I_{k-1}, I_{k-2}, ...) \neq P(I_k | I_{k-1}) \) although indeed \( P(X_k | X_{k-1}, X_{k-2}, ...) = P(X_k | X_{k-1}) \).

→ usually small loss and theory below would extend relatively straightforward
... ICSD: Source-Channel Decoding

- Coding scheme: serial concatenation of
  - explicit inner channel code (such as a convolutional code)
  - implicit source redundancies (outer “code”)
- Near-optimum joint decoding: compute exact values of bit vector APPs:

\[
P(I_k | \tilde{z}_k) = \frac{p(I_k, \tilde{z}_k, \tilde{z}_{k-1})}{p(\tilde{z}_k)} = \frac{p(\tilde{z}_k | I_k, \tilde{z}_{k-1}) \cdot p(I_k, \tilde{z}_{k-1})}{p(\tilde{z}_k)} = \frac{p(\tilde{z}_k | I_k, \tilde{z}_{k-1}) \cdot P(I_k | \tilde{z}_{k-1}) \cdot p(I_k)}{p(\tilde{z}_k)}
\]

\[
= \frac{p(\tilde{z}_{k-1})}{p(\tilde{z}_k)} \cdot \frac{P(I_k | \tilde{z}_{k-1})}{\text{const.} \neq f(I_k)} \cdot \frac{\tilde{z}_k}{\text{bit vector a priori info}} \cdot \frac{p(I_k) | I_k, \tilde{z}_{k-1}}{\text{channel term}}
\]

with

\[
P(I_k | \tilde{z}_{k-1}) = \sum_{I_{k-1}} P(I_k, I_{k-1} | \tilde{z}_{k-1}) = \sum_{I_{k-1}} \frac{P(I_k, I_{k-1}, \tilde{z}_{k-1})}{p(\tilde{z}_{k-1})} = \sum_{I_{k-1}} \frac{P(I_k | I_{k-1}, \tilde{z}_{k-1}) p(I_{k-1}, \tilde{z}_{k-1})}{p(\tilde{z}_{k-1})} = \sum_{I_{k-1}} \frac{P(I_k | I_{k-1}) \cdot P(I_{k-1} | \tilde{z}_{k-1})}{\text{Markov-model}} \cdot \frac{P(I_{k-1} | \tilde{z}_{k-1})}{\text{old bit vector APP}}
\]

Note: \(P(I_k | I_{k-1}, \tilde{z}_{k-1}) = P(I_k | I_{k-1})\) as, with \(I_{k-1}\) known, \(\tilde{z}_{k-1}\) will not improve the knowledge about \(I_k\).

\(\rightarrow\) recursive formula for a-priori term: no problem if the computation of the “old” bit vector APPs was feasible
ICSD: Source-Channel Decoding

Problem: computation of channel term $p(\tilde{z}_k \mid I_k, \tilde{z}_{k-1})$ in (122) is extremely complex if (as usual) many bit vectors are jointly channel-encoded.

- convenient notation: bit-vector $v_k$ that contains all the other bits that are jointly channel encoded together with the bits in $I_k$, but $v_k$ does not contain the bit-vector $I_k$; in practice $v_k$ may well consist of hundreds or thousands of bits.
- bit-vector $u_k$ contains all bits that are jointly channel-encoded, i.e.,

$$u_k \doteq (I_k, v_k).$$  \hspace{1cm} (124)

- now we rewrite the channel term:

$$p(\tilde{z}_k \mid I_k, \tilde{z}_{k-1}) = \sum_{\forall v_k} \frac{p(\tilde{z}_k, I_k, v_k, \tilde{z}_{k-1})}{p(I_k, \tilde{z}_{k-1})} = \sum_{\forall v_k} \frac{p(\tilde{z}_k \mid I_k, v_k, \tilde{z}_{k-1}) \cdot p(I_k, v_k, \tilde{z}_{k-1})}{p(I_k, \tilde{z}_{k-1})}$$

$$= \sum_{\forall v_k} \frac{p(\tilde{z}_k \mid I_k, v_k, \tilde{z}_{k-1})}{P(v_k \mid I_k, \tilde{z}_{k-1})} \cdot P(v_k \mid I_k, \tilde{z}_{k-1}) \cdot P(v_k \mid I_k, \tilde{z}_{k-1}) \cdot (125)$$

$\Rightarrow$ code word weighting factor:

- channel memoryless, so past of the received channel words contained in $\tilde{z}_{k-1}$ does not provide additional information about $\tilde{z}_k$ if $(I_k, v_k)$ is given, i.e., $p(\tilde{z}_k \mid I_k, v_k, \tilde{z}_{k-1}) = p(\tilde{z}_k \mid I_k, v_k)$.
- as channel code is deterministic, condition on $(I_k, v_k)$ can be equivalently replaced by a condition on the corresponding channel code word for the data bits, i.e., $p(\tilde{z}_k \mid I_k, v_k) = p(\tilde{z}_k \mid z(I_k, v_k))$; the latter is the pdf of the channel we assume to know.
⇒ a-priori information / other bits:

bits in $v_k$ and $I_k$ are independent (this was assumed in the system model) so we have

$$P(v_k \mid I_k, \tilde{z}_{k-1}) = P(v_k \mid \tilde{z}_{k-1}) .$$

(126)

This is what we know in advance about bit vector $v_k$ from the channel information $\tilde{z}_{k-1}$ received in the past.

- We obtain the optimum computation of the bit vector APPs for the joint source-channel decoding scheme (121):

$$P(I_k \mid \tilde{z}_{k-1}) = A_k \cdot P(\tilde{z}_k \mid I_k, \tilde{z}_{k-1}) \cdot P(I_k \mid \tilde{z}_{k-1})$$

(127)

$$\frac{1}{A_k} = \frac{p(\tilde{z}_k)}{p(\tilde{z}_{k-1})} = \sum_{I_k} P(\tilde{z}_k \mid I_k, \tilde{z}_{k-1}) \cdot P(I_k \mid \tilde{z}_{k-1})$$

(128)

$$P(I_k \mid \tilde{z}_{k-1}) = \sum_{I_{k-1}} \frac{P(I_k \mid I_{k-1})}{P(I_{k-1})} \cdot P(I_{k-1} \mid \tilde{z}_{k-1})$$

(129)

$$p(\tilde{z}_k \mid I_k, \tilde{z}_{k-1}) = \sum_{v_k} p(\tilde{z}_k \mid z(I_k, v_k)) \cdot P(v_k \mid \tilde{z}_{k-1})$$

(130)

→ equation (128) is an alternative way to compute the normalization factor $A_k \doteq p(\tilde{z}_{k-1})/p(\tilde{z}_k)$, based on the fact that the right-hand side of (127) must sum up to one.

→ due to complexity, scheme can only be used, when number of bits in $v_k$ (and $I_k$) is small: often NOT the case.

