ELT-43008 DIGITAL COMMUNICATION

Lecture material, Spring 2020

Mikko Valkama & Markku Renfors
mikko.valkama@tuni.fi

Updated: January 3, 2020

ELT-43008 DIGITAL COMMUNICATION, 5 cr
Spring 2020

All the basic information regarding the schedules (lectures, exercises, etc.) are always available in POP, www.tut.fi/pop

Follow also the course web site http://www.cs.tut.fi/courses/ELT-43007/ where the course materials (lecture notes, exercise questions, project works etc.) are delivered.

Objectives and content

– This course focuses on the digital information transmission and signal processing techniques utilized in all widely-used and emerging digital communication systems, such as 2G-5G mobile cellular networks, WLAN/ WiFi networks, digital TV, and DSL systems (to name a few). These techniques include error control coding, digital modulation, filtering, channel equalization, and demodulation/detection.

– The essential elements include the following:
  ✓ Information theoretic foundation of electrical communication; capacity laws under noise, band-limitation and fading
  ✓ Baseband and carrier modulation digital communications; baseband PAM, I/Q modulated PSK/QAM, digital frequency modulation
  ✓ Performance of digital communication systems; spectral and energy efficiency, symbol/bit/packet errors and their dependency on noise and fading
  ✓ Detection theory and inter-symbol interference (ISI) mitigation in receivers; ML and MAP detectors, channel equalizers, channel estimation methods, Viterbi algorithm and sequence detection
  ✓ Error control coding in digital communication systems; concepts of redundancy and coding gain, block codes, convolutional codes, Turbo codes, polar codes, interleaving and puncturing, retransmission principles
Requirements and implementation

- Personal project work (computer simulation based studies related to the course contents); mandatory for all
- For grades higher than 1: passing the exam or two mid-term exams is also mandatory
- With grade 1, the course can also be passed by earning sufficient amount of bonus points (more details about that soon).
- The course includes lectures, classroom exercises, Matlab exercises, project works, and exam (though the exam is not necessarily required to pass the course, see more information in bonus points section...).

Lectures

- WEDNESDAYS at 09:15-12 in room TC103
  - 1st lecture on 08.01.2020 (week #2)
- Lecturer and course responsible: Mikko Valkama

Classroom exercises

- 5 separate exercises per period (10 overall), each 2h or so
- Starting on week #4 (period 3), and week #11 (period 4)
- Time and place:
  - MONDAYS 15-17, room TC161
- Exercise materials can be found at the course web site
- Attending to the exercises AND by doing the associated homework tasks you can earn bonus points to the exam or even pass the course without an exam (see the information below)!
- In order to earn the bonus points from the exercises, one must do a small homework problem BEFORE each exercise session and return it to the exercise assistant in the beginning of the session.

Matlab exercises

- Matlab is a very versatile numerical computing environment and a programming language.
- It is a very important research and development tool and environment in the area of communications, as well as in many other fields of science and technology.
- Matlab will be used also many of the later courses as well as most likely in the professional engineering work and career
- 5 different exercise sets, each ~1.5-2h, organized in room SM201 at 15-17 on the following TUESDAYS

Course materials

- Other good books are, for example
- Lecture notes and exercise materials, as well as model solutions, are all available at the course website

Language

- Course is given in English, however there is also consultation and guidance available in Finnish for Finnish students based on needs.
**Bonus points :^/**

- By attending to the exercises AND by doing homework tasks you can earn 1-3 bonus points for the exam 😊 (the maximum number of points in the exam is 30, and the grading will be based on normal system with roughly 3 point difference per grade).
- If you earn enough points, you can even pass the course, with grade 1, without taking the exam.
- In total there are 29 learning sessions in the course: 14 lectures, 10 classroom exercises (attendance+homework), and 5 Matlab exercises.
- Algorithm (valid over the 3 next exams):

<table>
<thead>
<tr>
<th>Attendance at learning sessions* (max. 29)</th>
<th>Earned bonus points</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
</tr>
</tbody>
</table>

* learning sessions include:
  - Lectures (1 point for attending one 3h lecture session)
    ° i.e., max 14 attendances from the lectures
  - Classroom exercises (a half of the attendance through attending, and another half based on the homework solution)
    ° i.e., max 10 attendances from the classroom exercises
  - Matlab exercises (1 point for attending one 2h session)
    ° i.e., max 5 attendances from the Matlab exercises

- For example, if one attends 8 lecture sessions, 7 classroom exercises and 5 Matlab exercises, and returns 6 reasonably good homework solutions in the beginning of each classroom exercise, then the total number of attendances is 8+7*0.5+5+6*0.5 = 19.5 ≈ 20. This means 2 extra points for the exams.