Example mobile radio: the enhanced full-rate speech codec generates 244 bits for each block of 20ms of speech (12.2kbits/s), most of the bits are coded by a convolutional code. If we were to decode by the scheme above a 5-bit energy index we would have in (130) to sum over $2^{244-5}$ different combinations $v_k$ could possibly take.
... ISCD: less complex computation of the "Channel Term" by an approximation

- We re-write the left-hand side of (130):

\[
p(\tilde{z}_k | I_k, \tilde{z}_{k-1}) = \frac{p(\tilde{z}_k, I_k, \tilde{z}_{k-1})}{p(I_k, \tilde{z}_{k-1})}. \tag{131}
\]

Then we replace the bit-vector probability densities by the products over the corresponding bit probability densities:

\[
p(\tilde{z}_k | I_k, \tilde{z}_{k-1}) \approx \prod_{n=1}^{N} \frac{p(\tilde{z}_k, I_{k,n}, \tilde{z}_{k-1})}{p(I_{k,n}, \tilde{z}_{k-1})} = \prod_{n=1}^{N} \frac{P(I_{k,n} | \tilde{z}_k)}{p(\tilde{z}_{k-1})}. \tag{132}
\]

- We insert (132) into (127) and we obtain the approximation for the bit-vector APPs

\[
P(I_k | \tilde{z}_k) \approx P(I_k | \tilde{z}_{k-1}) \cdot \prod_{n=1}^{N} \frac{P(I_{k,n} | \tilde{z}_k)}{P(I_{k,n} | \tilde{z}_{k-1})}. \tag{133}
\]

- The bit a-posteriori probabilities \( P(I_{k,n} | \tilde{z}_k) \) can be efficiently computed by a symbol-by-symbol APP algorithm for a binary convolutional channel code with a small number of states. Note: here a "symbol" is a "bit"!! Details of the algorithms are given in

⇒ To understand iterative source-channel decoding, we don’t need to know the exact details of this APP algorithm!
ISCD: less complex computation of the “Channel Term” by an approximation

- Although the APP-algorithm computes $P(I_{k,n} \mid \tilde{z}_k)$, and hence it only decodes the currently received channel word $\tilde{z}_k$, it still uses all the received channel words $\tilde{z}_k$ up to the current time $k$, because

$$P(I_{k,n} \mid \tilde{z}_{k-1}) = \sum_{\forall I'_{k,n} : I'_{k,n} = I_{k,n}} P(I'_k \mid \tilde{z}_{k-1})$$  \hspace{1cm} (134)

is used as bit-based a-priori information for the APP algorithm, with $P(I_k \mid \tilde{z}_{k-1})$ being computed according to (129) from the “old” bit vector APPs and the Markov source model.

- When we use $P(I_{k,n} \mid \tilde{z}_{k-1})$ according to (134) as a-priori information for the APP algorithm which will produce $P(I_{k,n} \mid \tilde{z}_k)$ as a decoding result we can interpret the fraction of both in (133) as the extrinsic information $P^{(C)}_{e}(I_{k,n})$ we get from the channel decoder:

$$P(I_k \mid \tilde{z}_k) \approx P(I_k \mid \tilde{z}_{k-1}) \cdot \left[ \prod_{n=1}^{N} P^{(C)}_{e}(I_{k,n}) \right] \hspace{1cm} \text{with} \hspace{1cm} P^{(C)}_{e}(I_{k,n}) = \frac{P(I_{k,n} \mid \tilde{z}_k)}{P(I_{k,n} \mid \tilde{z}_{k-1})}.$$  \hspace{1cm} (135)

Note that we have introduced the superscript “(C)” to indicate that $P^{(C)}_{e}(I_{k,n})$ is the extrinsic information produced by the APP channel-decoding algorithm.

- Note the typical characteristic of extrinsic information: we use the decoding result of the APP decoder in the numerator and the “input” a-priori information to the APP decoder in the denominator: this removes a-priori information from the result and reduces the decoder output to the “novelty” it contains! In L-value notation (see later) the division will turn into a subtraction – as we know it from our earlier discussion of iterative channel decoding.
... ISCD

- In principle, we now could compute the mean-square estimates

\[ \tilde{x}_k = \sum_{\forall I_k} y_{I_k} \cdot P(I_k | \tilde{z}_k) \]  

(136)

(our joint source-channel decoder outputs) using the approximated \textit{bit-vector} APPs from (135).

- The \textit{bit-vector} APPs from (135) are only approximations of the correct values, since the \textit{bit} a-priori informations, that were used for APP channel decoding, did \textit{not} contain the mutual dependencies (which we actually want to exploit for decoding) of the bits within the bit-vectors: they were removed by the summation in (134) to limit complexity!

- The idea to make up (in part) for this loss of information due to the bit-based information exchange between the constituent APP decoders is to perform several iterations of the constituent APP decoders.

- Hence, new \textit{bit} APPs are computed from the intermediate results (135) for the bit vectors by “marginalisation”:

\[ P^{(S)}(I_{k,n} | \tilde{z}_k) = \sum_{\forall I'_k : I'_k,n = I_{k,n}} P(I'_k | \tilde{z}_k) . \]  

(137)

The superscript “(S)” was introduced, as (137) is computed after source decoding. Now, we can derive new \textit{bit} extrinsic information from the source decoder by

\[ P^{(S)}_e(I_{k,n}) \doteq P^{(S)}(I_{k,n} | \tilde{z}_k) / P^{(C)}_e(I_{k,n}) . \]  

(138)

- The extrinsic information from the last run of the channel decoder is removed by the denominator in (138), since we do not want to loop back information to the channel decoder that it has produced itself in the previous iteration.
• The extrinsic information $P_e^{(S)}(I_{k,n})$ computed by (138) is used as the new a-priori information for the second and similarly for further runs of the APP channel decoder.

• At time $k = 0$ we have to initialise the bit vector a-priori information $P(I_k | \tilde{z}_{k-1})$. A possible choice is

$$P(I_k | \tilde{z}_{k-1})|_{k=0} = P(I),$$

(139)

i.e., we can use the static (hence no time index) unconditional probability distribution of the source-encoder output bit vectors as an initialisation for the decoding iterations.

• Note: this principle to define iterative source-channel decoding is arguably reasonable, but by no means straightforward or a even logical consequence derived by maths: as our approach contains a couple of approximations, the ultimate justification that it makes sense is given by the simulation results!

• Note that the key to limited complexity is to break the exponential law in (130); and this can be achieved by looking at bits separately instead of full code words or bit vectors.

• Role of interleaver: makes subsequent bits in the channel encoder input “virtually” independent, i.e., it compensates as good as possible for the approximations used in our derivations.
ICSD: Summary and Illustration

\[
P(I_k | \tilde{z}_{k-1}) = \sum_{I_{k-1}} P(I_k | I_{k-1}) \cdot P(I_{k-1} | \tilde{z}_{k-1})
\]

\[
P(I_{k,n} | \tilde{z}_{k-1}) = \sum_{I_k | I_{k,n}} P(I_k | \tilde{z}_{k-1})
\]

bit vector a priori info

old bit vector APP

bit extrinsic info

\[
P^{(S)}(I_{k,n} | \tilde{z}_k) = \sum_{I_k | I_{k,n}} P(I_k | \tilde{z}_k)
\]

\[
P^{(S)}(I_k | \tilde{z}_k) \implies P^{(S)}(I_{k,n} | \tilde{z}_k) / P_e^{(C)}(I_{k,n})
\]

bit APP

bit extr. info

\[
\tilde{x}_k = \sum_{I_k} y_{I_k} \cdot P(I_k | \tilde{z}_k)
\]

quantizer level

decoder output

AP source decoder

APP channel decoder
(incl. interleaving)

0th iteration

1, 2, ..., iteration

bit a priori info

Wireless Communications 2

Iterative Source-Channel Decoding
... ICSD: Efficient Implementation of the Decoder means of by L-Values

- Implementation by algorithms that exchange L-Values
- Parallel source-signal channels are included, but the bit vectors are shown for $I_k$ only in the figure below.
- The quantities in the figure such as $L^{(S)}(I_k)$ are to be interpreted as vectors of L-values, i.e.,

$$L^{(S)}(I_k) = \{L^{(S)}(I_{k,1}), \ldots, L^{(S)}(I_{k,N})\}$$

(140)

- Remember that, in generalised form, $U_k = \{\ldots, I_k, \ldots\}$ with $I_k$ the source index we consider.
- Similar to (140), e.g., $L^{(S)}(U_k)$ is a vector of L-values corresponding to the random bit vector $U_K$.
ICSD: Efficient Implementation of the Decoder means of by L-Values

- Soft-values passed between the APP decoders for the channel code and the source redundancy are now L-values.
- Example: extrinsic L-value from the APP source decoder for the bit \( I_{k,n} \) is defined by (natural log)

\[
L_e^{(S)}(I_{k,n}) \doteq \log \frac{P_e^{(S)}(I_{k,n} = 0)}{P_e^{(S)}(I_{k,n} = 1)} .
\] (141)

- Similar definition for the other L-values.

**APP Channel Decoding with L-values**: BCJR-algorithm, an efficient implementation of the APP-algorithm, for decoding of binary convolutional codes, can be completely carried out in the L-value domain (Log-MAP-algorithm).


- The received channel values \( \tilde{z}_{k,l} \) are converted to L-values for the input of the APP channel decoder by multiplication with the factor \( L_c = 4 \frac{E_s}{N_0} \). This follows from the definition of the L-values and the pdf of the Gaussian channel we assume for simplicity (we use (61) with \( \sigma^2 = N_0/(2E_s) \)):

\[
L(\tilde{z}_{k,l} | Z_{k,l}) = \log \frac{p(\tilde{z}_{k,l} | Z_{k,l} = 0)}{p(\tilde{z}_{k,l} | Z_{k,l} = 1)} = \log \frac{\exp \left( -\frac{1}{2\sigma^2} (\tilde{z}_{k} - 1)^2 \right)}{\exp \left( -\frac{1}{2\sigma^2} (\tilde{z}_{k} + 1)^2 \right)} = 4 \frac{E_s}{N_0} \tilde{z}_{k,l} = L_c \cdot \tilde{z}_{k,l} .
\] (142)

- The APP algorithm in the L-value domain will produce (including the interleaver/de-interleaver):

\[
L^{(C)}(U_k) \doteq \left\{ L(U_{k,m} | \tilde{z}_k) = \log \frac{P(U_{k,m} = 0 | \tilde{z}_k)}{P(U_{k,m} = 1 | \tilde{z}_k)}, \quad m = 1, 2, \ldots, M \right\}
\] (143)

with \( \tilde{z}_k \) the receive vector for the WHOLE channel code word and \( M \) the total number of data bits.
- **APP Source Decoding with L-Values:** APP source decoder operates with probabilities of bit vectors, i.e., non-binary quantities, for which L-values are not as useful as for bits. Hence, an interface to the APP channel decoder operating in the L-value domain is needed.

- The computation of the bit-vector APPs by (135) (repeated here)

\[
P(I_k | \tilde{z}_k) \approx P(I_k | \tilde{z}_{k-1}) \cdot \left[ \prod_{n=1}^{N} P_e^{(C)}(I_{k,n}) \right] \quad \text{with} \quad P_e^{(C)}(I_{k,n}) = \frac{P(I_{k,n} | \tilde{z}_k)}{P(I_{k,n} | \tilde{z}_{k-1})}.
\]

requires the bit-probabilities \( P_e^{(C)}(I_{k,n}), n = 1, \ldots, N \). The latter can be computed from the output L-values \( L_e^{(C)}(I_{k,n}) \) of the APP channel decoder by inversion of the L-value definition (141):

\[
P_e^{(C)}(I_{k,n} = b_n) = \frac{\exp \left( L_e^{(C)}(I_{k,n}) \right)}{1 + \exp \left( L_e^{(C)}(I_{k,n}) \right)} \cdot \exp \left( - L_e^{(C)}(I_{k,n}) \cdot b_n \right), \quad b_n \in \{0, 1\}.
\]

**Remark:** Up to now, we did not make any difference between the notation for a random variable \( I_{k,n} \) and the values this random variable may take; when writing \( P(I_{k,n}) \) we actually mean the probability that the random variable \( I_{k,n} \) will take one of the possible values as a realisation.

When we now convert between probabilities and L-values we have, however, to distinguish between the random variable and its realisations because the L-value \( L(I_{k,n}) \) is a property of the random variable but there is no “number” for \( I_{k,n} \) to insert as it is the case for \( P(I_{k,n}) \). This “number” is denoted by \( b_n \in \{0, 1\} \), with \( b \) being the vector that contains \( b_n \) at position \( n \).
Since in (144) the product over all these probabilities is computed for one bit-vector, this operation can be simplified by inserting (145) into (144), i.e.,

$$P(I_k = b | \tilde{z}_k) = G_k \cdot \left[ \prod_{n=1}^{N} \exp \left( - L_e^{(C)}(I_k,n) \cdot b_n \right) \right] \cdot P(I_k = b | \tilde{z}_{k-1}) ,$$

(146)

with $b \doteq \{b_1, ..., b_N\} \in \{0, 1\}^N$ and the normalizing constant

$$G_k \doteq \prod_{n=1}^{N} \frac{\exp \left( L_e^{(C)}(I_k,n) \right)}{1 + \exp \left( L_e^{(C)}(I_k,n) \right)}$$

(147)

that does not depend on the particular choice $I_k = b$ of the random bit-vector.

Now, the product in (146) can be turned into a summation in the L-value domain:

$$P(I_k = b | \tilde{z}_k) = G_k \cdot \exp \left( - \sum_{n=1}^{N} L_e^{(C)}(I_k,n) \cdot b_n \right) \cdot P(I_k = b | \tilde{z}_{k-1}) .$$

(148)

Thus, the extrinsic L-values $L_e^{(C)}(I_k,n)$ from the APP channel decoder can be integrated into APP source decoding without converting the individual L-values back to probabilities if (148) is used instead of (144). $\Rightarrow$ Simplification that also has strong numerical advantages!