**Passing the course without taking an exam**

- To pass the course without an exam (or mid-term exams), one must earn 3 bonus points (i.e. get minimum of 25 attendances from the learning sessions) and complete the personal project work.
- If the above requirements are fulfilled and the student decides not to take the exam, or the student fails in passing the exam, the minimum grade of 1 ("one") is automatically given.
- you can always anyway go to the exam, or mid-term exams, for sure.
- Midterm exams (alternative to full exam) will be organized as follows:
  - 1.midterm exam: MON March 2nd, 2020, at 15:00-17, in TC161
  - 2.midterm exam: THU April 23rd, 2020, at 16:00-18, in TC161
    (these show as “exercise” sessions in POP)

**Prerequisite knowledge**

- ELT-41307 / ELT-41308 Communication Theory
  ° … or similar knowledge
  ° In summary: General understanding of linear systems and filters, spectrum concept, basic modulation techniques, and basics of probability calculus

**Contact information**

- Taneli Riihonen, taneli.riihonen@tuni.fi, TD409; main lecturer
- Mikko Valkama, mikko.valkama@tuni.fi, TD428; course responsible
# PLANNED CONTENTS vs. LECTURE SESSIONS

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topics</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction; recap on digital comms basics, information theory (basics + fading channels)</td>
<td>1-60</td>
</tr>
<tr>
<td>2</td>
<td>Baseband PAM and I/Q modulated PSK/QAM</td>
<td>61-135</td>
</tr>
<tr>
<td>3-4</td>
<td>Bit errors, symbol errors, packet errors and their dependency on SNR and fading</td>
<td>136-177</td>
</tr>
<tr>
<td>4</td>
<td>Modulation spectral and energy efficiency</td>
<td>178-193</td>
</tr>
<tr>
<td>5</td>
<td>Digital frequency modulation</td>
<td>194-218</td>
</tr>
<tr>
<td>6</td>
<td>Maximum likelihood (ML) and maximum a posteriori (MAP) detection frameworks</td>
<td>219-250</td>
</tr>
<tr>
<td>7</td>
<td>Signal space concepts &amp; optimum receivers, concept of matched filtering</td>
<td>251-275</td>
</tr>
<tr>
<td></td>
<td>… exam week … 1st midterm exam (on first week of Period 4)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Dealing with intersymbol interference (ISI) in receivers 1; ML sequence detection (MLSD) and Viterbi algorithm</td>
<td>276-301</td>
</tr>
<tr>
<td>9</td>
<td>Dealing with ISI in RX 2; channel equalizers</td>
<td>302-334</td>
</tr>
<tr>
<td>10</td>
<td>Dealing with ISI in RX 3; channel estimation, and adaptive filtering based estimators and equalizers</td>
<td>335-367</td>
</tr>
<tr>
<td>11-12</td>
<td>Error control coding 1; coding fundamentals, block codes, hard and soft decoding</td>
<td>368-402</td>
</tr>
<tr>
<td>12</td>
<td>Error control coding 2; convolutional codes, interleaving and concatenated codes, (H)ARQ</td>
<td>403-426</td>
</tr>
<tr>
<td>13</td>
<td>Error control coding 3; Turbo codes, LDPC codes, polar codes</td>
<td>427-451</td>
</tr>
<tr>
<td>14</td>
<td>Error control coding 4; trellis coding principle, scrambling in short; course summary+wrap-up</td>
<td>452-466</td>
</tr>
<tr>
<td></td>
<td>… 2nd midterm exam (on last week of Period 4)</td>
<td></td>
</tr>
</tbody>
</table>

---

This course – the big picture

![Diagram](image)

(A): Uncoded system  
(B): Coded system, SD  
(C): Coded system, HD

---

![Graph](image)

**Ergodic Capacity in Rayleigh Fading**

- **10 MHz Bandwidth, true**
- **10 MHz Bandwidth, upper bound**

---

![Graph](image)

**E_{s}/N_{0}(dB)**

**P_{f}(e)**
1. SHORT INTRODUCTION

Digital communications, sometimes called also digital transmission, has become by far the dominant communication technology during the last 10-15 years.