Additionally, the left-hand side of (148) is a probability that must sum up to one over all bit-vectors $I_k$. Hence, the constant $G_k$ can be computed from this condition instead of (147).
The computation of new bit APPs (from (148)) within the iterations must be still carried out by (137) repeated here:

\[
P^S(I_{k,n} = b_n | \tilde{z}_k) = \sum_{\forall I'_{k,n}=b_n} P(I'_{k} | \tilde{z}_k).
\] (149)

The derivation of the extrinsic L-values \(L^S(I_{k,n})\), that are generated by the APP source decoder, can be simplified, since (138) requires a division which is turned into a simple subtraction in the L-value-domain:

\[
L^S(I_{k,n}) = \log \frac{P^S(I_{k,n} = 0 | \tilde{z}_k) / P^C(I_{k,n} = 0)}{P^S(I_{k,n} = 1 | \tilde{z}_k) / P^C(I_{k,n} = 1)} = L^S(I_{k,n}) - L^C(I_{k,n}).
\] (150)

Thus, in the whole ISCD algorithm the L-values \(L^C(I_{k,n})\) from the APP channel decoder are used and the probabilities \(P^C(I_{k,n})\) are no longer required:
ICSD: Quantizer Bit Mappings: \( I = \gamma(J) \)

- Assume quantizer index at time \( k-1 \): \( J_{k-1} = 1 \) (known with high probability)
  - At time \( k \), quantizer levels \( y_0, y_1, y_2 \) are highly probable, due to auto-correlations.
- Assumption: Channel decoder produces perfect extrinsic information for the two front bits (both are "0")
- Now APP source decoder tries to generate extrinsic information for the rightmost bit:
  - Optimized mapping: safe decision in favour of "1" as \( y_7 \) is highly improbable, due to correlation
  - "Strong" extrinsic information to aid the channel decoder in the next iteration
- Bit mapping can be optimised for iterative decoding!
ICSD: Quantizer Bit Mappings: \( I = \gamma(J) \)

- Optimization criterion: average distance \( D \) in the source signal space should be large for any two quantizer levels whose bit-mappings differ in exactly one bit position
  
  Note: this is very similar to Bit-Interleaved Coded Modulation with Iterative Decoding (BICM-ID)
  
  → Optimisation idea easily extendable to vector quantization

- Brute-force approach infeasible as for an \( N \)-bit quantizer \( 2^N! \) different mappings exist (factorial of the integer \( 2^N \))

  Example: for a \( N = 6 \) bit quantiser, we have \( 2^6! = 64! \approx 1.27 \cdot 10^{89} \) different mappings.

- Suboptimal solution: **Binary Switching Algorithm (BSA)**

  originally developed for index optimization in robust vector quantization (Zeger, Gersho, 1990)

  principle of the BSA, adapted to ISCD:
  
  - \( D \) is maximized step-by-step, by switching the bit mappings of selected pairs of quantizer levels
  
  - the mapping of the quantizer level that causes the lowest contribution to the average distance \( D \) is tried to be switched first (note: we wish to maximise average distance!)
  
  - switching is carried out until no further increase of \( D \) is possible
  
  - locally optimal solution
Some Simulation Results

Source signals:
• Gauss-Markov source, \( H(z) = \frac{z}{z-a} \), \( a = 0.9 \)
• scalar optimal 5-bit quantizer (Lloyd-Max)
• three bit mappings: natural binary, Gray, and optimized
• 50 of such signals transmitted in parallel \( \rightarrow \) 250 data bits

Channel coding:
• random interleaving of the data bits
• rate–1/2 convolutional code, memory 4, recursive systematic encoder
• terminated after each block of 250 data bits
• exact code rate \( R = 250/508 \)
• Plots: \( E_b/N_0 = \frac{E_s}{N_0} \cdot \frac{1}{R} \), with \( \sigma^2 = N_0/(2E_s) \) the variance of the receiver noise on the Gaussian channel.

Iterative source-channel decoding:
• 0, 1, 2, 4, 9 iterations
... Some Simulation Results

![Graph showing some simulation results. The x-axis represents $E_b/N_0$ in dB, and the y-axis represents average source SNR in dB. The legend indicates different mapping iterations: opt. mapping, 9 it.; opt. mapping, 4 it.; opt. mapping, 2 it.; opt. mapping, 1 it.; opt. mapping, 0 it.]}
Some Simulation Results

![Graph showing iterative source-channel decoding results. The x-axis represents $E_b/N_0$ in dB, and the y-axis represents the average source SNR in dB. The graph compares different mapping strategies: optimal mapping with 9, 4, 2, 1, and 0 iterations, as well as a natural mapping. The graph highlights the performance improvement as the number of iterations increases.]
... Some Simulation Results

![Graph showing simulation results with labels for average source SNR in dB, $E_b/N_0$ in dB, channel SNR gain, source SNR gain, and mappings including natural, Gray, and optimal mappings with 9, 1, and 0 iterations.](image-url)
ICSD: Conclusions

- Iterative source-channel decoding exploits the source redundancies with realizable complexity
- Large gain by optimized quantizer bit mappings, especially for strongly correlated sources
- For $E_b/N_0 > 0$ dB the optimized bit mapping outperforms the other mappings
  → but it takes some iterations
- Gray and natural mapping: no further gain if more than two iterations are performed
Iterative source-channel decoding was discussed in some detail, as it allows for a good illustration of the general “turbo” decoding/detection principle. Very similar arguments will apply for advanced coded modulation schemes (such as Bit-Interleaved Coded Modulation with Iterative Decoding).

For the coded-modulation case the optimisation scheme for the bit mappings can also be applied.

Soft-in/Soft-out processing has revolutionised receiver design: the key problem for any receiver is to define the suitable APP decoding/demapping algorithms. Once this has been done, the information exchange between the components always follows the same principle: generate extrinsic information and use it as a-priori information elsewhere.

Try to avoid correlated L-values by using interleavers – some people speak of the “interleaving gain” in the context of block size.
Cross-Layer Design

Recently, the topic of “cross-layer design” has gained a lot of attention in research.

The idea is to get a performance gain in communication networks from giving up the strict hierarchy of the so-called OSI reference model (Open Systems Interconnection Reference Model).

As source coding usually is based in the application layer #7 and channel coding in the physical layer #1, joint source-channel coding is in fact a cross-layer concept.

However, there are other cross-layer ideas, some highly efficient, that can only be understood in the context of multi-user systems.

The field is very wide and research is ongoing so there is as yet no established unified theory and no text-book reference.

Common to most works is that wireless (access) networks with scarce resources are used as a justification for breaking with the OSI model.

We use a generic cellular system to introduce the basic notions and develop some theory.

Central to cross-layer design is the exploitation of multi-user diversity.
A Generic Cellular System

Diverse traffic with different Quality-of-Service criteria (bit rate, bit error rate and delay)

Time variant channels with independent fading

System performance can not be maximised by serving users separately and independently!

Cross-Layer Concept required!
Possible Ways to improve System Performance

- **Application-Specific Resource Allocation:**
  - **Cross-Layer Scheduling** at the Base-Station for Uplink und Downlink
    - **Multiuser Diversity Gains** possible, if channel coefficients are known at the transmitter
    - Different service requirements of the applications may or may not complement each other

- **Application-Specific Efficient Use of Allocated Resources:**
  - Classical “Physical-Layer Design”
  - Modulation, equalisation, MIMO, channel coding, source coding, ...
  - Includes joint source-channel coding

- **Error Concealment:**
  - Fall-back solution when transmission scheme fails: completely dependent on the application
  - Sometimes impossible (data file transfer): solution on protocol level required (such as ARQ)

- **User Cooperation:**
  - “Relaying”: very helpful to serve users at the cell boundaries
  - Virtual MIMO systems and “Cooperative Diversity”; specific channel coding schemes
  - Several dimensions of additional complexity (routing, protocols, ...)
Cross-Layer Scheduling and Multiuser Diversity Gains

Multiuser Diversity:

- "Many" users in the system: one of them very likely to have a large channel coefficient.
  - Make efficient use of power by transmitting to users with large channel coefficients
- "Multiuser Diversity Gains"
  - There is a trade-off with Quality-of-Service!

Full channel-knowledge at the transmitter is crucial to achieve Multi-User Diversity Gains

Knowledge about the channel and the applications should be exploited:
  - Cross-Layer Scheduling!