All new communications systems under development are based on digital communication techniques and digital modulation methods, e.g.

- Wireline telephone networks, for example
  - xDSL: Digital Subscriber Line techniques (HDSL, ADSL, VDSL, …) enable from tens up to hundreds of Mbps connection rates

- Audio and TV broadcasting:
  - DAB (Digital Audio Broadcasting)
  - DVB-T/T2/C/S/H/NGH (Terrestrial, Cable, Satellite, Handheld)
  - Cable modems

- Mobile cellular networks
  - 2nd generation (2G): GSM, GPRS, EDGE
  - 3rd generation (3G): WCDMA, UMTS, IMT-2000, HSPA
  - Beyond 3G, 3.9G: Dual Carrier HSPA, “4G” (LTE)
  - 4th generation (4G): LTE-Advanced, LTE-Advanced Pro
  - 5th generation (5G): called 5G New Radio (NR), currently under intensive R&D and standardization phase…

- WLAN/WiFi networks, offering wireless local area connectivity with 0.1 / 2 / 10 / 20 / 50 / 100 / 1000 / 1000+ Mbps rates
  - IEEE 802.11a/b/g/n/ac/ax/ay
  - IEEE 802.11ah is longer-rate low-bitrate technology for sensor networks and IoT

- Other ISM-band techniques for long(er)-range wide-area IoT/machine comms
  - e.g. LORA and Sigfox

- ISM-band techniques for short range wireless connectivity, e.g.
  - Bluetooth: low-power low rate applications
  - IEEE 802.11ad/ay/WiGig: very high throughput applications

Important thing is to understand the nature of different signals, digital vs. analog, in different interfaces in the above diagram

The original message can be analog (e.g. speech) or digital (e.g. a digital file)

- independently of that, the message can be communicated using digital communication techniques
- for analog messages, proper message digitalization is needed (e.g. voice or video codec)
Core elements of a digital communication system, cont’d

On the other hand, since the transmission media (cable, radio path, etc) are anyway analog and continuous-time in nature, the final transmitted and received digital communication waveforms are also analog and continuous

- but they carry the digital data (bits) inside

The analog waveforms are designed and optimized such that the key physical limitations related to all comms, that is bandwidth and power/energy, are taken properly into account.

The overall chain from source to destination/sink includes always:

Source coding/decoding: To (digitize and) reduce the bit rate of the digital message signal by removing any existing redundancy (compression; as efficient digital representation of the message as possible)

One of the most essential results of information theory is that the source coding and the channel coding can be performed independently of each other.

Channel coding/decoding: Introduce controlled redundancy to reduce the effects of errors in the transmission channel (error control coding)

Almost in any reasonable channel, an arbitrary small bit error probability can be achieved by adding controlled redundancy in the transmitted signal.

Modulation/demodulation: Converting a digital coded signal into an analog waveform and vice versa. Can be further divided into data and carrier modulation/demodulation.

Channel: Here harmful noise, interference and distortion are always encountered.

Fundamental parameters and characteristics

The external operation of the previously defined digital communication system can be described with the following core parameters:

- Transmission rate (bit rate; bits/s, bps)
- Error probability (e.g. $10^{-1}$, $10^{-4}$, $10^{-10}$, …)
- Latency/delay, caused by signal propagation as well as the signal processing stages in the devices (μs, ms, …)

From system’s external operation point of view only, these parameters have significance

- The technical details regarding how, e.g., a certain transmission rate is achieved, is not important for the end user
  => more degrees of freedom / flexibility compared to analog comms, even if the original message would be analog (such as speech)

Thus, the internal operation of the system (channel coding, modulation method, etc) can be optimized with respect to the properties of the used transmission medium in terms of the key resources

- Bandwidth (Hz)
- Transmission power/energy (W, J)

so that the requirements for the external operation are fulfilled! The concept of spectral efficiency is relating the achievable bit rate and the used bandwidth as

- spectral efficiency = bit rate / bandwidth [bits/s/Hz]

Notice that also the requirements for the external operation parameters are strongly application specific

- E.g., intelligible speech vs. file transfer
- (.. examples ?)
Fundamental parameters and characteristics

In communications engineering, when referring to the digital communication system, we commonly mean the part of the previous overall chain, in which the interfaces are the input of the channel coding and the output of the channel decoding.