“Scheduling” means “to know whose time has come”:
  - includes control of user access to the channel and resource allocation (e.g. power)
**Generic System Example / Downlink:**

- Cellular system with \( J = 8 \) users
- Bandwidth: 1.25 MHz, i.e., symbol time \( t_s = 0.4 \, \mu s \) (HDR/CDMA 2000)
- Time slot: \( t_{\text{slot}} = 0.4 \, \text{ms} \), i.e., 1000 symbols per time slot (1.67 ms in HDR/CDMA 2000)
- System example **used throughout the talk**:

<table>
<thead>
<tr>
<th>User ( j )</th>
<th>Rate Request ( r_j ) in kbits/s</th>
<th>Delay Limit ( \tau_j ) Slots (Time/ms)</th>
<th>Potential Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50.0</td>
<td>50 (20)</td>
<td>Mobile Phonecall</td>
</tr>
<tr>
<td>2</td>
<td>100.0</td>
<td>186</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>150.0</td>
<td>321</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>200.0</td>
<td>457 (183)</td>
<td>Audio Streaming</td>
</tr>
<tr>
<td>5</td>
<td>250.0</td>
<td>593</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>300.0</td>
<td>729</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>350.0</td>
<td>864</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>400.0</td>
<td>1000 (400)</td>
<td>Video Streaming</td>
</tr>
</tbody>
</table>

- All users are "backlogged"; if users have no data to transmit they are not scheduled.
- Queue stability: not a problem as data rates have to be delivered within the delay limits.
- For binary data (emails, web surfing): transmit buffer size determines delay limits.
Delay Constraints:

- Formal definition of an “average” rate within a sliding rectangular window of length $\tau_j$:

$$\bar{R}_j(k) \doteq \frac{1}{\tau_j} \sum_{i=0}^{\tau_j-1} R_j(k-i) \quad \text{for all users } \ j = 1, 2, \ldots, J$$

  - $k$: time slot index $k = 0, 1, 2, \ldots$
  - $\tau_j$: delay constraint – counted in time slots – for user $j$
  - $R_j(\kappa)$ is the rate achieved by user $j$ in time slot $\kappa$; “zero” when not scheduled!
  - $\bar{R}_j(k)$ is the average rate at time $k$ achieved in the past $\tau_j$ slots

- Formal definition of a “hard” delay constraint, with $r_j$ the rate requested by user $j$:

$$\bar{R}_j(n \cdot \tau_j - 1) \geq r_j \quad \forall \ n = 1, 2, \ldots,$$

  - This constraint is practically very useful
    - directly related to block-wise source and channel coding
    - source signal blocks don’t overlap
Source signal Red User

Shared Channel Access (TD)

Rate

Buffering Block A Buffering Block B Buffering Block C Buffering Block D

Coding Block A Coding Block B Coding Block C

Transmission Transmission Block A Block B

Decoding Block A

τ_j = 12 (red user)
Channel Model:

- Within time slots: constant channel coefficients but independent fading between slots.
  - **Block-Fading Model**: very popular in Information Theory to model cellular wireless systems

- $H_j(k)$: complex baseband-channel coefficient of user $j$ in time slot $k$
  - known at both the transmitter and the receiver (where it is estimated and returned)
  - allows for coherent detection, i.e., no phase error and same channel gain $|H_j(k)|$ in “I” and “Q”

- Average (“long-term”) power gain (numbers assumed in table):
  
  $$G_j = \mathbb{E}_{H_j}\{|H_j|^2\} \approx \frac{1}{L} \sum_{k=1}^{L} |h_j(k)|^2 \quad \text{for large} \quad L$$

  - assumes ergodic random process for channel coefficients

- For simulations below we assume Rayleigh fading for simplicity, i.e.
  
  $$|H_j| = \sqrt{G_j \cdot W} \quad \text{with} \quad p_W(w) = 2w \cdot e^{-w^2} \quad \text{(Rayleigh-Distribution, } w \geq 0)$$
What does Information Theory have to say?

- For a couple of special cases, Multiuser Information Theory tells us what optimal schemes are for scheduling and power allocation.

- No need to deal with classical “best-effort” scheduling techniques known from wired networks:
  - not suitable for the problem we consider – particularly not for delay limited applications.
  - We don’t want a best-effort service – we want a guaranteed service!
  - there is no notion of “multiuser diversity gains” in wired networks

- It is NOT true that Information Theory systematically ignores delay, “fairness” or “QoS”
  - But suitable analytical results are very hard to get!

Some notes on Block-Fading Gaussian Channel Models.
Why Block-Fading Gaussian Channels are popular in Information Theory

Within each block $k$, the receive signal of user $j$ is

$$y_{j,k}(i) = |h_{j,k}(i)| \cdot x_{j,k}(i) + n_{j,k}(i) \quad \text{with } h_{j,k}(i) \text{ being estimated at the receiver!}$$

$n_{j,k}(i)$: Gaussian noise with variance $N_0$; $x_{j,k}(i)$: transmit signal with average power $P_j$.

If $N \to \infty$, then the standard Gaussian channel capacity can be achieved within each block, i.e., the achievable rate for user $j$ in slot $k$ equals (carrier-modulated system, i.e., “complex modulation”)

$$R_j(k) = \log_2 \left( 1 + \frac{|H_{j,k}(k)|^2 \cdot P_j}{N_0} \right) \quad \text{in bits of information per channel-use}$$

However, the rate $R_j(k)$ is a random variable as $|H_{j,k}(k)|$ is one!
Ergodic Capacity: Classical Shannon Capacity:
Average over “many” blocks, i.e., $M \to \infty$, each block also has infinite length $N \to \infty$:

$$C_j = \lim_{M \to \infty} \frac{1}{M} \sum_{k=1}^{M} R_j(k) = \int_{0}^{\infty} \log_2 \left( 1 + \frac{|h_j(k)|^2 \cdot P_j}{N_0} \right) p_{H_j}(h_j) dh_j$$

This is a “number” that equals ergodic capacity for a fading channel where channel coefficients may change every channel symbol.

- Can be achieved either by separate coding within each block or by coding across all $M$ blocks.
- Optimal power allocation schemes are known (which implicitly includes scheduling)!
- Problem: Delay is ignored as $M \to \infty$ and $N \to \infty$

$\varepsilon$-Outage-Capacity for limited number $M$ of blocks:

$$\text{If } \Pr \left\{ \frac{1}{M} \sum_{k=1}^{M} R_j(k) < C_j \right\} = \varepsilon \text{ then } C_j \text{ is the } \varepsilon\text{-Outage-Capacity}$$

- Choice of $M$ expresses delay constraint: very useful (and tricky as still $N \to \infty$)!
- Problem: results only known for $M \leq 2$ for single-user case and $M = 1$ for the multiuser case
Delay-Limited Capacity / Zero-Outage Capacity
- Same as $\varepsilon$-Outage-Capacity with $\varepsilon = 0$
- Ergodic capacity (with $M \to \infty$) is strictly larger than delay-limited capacity ($M \ll \infty$).
- Multiuser case: ergodic capacity region contains zero-outage region (NOT true for any $\varepsilon > 0$)

Expected Capacity
- Averaged rate treated as a random variable: “capacity” as the expected value of this RV

Minimum-Rate Capacity
- Combination of ergodic and zero-outage capacity: power allocation for maximisation of ergodic capacity subject to minimum-rate requirements for all users in all fading states.
- Analytical results again only for $M = 1$; result is “zero” for Rayleigh fading.

Summary:
- Essentially, there are no capacity results for the multiuser case with optimal power allocation for $1 < M \ll \infty$: but this case is practically very important and we consider it!
- Zero-Outage and Minimum-Rate Capacity results for $M = 1$ are not very useful: both capacities are “zero” for a Rayleigh fading channel. What would this tell us about $M = 50$?
- Pragmatic: “some” $M > 1$ may be “large enough” so that ergodic capacity is a close upper bound.
Ergodic Capacity Region of a (Block-)Fading Gaussian Broadcast Channel

Assumptions:
- Fast fading so that capacity computations average over many channel realisations.
- Full channel information (F-CSI): both receiver and transmitter know channel coefficient realisations.
- All random processes are ergodic and independent of each other.

Every point on the boundary is optimal in the sense that is maximises the sum \( \mu_1 \cdot R_1 + \mu_2 \cdot R_2 \) for some pair \( \mu_1, \mu_2 \), with \( \mu_1 + \mu_2 = 1 \).

We first consider the maximum sum-rate: this is only ONE point on the boundary of the capacity region, which we obtain for \( \mu_1 = \mu_2 = 1/2 \).
Maximum Sum-Rate / Throughput:

- Upper limit for the sum-rate of all users:

\[
\sum_{j=1}^{J} R_j < E_{H_{\text{max}}} \left\{ \log_2 \left( 1 + \frac{P^* \cdot H_{\text{max}}}{N_0} \right) \right\} \approx \frac{1}{L} \sum_{k=1}^{L} \log_2 \left( 1 + \frac{P^*(k) \cdot h_{\text{max}}(k)}{N_0} \right)
\]

with \( H_{\text{max}} = \max_{j=1,\ldots,J} |H_j|^2 \), \( P^* = \left( \frac{1}{\lambda} - \frac{N_0}{H_{\text{max}}} \right)^+ \) and \((x)^+ = \max(x, 0)\).

Choice of \( \lambda \) such that \( P = E_{H_{\text{max}}} \{P^*\} \approx \frac{1}{L} \sum_{k=1}^{L} \left( \frac{1}{\lambda} - \frac{N_0}{h_{\text{max}}(k)} \right)^+ \)

- Well-known “waterfilling”-solution for power allocation!
- Sum-rate is achievable by scheduling only the user with the “best” channel and allocating them the power \( P^* \):
  - Theory suggests simple channel-adaptive time-division multiple-access scheme!
- Many users (large \( J \)): high chance to have large \( H_{\text{max}} \): Multiuser Diversity Gain!
- Use of constant power \( P \) instead of \( P^* \) causes rather small loss in sum rate for \( P/N_0 \) above 0dB.
- Simplicity of this solution is very appealing for practical use – however, it is a trap!
Maximisation of sum-rate: no “fairness” – but it’s unfair to blame theory!

- user with the best channel is served with high rate – even though they might not need it
- other users with “bad” channels are not served at all
- Throughput maximisation is often deemed “unfair”.
- To maximise sum-rate means to pick a particular point on the boundary of the capacity region.
- Popular solution mainly because it is simple.

- Our setting: data rates and delay constraints requested by the users
- What point on the boundary of the capacity region does solve our problem?
- How do we find this point and a practical scheme that can closely achieve it?
  - The average power used by the base station is just a consequence of the solution.
- For this we would actually need the whole boundary of the capacity region – not only one point.
- We also don’t specify average power: we rather specify requested average rates!
- No simple solution!
Ergodic Capacity Region of the Fading Gaussian Broadcast Channel and its Boundary Surface

General solution given in

Capacity region:

\[ C(\bar{P}) = \bigcup_{\mathcal{P} \in \mathcal{F}} C_{CD}(\mathcal{P}) \]

with \( \mathcal{F} \) the set of all power allocation policies that satisfy the average power constraint, i.e., \( E_n \left( \sum_{j=1}^{J} P_j(n) \right) = \bar{P} \) with the fading state \( n \doteq N_0 \cdot \{ \frac{1}{|h_1|^2}, \frac{1}{|h_2|^2}, \ldots, \frac{1}{|h_J|^2} \} \) and \( P_j(n) \) the power transmitted to user \( j \) in the fading state \( n \).

The set of \( J \)-dimensional “capacity-points” \( C_{CD}(\mathcal{P}) \) for any particular power allocation policy \( \mathcal{P} \) is defined as

\[ C_{CD}(\mathcal{P}) = \left\{ \mathbf{R} : R_j \leq E_n \left[ \log_2 \left( 1 + \frac{P_j(n)}{n_j + \sum_{i=1}^{J} P_i(n) \cdot 1(n_j > n_i)} \right) \right] \right\} \]

with the achievable-rate vector \( \mathbf{R} \doteq \{ R_1, R_2, \ldots, R_J \} \) and the indicator function \( 1[x] = 1 \) if \( x \) is true (“0” otherwise); the capacity result is in bits per channel-use (of any user).

The capacity region \( C(\bar{P}) \) can be shown to be convex, i.e., we can use Lagrangian optimisation to find power allocation policies that maximise this capacity (find points on the region’s boundary).
Ergodic Capacity Region of the Fading Gaussian Broadcast Channel / Boundary Surface

What we really want to know is what we can achieve in the best case with an average given power, i.e., we want to know the boundary surface of the capacity region.

As the region is convex we can write the optimisation problem as follows:

$$\max_{\mathbf{R}} \mathbf{\mu} \cdot \mathbf{R} \quad \text{subject to} \quad \mathbf{R} \in \text{the capacity region}, \quad \mathbf{\mu} = \{\mu_1, \ldots, \mu_J\} \quad \text{and} \quad \sum_{j=1}^{J} \mu_j = 1.$$ 

Convexity means that if two points $R_A$ and $R_B$ are in the capacity region, then any point on the straight line connecting $R_A$ and $R_B$ will also lie in the capacity region.

$\mu_j$: “rate-rewards” – picked such that we get a desired point on the boundary surface.

$\mu_j = 1/J$ corresponds to maximisation of sum-rate with its simple solution we discussed above.

Due to convexity the optimisation problem can be written in an equivalent form (not really obvious):

$$\max_{P(n)} E_n \left\{ J_0(P(n)) \right\} \quad \text{subject to} \quad E_n \left\{ \sum_{j=1}^{J} P_j(n) \right\} = \bar{P} \quad \text{with} \quad P(n) = \{P_1(n), \ldots, P_J(n)\}$$

with (the Lagrangian multiplier $\lambda$ must be chosen such that the average power constraint is met)

$$J_0(P(n)) = \sum_{j=1}^{J} \mu_j \cdot \ln \left( 1 + \frac{P_j(n)}{n_j + \sum_{i=1}^{J} P_i(n) \cdot 1(n_j > n_i)} \right) - \lambda \cdot \sum_{j=1}^{J} P_j(n)$$
Ergodic Capacity Region of the Fading Gaussian Broadcast Channel / Boundary Surface

Analytical solution for the best power allocation/scheduling policy is rather complicated – no further details given here (but numerical results below).

The best scheme will achieve a suitable operation point (chosen depending on the rate requests) on the boundary surface of the capacity region.