Therefore, source coding and decoding are not included, strictly speaking, in the actual communication system:

- the source coding methods (compression of information, speech/audio/video coding, etc.) are more related to the type and nature of the information source – not to the fundamental communication system
- thus, at the input of the communication system (input of the channel coder), we simply assume an efficient digital representation of the source information
  - essentially a bit stream without redundancy
- Based on this definition, we are able to develop, design and analyze the key processing solutions of the communication systems independent of the exact nature of the transmitted information (independent of what type of information do the input bits exactly represent)
- ..but on the other hand, one should certainly always keep in mind that different applications have different requirements for the key parameters (bit rate, error probability, latency)! (like discussed on the previous slide)

2. INFORMATION THEORETIC FOUNDATION

In this part we address the very basics of such concepts as information, entropy, mutual information and channel capacity.

In the general electrical and electro-magnetic communications context, by using these concepts, it is possible to determine the largest possible information transmission rate through a given channel:

- This is called the channel capacity
- Formulated by the so called Shannon-Hartley law

Even though it is usually not possible to achieve the channel capacity in any practical system, it is an important reference point when evaluating the performance of practical systems.

In fact, the Shannon-Hartley law is one of the most important and fundamental laws of nature in the field of electrical and electro-magnetic communications, and it is quite useful also in practical engineering work.

This part of the notes is largely a recap from the ELT-41307 Comm Theory course but also includes new results regarding, e.g., the channel capacity under time and frequency selective fading

- source coding related aspects are not reproduced here, mostly focusing on channel capacity

Sources and references:

Information theory builds on probabilistic thinking, thus we first shortly review some of the basic statistics of discrete and continuous random variables below.

This is actually useful also in many of our other topics later in the course (error probability analysis, detection methods, etc).

**Discrete random variables**
- \( X \), discrete sample space \( \Omega_X = \{x_1, x_2, \ldots, x_K\} \);
- \( Y \), discrete sample space \( \Omega_Y = \{y_1, y_2, \ldots, y_L\} \), …
- probabilities \( p_X(x_k), p_Y(y_l), \ldots \)
- joint probabilities \( p_{X,Y}(x_k, y_l) \)
- conditional probabilities \( p_{X|Y}(x_k | y_l) = \frac{p_{X,Y}(x_k, y_l)}{p_Y(y_l)} \)
- statistical expected values or moments \( E[X] = \sum_{x \in \Omega_X} x p_X(x), E[X^n] = \sum_{x \in \Omega_X} x^n p_X(x), E[g(X)] = \sum_{x \in \Omega_X} g(x) p_X(x) \)

**Continuous random variables**
- \( X \), continuous sample space \( \Omega_X \);
- \( Y \), continuous sample space \( \Omega_Y \), …
- probability densities or distributions \( f_X(x), f_Y(y), \ldots \)
- joint distribution \( f_{X,Y}(x, y) \)
- conditional distribution \( f_{X|Y}(x | y) = \frac{f_{X,Y}(x, y)}{f_Y(y)} \)
- statistical expected values or moments \( E[X] = \int_{\Omega_X} xf_X(x)dx, E[X^n] = \int_{\Omega_X} x^n f_X(x)dx, E[g(X)] = \int_{\Omega_X} g(x)f_X(x)dx \)

**Definition of Information**

In information theory, the message signal is modeled as a random process.

We begin first by considering observations of a single random variable:
- Each observation gives a certain amount of information.
- But rare observations give more information than the usual ones (intuition?).

**Example:** The statement "The sun rose this morning" gives very little information (very high probability).

The statement "Turku was destroyed this morning by an earthquake" gives a lot of information (low probability).