General optimal solution involves
- code superposition at the transmitter, adapted to current fading state
  (adaptation includes code and order of encoding)
- successive interference cancellation (SIC) at the receiver.
- multiple-access based on (non-orthogonal) Code Division (CD)!

Code superposition and SIC are not preferred options in practice:
- likely to violate delay constraints!
- coding scheme and order of cancellation need to be signalled → overhead!
- high complexity of coding and decoding

We try a suboptimal time-division (TD) access scheme
- performs close to the optimal CD solution for our problem
Ergodic Capacity Region with Time-Division Signaling Constraint / Boundary Surface

- Time division (TD) signaling: a maximum of one user is scheduled for each fading state (it may be that no user is scheduler at all): this rules out superposition coding and SIC

**TD Capacity Region:**

\[
C(\bar{P}) = \bigcup_{\mathcal{P} \in \mathcal{F}_{TD}} C_{TD}(\mathcal{P})
\]

\(\mathcal{F}_{TD}\) is the set of all time division power allocation policies \(\mathcal{P}\) that satisfy the power constraint, i.e.,

\[
E_n \left( \sum_{j=1}^{J} P_j(n) \right) = \bar{P}, \text{ with the fading state } n \doteq N_0 \cdot \{ \frac{1}{|h_1|^2}, \frac{1}{|h_2|^2}, \ldots, \frac{1}{|h_J|^2} \}
\]

- \(P_j(n)\) the power transmitted to user \(j\) in the fading state \(n\)

- as we use TD, we have \(P_j(n) > 0\) for at most one user \(j\).

The set of \(J\)-dimensional “capacity-points” \(C_{TD}(\mathcal{P})\) for any particular TD power allocation policy \(\mathcal{P}\) is defined as

\[
C_{TD}(\mathcal{P}) = \left\{ R : R_j \leq E_n \left[ \log_2 \left( 1 + \frac{P_j(n)}{n_j} \right) \right] \forall j \right\} \text{ in \ bits/channel-use (any user)}
\]

with the achievable-rate vector \(R \doteq \{ R_1, R_2, \ldots, R_J \}\).

- Capacity region is again convex so again we solve: \(\max_{R} \mu \cdot R \text{ subject to } R \in C_{TD}(\mathcal{P})\)
... Ergodic Capacity Region / Time-Division Signalling Constraint / Boundary Surface

- Power for user \( j \) in fading state \( n_j = N_0/|h_j|^2 \) (\( h_j \): channel coefficient, \( N_0 \): noise variance)

\[
P_j = \left[ \frac{\mu_j}{\lambda} - n_j \right]^+ = \max \left( \frac{\mu_j}{\lambda} - n_j, 0 \right)
\]

with \( \sum_j \mu_j = 1 \)

- \( \mu_j \): factors used to reach any desired point on the boundary of the capacity region.
  - rate-tradeoffs between users \( \Rightarrow \) maximum throughput iff \( \mu_j = \text{const} \ \forall \ j \)

- Only ONE scheduled user: \( j^* = \arg \max_j \mu_j \cdot \log_2 \left( 1 + \frac{P_j}{n_j} \right) \) and \( P_j = 0 \) for \( j \neq j^* \)

- The parameter \( \lambda \) is chosen such that \( \bar{P} = \mathbb{E}_n \left( P_{j^*} \right) \approx \frac{1}{L} \sum_{k=1}^{L} \max \left( \frac{\mu_{j^*(k)}}{\lambda} - n_{j^*(k)}, 0 \right) \)

\( \bar{P} \): average transmit power used by the base station.

- Achieved rates (\( L \) large): \( R_j = \mathbb{E}_n \left[ \log_2 \left( 1 + \frac{P_j}{n_j} \right) \right] \approx \frac{1}{L} \sum_{k=1}^{L} \log_2 \left( 1 + \frac{P_j(k)}{n_j(k)} \right) \ \forall j \)

\( \Rightarrow \) Simple solution that can be used as a scheduler in practice!

\( \Rightarrow \) Power \( \bar{P} \) needs to be adjusted such that rate requests are met.
**Code Division (CD) vs. Time Division (TD):**

**Numerical Comparison of Theoretical Limits for our Generic System:**

- Numerical results, with $SNR_j = \frac{P_j}{N_0}$
- SNR-values are without zero-powers (when user is not scheduled)
- SNR-values in brackets do contain zero-values

<table>
<thead>
<tr>
<th>User $j$</th>
<th>Rate Request $r_j$ in kbits/s</th>
<th>Channel Power Gain $G_j$ in dB</th>
<th>CD optimal $SNR_j$ in dB</th>
<th>TD $SNR_j$ in dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50.0</td>
<td>-27.0</td>
<td>17.21 (3.17)</td>
<td>17.71 (3.05)</td>
</tr>
<tr>
<td>2</td>
<td>100.0</td>
<td>-26.0</td>
<td>17.57 (5.90)</td>
<td>18.00 (5.77)</td>
</tr>
<tr>
<td>3</td>
<td>150.0</td>
<td>-25.0</td>
<td>17.53 (7.15)</td>
<td>17.90 (7.05)</td>
</tr>
<tr>
<td>4</td>
<td>200.0</td>
<td>-24.0</td>
<td>17.31 (7.78)</td>
<td>17.66 (7.71)</td>
</tr>
<tr>
<td>5</td>
<td>250.0</td>
<td>-23.0</td>
<td>16.96 (8.06)</td>
<td>17.32 (8.05)</td>
</tr>
<tr>
<td>6</td>
<td>300.0</td>
<td>-22.0</td>
<td>16.50 (8.15)</td>
<td>16.93 (8.19)</td>
</tr>
<tr>
<td>7</td>
<td>350.0</td>
<td>-21.0</td>
<td>15.93 (8.07)</td>
<td>16.48 (8.16)</td>
</tr>
<tr>
<td>8</td>
<td>400.0</td>
<td>-20.0</td>
<td>15.22 (7.87)</td>
<td>16.02 (8.09)</td>
</tr>
</tbody>
</table>

|        | Average                       | 16.30                          | 16.31                    |

- Note: *more* than one user may be scheduled in CD!

- Little difference between CD and TD: much simpler TD system will do a good job!
Access rates for TD and CD fairly similar.

Access rate > 100% for CD as more than one user can be scheduler at a time → does not happen very often though!

Average power for CD and TD differ very little.

Simple TD system will do for us!
Power Allocation for TD Ergodic Capacity as a Scheduling Scheme ("TD Waterfilling")

- What happens if TD power allocation and single-user selection are used in our generic system?
- Ignored by Ergodic Theory: delay constraints...

![Graph showing TD Waterfilling results]

- \( SNR_j = 16.0 \text{ dB}, G_j = -20.0 \text{ dB} \)
- \( \tau_j = 1000 \) (j=8)
- \( R_j(k) = R_j(k-i) \frac{\tau_j-1}{\tau_j} \sum_{i=0}^{\tau_j-1} R_j(k-i) \)

with \( k = n \cdot \tau_j - 1 \),
\( n = 1, 2, \ldots \), for all users \( j \)
\( \tau_j \) are the delay limits
SNR is \( P/N_0 \) without zero-power values.

- Low power and long-term average rates achieved
- Delay constraints

\[ \bar{R}_j(n \cdot \tau_j - 1) \geq r_j \]
are heavily violated!
... Power Allocation for TD Ergodic Capacity as a Scheduling Scheme ("TD Waterfilling")

- Effect of TD Waterfilling on the channel coefficients “used” with non-zero transmit power:

Channel coefficients are only used when they are large!

Water-filling enforces a hard threshold below which the channel is never used.
- strictly limited peak power!
- This saves power
  ... but at the same time
- causes serious delay problems!
- bad option for multi-media applications with delay constraints
The other extreme: Round Robin Scheduler with Channel Inversion:

- **Round Robin**: fixed pattern which schedules users independently of their current channel quality.
- **Channel inversion**: when user is scheduled, requested rate is enforced in every slot by power.
- No advantage taken of possible averaging within the delay limits.

\[
\tilde{R}_j(k) = \frac{1}{\tau_j} \sum_{i=0}^{\tau_j-1} R_j(k - i)
\]

for all users \( j = 1, 2, \ldots, J \)

\( \tau_j \) are the delay limits

**SNR**: \( P/N_0 \) without zero-power values.

- **Huge power**: 15dB more than TD waterfilling. Even worse:
- peak power unlimited; must be clipped in practice!
- Delay constraints are met:

\[
\tilde{R}_j(n \cdot \tau_j - 1) \geq \tau_j
\]
Round Robin with Channel Inversion

Effect of Round Robin scheduling with channel inversion on the channel coefficients used:

Channel coefficients are always used, not only when they are large!

Channel inversion causes huge (theoretically unlimited!) peak power.

Simple scheme that strictly meets delay constraints but terrible power efficiency!
Modified TD Waterfilling

Modifications:

- **Increase average rate requests** such that minimum of the averaged rate

\[
\tilde{R}_j(k) = \frac{1}{\tau_j} \sum_{i=0}^{\tau_j-1} R_j(k - i)
\]

does not drop below the actual rate request ("min-rate capacity idea").

Problem: \( \tilde{R}_j(k) \) is a random variable ...

- **Stop scheduling** a user, once the **average rate achieved in a fraction of a delay window already meets the original (non-increased) rate request**

  ➞ even though the end of the delay window has not yet been reached and the user might be getting a good channel
  ➞ it would not be possible to make use of the extra "rate" for delay-limited applications, as new data "bits" are not yet available
... Modified TD Waterfilling

![Graph showing rate-average in kbits/s (rect. window of length $\tau$)]

- **SNR$_j = 16.8$ dB, $G_j = -20.0$ dB, $\tau_j = 1000$ (j=8)**
- **SNR$_j = 17.3$ dB, $G_j = -21.0$ dB, $\tau_j = 864$ (j=7)**
- **SNR$_j = 17.9$ dB, $G_j = -22.0$ dB, $\tau_j = 729$ (j=6)**
- **SNR$_j = 18.4$ dB, $G_j = -23.0$ dB, $\tau_j = 593$ (j=5)**
- **SNR$_j = 18.8$ dB, $G_j = -24.0$ dB, $\tau_j = 457$ (j=4)**
- **SNR$_j = 19.4$ dB, $G_j = -25.0$ dB, $\tau_j = 321$ (j=3)**
- **SNR$_j = 20.1$ dB, $G_j = -26.0$ dB, $\tau_j = 186$ (j=2)**
- **SNR$_j = 22.3$ dB, $G_j = -27.0$ dB, $\tau_j = 50$ (j=1)**

- **Average transmit SNR across all users: 17.4 dB**

- **Characteristic similar to waterfilling**
- **Strictly limited peak power!**
- **Low-delay application must still be served for channels of low quality**
- **$\Rightarrow$ power/delay-tradeoff!**

Only 1dB more average power than theoretical limit!

Low delay application still not served satisfactorily:
- large rate fluctuations – only $\approx 4\%$ access rate
- sudden drops below rate request
- delay limits not used explicitly by the scheduler
“Proportional Fair Scheduling” (PFS) – a “Best-Effort” Concept


- Used in wireless standards such CDMA 2000 (3rd generation mobile radio standard in the US)
- Average $T_j(k)$ of the data rate of every user $j$ is computed in an exponential decaying window; $T_j(k)$ is re-computed every time-step $k$:
  \[
  T_j(k) = \begin{cases} 
  (1 - \frac{1}{t_0})T_j(k-1) + \frac{1}{t_0}R_j(k-1), & \text{if user } j \text{ scheduled in slot } k-1 \\
  (1 - \frac{1}{t_0})T_j(k-1), & \text{otherwise}
  \end{cases}
  \]
  
  “Window-factor” $t_0$: small value for delay-critical applications

- $R_j(k)$: Rate user $j$ can get in time slot $k$; rate depends on current channel-coefficient $H_j(k)$!
  \[
  R_j(k) = \log_2 \left( 1 + \frac{P |H_j(k)|^2}{N_0} \right) \quad (P: \text{fixed power of the base station}, \ N_0: \text{noise variance})
  \]

- Scheduling: user $j^*$ will be served in time slot $k$, when $\frac{R_j(k)}{T_j(k)}$ is maximised for $j = j^*$

- $T_j(k)$ can be seen as a “soft” approximation of our average-rate definition $\bar{R}_j(k)$ (which is a rectangular window of size $\tau_j$ for that we apply hard rate requests $r_j$).

- As $\bar{R}_j(k)$ may “suddenly” drop to “zero”, it will cause difficulties when used to weight the metric!

- PFS as such is best-effort service, as users have to be happy with the rate they get: purely dependent on the channel and the fixed power of the base station.
Modified Proportional Fair Scheduling

Modifications:

- Optimised user-specific constant transmission powers $P_j$ used by the base station.
- Users who have already achieved their requested average rates in the current delay window are no longer scheduled.
- User-specific window-factors are introduced that depend on their delay-constraints!
Only 1.5dB more average power than theoretical limit!

Strictly limited peak power!

Low delay application now served “satisfactorily”.

Characteristic no longer like classic waterfilling: no sharp cut-off!

Very high power is still avoided!

Implements power/delay-tradeoff!
Some Open Problems

- Ad-Hoc solution above used fixed powers optimised in advance: not realistic in practice as channels will change e.g. due to fading.
- We could interpret the problem as a stochastic control problem...
- Key problem in the general setting: channel coefficients not known in advance for the time slots to follow: we only know their short-term statistics.
- In so-called multicarrier systems (WLAN, OFDM), we have a set of parallel channels at each time instant, i.e., we know a set of channel coefficients and we want to use them efficiently. This setting may be “easier” to control.
- In practice, “delay” is critical for quality-of-service, and “pragmatic” solutions as well as good theory are required, so we can go beyond “ad-hoc solutions”.

Wireless Communications 2
Cross-Layer Design 144
Cross-Layer Summary

- Take the time to wait for better channel coefficients: exploit Multiuser Diversity!
- Applications with relaxed delay constraints: no problem, no matter what the rate requests are.
- Applications with hard delay constraints:
  - Cross-Layer Scheduling required: exploits channel quality and Quality-of-Service requirements
  - Suitable power-delay tradeoff needs to be found!
  - Still large Multiuser Diversity Gains possible!
- Knowledge of channel coefficients at the transmitter can be difficult to realise.
- Flexible variable-rate channel coding and modulation schemes that realise the promised gains still have to be found and implemented efficiently.
- Combination with MIMO and cooperative communication is an open problem.
- Good schedulers are required, particularly challenging for:
  - inter-cell interference reduction \textit{without} any “central” coordination
  - distributed scheduling and power allocation in ad-hoc networks