**Definition:**
Observing a random variable \( X \) that takes its values from the discrete set \( \Omega_X = \{a_1, a_2, \ldots, a_K\} \), the information obtained from a particular observation \( a_m \) is defined as
\[
h(a_m) = \log_2 \left( \frac{1}{p_X(a_m)} \right) = - \log_2 \left( p_X(a_m) \right)
\]
where \( p_X(a_m) \) is the probability of the outcome \( a_m \).
Definition of Information - Interpretations

Based on the definition $h(a_m) = -\log_2(p_X(a_m))$, it is easy to see that

$0 \leq h(a_m) \leq \infty$

- for a rare event, the probability $p_X(a_m)$ is small and the information is large
- for a usual event, the probability $p_X(a_m) \approx 1$ and the information is small (close to zero)

Why logarithm? With two independent random variables $X$ and $Y$, for which the discrete sample spaces read $\Omega_X = \{a_1, a_2, \ldots, a_K\}$ and $\Omega_Y = \{b_1, b_2, \ldots, b_N\}$, the information obtained from observing a joint event $a_m, b_n$ reads

$h(a_m, b_n) = -\log_2(p_{X,Y}(a_m, b_n)) = -\log_2(p_X(a_m)) - \log_2(p_Y(b_n))$

$= h(a_m) + h(b_n)$

- above holds as long as the events are independent, and thus $p_{X,Y}(x_k, y_l) = p_X(x_k)p_Y(y_l)$
- Thus in case of independent events, the information is additive, which makes sense intuitively, right?
- $\log(.)$ is also a monotonic function, so it only normalizes or scales the information measure, in principle

Using base 2 logarithm => the unit of information is bit (becomes clear soon)

Sometimes also other units are used, corresponding to the different log bases as:

- base 10 logarithm => the unit is dit
- base $e$ (natural) logarithm => the unit is nat or Hartley

Entropy

The average information of a random variable is

$$H(X) = E[-\log_2(p_X(X))] = -\sum_{x \in \Omega_X} p_X(x) \log_2(p_X(x))$$

This is called the entropy.

Entropy has the following interpretations:

- Average information obtained from an observation
- Average uncertainty about $X$ before the observation (thus notice the connection to the concept of entropy in physics)

Example: Binary random variable $X$, $\Omega_X = \{0, 1\}$, $p_X(1) = q$

- The entropy is now

$$H(X) = -q \log_2(q) - (1-q) \log_2(1-q)$$

- Illustrated below, for varying $q$
- The maximum of entropy is 1 bit and is obtained when $q = 1/2$.
- The entropy becomes zero for $q = 0$ or $q = 1$

- do these results conform with your intuition?
Entropy, some general properties

In general, it is easy to show for a discrete random variable $X$ that

$$0 \leq H(X) \leq \log_2(K)$$

where $K$ refers to the sample size of $X$ (i.e., the number of different possible outcomes).

The lower limit (zero) is reached when one of the outcomes has probability 1 and others have probability zero

- thus, one of the cases is a "certain" case (happens with probability 1)
- in the previous binary example, this corresponds to cases with $q = 0$ and $q = 1$

The upper limit, in turn, is achieved when all different outcomes are equally likely

- in this case, $p_X(x) = 1/K \quad \forall x \in \Omega_X$
- and the upper limit follows directly by substituting these probabilities to the definition of entropy

In general, it should be noted that the concept of entropy in information theory is always only related to the probabilities (see the definition on previous slide), nothing else.

Example: throwing a fair dice

- six possible outcomes (normal dice), each outcome has a probability $1/6$
- entropy is thus

$$H(X) = \log_2(6) \approx 2.585 \text{ bits}$$

Example: entropy of English text

Memoryless model of English text results to the following probabilities or rates of occurrence of different alphabets:

<table>
<thead>
<tr>
<th>Source Symbol $a_i$</th>
<th>Probability $p_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>0.186</td>
</tr>
<tr>
<td>A</td>
<td>0.064</td>
</tr>
<tr>
<td>B</td>
<td>0.013</td>
</tr>
<tr>
<td>C</td>
<td>0.022</td>
</tr>
<tr>
<td>D</td>
<td>0.032</td>
</tr>
<tr>
<td>E</td>
<td>0.103</td>
</tr>
<tr>
<td>F</td>
<td>0.021</td>
</tr>
<tr>
<td>G</td>
<td>0.015</td>
</tr>
<tr>
<td>H</td>
<td>0.047</td>
</tr>
<tr>
<td>I</td>
<td>0.058</td>
</tr>
<tr>
<td>J</td>
<td>0.001</td>
</tr>
<tr>
<td>K</td>
<td>0.005</td>
</tr>
<tr>
<td>L</td>
<td>0.032</td>
</tr>
<tr>
<td>M</td>
<td>0.020</td>
</tr>
<tr>
<td>N</td>
<td>0.057</td>
</tr>
<tr>
<td>O</td>
<td>0.063</td>
</tr>
<tr>
<td>P</td>
<td>0.015</td>
</tr>
<tr>
<td>Q</td>
<td>0.001</td>
</tr>
<tr>
<td>R</td>
<td>0.048</td>
</tr>
<tr>
<td>S</td>
<td>0.051</td>
</tr>
<tr>
<td>T</td>
<td>0.080</td>
</tr>
<tr>
<td>U</td>
<td>0.023</td>
</tr>
<tr>
<td>V</td>
<td>0.008</td>
</tr>
<tr>
<td>W</td>
<td>0.017</td>
</tr>
<tr>
<td>X</td>
<td>0.001</td>
</tr>
<tr>
<td>Y</td>
<td>0.016</td>
</tr>
<tr>
<td>Z</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Entropy:

$$H(X) = -\sum_i p_i \log_2 p_i = 4.03 \text{ bits} \quad (< \log_2(27) \approx 4.755)$$

=> thus there is some structure in text, even when interpreted as a memoryless sequence of alphabets

=> leads to the source coding interpretations, see ELT-41307 or ELT-41308 course notes
**CHANNEL CAPACITY, BASICS**

The concept of channel capacity is addressing the fundamental physical limits for the amount of information, per unit time or channel use, which in theory can be communicated error-free over a given channel.

This is the most fundamental information theoretical aspect of all communication theory, by far.

- **SOURCE** \( X \) \( \xrightarrow{H(X)} \) **CHANNEL** \( Y \) \( \xrightarrow{H(Y)} \) **SINK**

The source is here modeled as a sequence of independent observations of a source random variable \( X \).

The observation at receiver consists of another random variable \( Y \).

Based on the earlier discussions, the average amount of information at source, per observation, is \( H(X) \).

The questions in the following are then:

- How much of this information can pass through the channel?
- And in general: What’s the maximum information transfer rate that a given channel can support?
- And how do these eventually map to physical quantities, in particular transmission bandwidth and received signal SNR?

These are addressed in the following, by adopting the concepts of conditional entropy and mutual information.

---

**Channel capacity concept and conditional entropy**

In the following, the capacity analysis will build on the following thinking:

- average information (entropy) at source is \( H(X) \), average information (entropy) at receiver is \( H(Y) \)
- on the other hand, average uncertainty about \( X \) is, in general \( H(X) \), while it is \( H(X | Y) \) after observing \( Y \)

The quantity \( H(X | Y) \) is called average conditional entropy of \( X \) given \( Y \) and will be more formally defined, in terms of math and probabilities, soon.

The interpretations of this conditional entropy are that it represents

- the average information obtained from observing \( X \) given that the value of \( Y \) is known
- the average uncertainty about the value of \( X \) given that the value of \( Y \) is known (i.e., after \( Y \) has been observed)

In general, the conditional entropy measures the statistical dependence between \( X \) and \( Y \) in information theoretic sense, and allows us to quantify the information that \( Y \) contains about \( X \):

- if \( X \) and \( Y \) are statistically independent, \( H(X | Y) = H(X) \)
- in this case, the channel does not convey, any information, right?
The difference in the uncertainty about $X$ before and after $Y$ has been observed, $H(X) - H(X | Y)$, is thus the information that the channel conveys about $X$, that is, the information about $X$ contained in $Y$. This is called **average mutual information** between $X$ and $Y$, written here as

$$I(X, Y) = H(X) - H(X | Y)$$

$$= H(Y) - H(Y | X)$$

- the latter expression is seeking to emphasize that the mutual information between $X$ and $Y$ is symmetric, and is also often easier to adopt in practical calculations
- mutual information describes the information that $Y$ contains about $X$, and vice versa

Mutual information $I(X, Y)$ is a measure of statistical dependence between $X$ and $Y$, and of the amount of information conveyed by the channel

- if $X$ and $Y$ are fully dependent, $H(Y | X) = H(X | Y) = 0$
- if $X$ and $Y$ are statistically independent, $H(Y | X) = H(Y)$ and $H(X | Y) = H(X)$
- (why? . . . think of the uncertainty interpretations)

In the latter case of independent $X$ and $Y$, it then obviously follows that

$$I(X, Y) = H(X) - H(X | Y) = H(X) - H(X) = 0$$

The actual channel capacity, per channel use, is obtained when the mutual information is maximized over the input distribution (maximum mutual information between $X$ and $Y$):

$$C_S = \max_{p_X(y)} I(X, Y) \quad \text{[bits per channel use]}$$

Finally, the actual capacity $C$ in bits per second is obtained by multiplying the above value by the amount of independent channel uses per second $s$ (over which the information can be added together, due to the independence assumption), written as

$$C = sC_S \quad \text{[bits per second]}$$

In the continuation, this fundamental framework is developed further, in different cases regarding the nature of the input and output distributions, reflecting different levels of abstraction in modeling the physical systems.

Before that, we state the Shannon capacity theorem, next page, which formally establishes the meaning of the channel capacity concept as the ultimate limit of the amount of information that can be communicated, per unit time.
Shannon capacity theorem, in general form

Suppose that

- the rate of the source is \( R = rH(X) \) [bits/s]
- the channel capacity is \( C = sC_s \) [bits/s]
- \( R < C \)

Then there exists a combination of source coding, channel coding and modulation methods, such that it is in principle possible to communicate the information, error free, from source to the destination.

This means that as long as the source bit rate is lower than the channel capacity, all bits can be, in theory, communicated error free (with zero bit error probability).

Notice that the theorem is abstract, that is, it does not say anything that how this can be realized in practice.

Also, in practice, seeking to realize zero bit error probability is not exactly feasible, since that would commonly impose an overly large processing latency and complexity

- thus, an important part of practical engineering work is to develop coding and modulation methods whose performance is close to the Shannon bound, with feasible processing latency and complexity

Furthermore, this theorem also states another fundamental fact, when turned the other way around:

- if the rate of the source is larger than the capacity of the channel, then it is not possible to communicate all the bits error free, even in theory!

Channel capacity concept, cont’d

In the following, we’ll first look into this in more details in the below example cases:

- capacity per channel use, assuming
  - case #1: discrete-valued input and discrete-valued output
  - case #2: discrete-valued input and continuous-valued output
  - case #3: continuous-valued input and continuous-valued output

- capacity of bandlimited continuous-time / waveform channels under noise
  - celebrated Shannon-Hartley law for maximum mutual information in the form of bandwidth and SNR
  - very fundamental result

Notice that the different cases reflect different levels of abstraction in modeling the physical communication system between transmitted and received bits or symbols.

After these, we’ll then generalize the capacity analysis to the time- and frequency-selective fading channels

- e.g. concepts of instantaneous and average/ergodic capacity in case of time-selective fading
CAPACITY PER CHANNEL USE, 
case 1: discrete-valued input and output

Now, the channel input is a discrete random variable $X \in \Omega_X$ while the output is another discrete random variable $Y \in \Omega_Y$.

The channel characteristics are assumed to be described through the conditional probabilities of the form (also called transition probabilities, commonly)

$$p_{Y\mid X}(y|x), \quad x \in \Omega_X, y \in \Omega_Y$$

Concrete example: Binary symmetric channel, BSC

- $\Omega_X = \Omega_Y = \{0,1\}$
- transition probabilities illustrated in the diagram, on the right, where $p$ denotes the bit flip probability
- notice that any communication system obeys this model, when viewed at proper interfaces

Now, in the case of discrete-valued input and discrete-valued output, the mutual information $I(X,Y) = H(X) - H(X\mid Y) = H(Y) - H(Y\mid X)$ can be obtained through the following steps:

1) Entropy of $Y$ for a fixed input $X = x$:

$$H(Y\mid X = x) = E\left[-\log_2 p_{Y\mid X}(Y\mid x)\right]$$

$$= -\sum_{y\in\Omega_Y} p_{Y\mid X}(y\mid x) \log_2 p_{Y\mid X}(y\mid x)$$

- this is intuitive since when the conditioning variable is fixed as $X = x$, conditional probabilities behave like normal probabilities
- thus the above is nothing but the basic entropy expression where the conditional probabilities $p_{Y\mid X}(y\mid x)$ are adopted for fixed $X = x$

2) Then, the actual average conditional entropy $H(Y\mid X)$ can be obtained by averaging the previous over different values of $X$ written as

$$H(Y\mid X) = \sum_{x\in\Omega_X} H(Y\mid X = x) p_X(x)$$

$$= -\sum_{x\in\Omega_X} p_X(x) \sum_{y\in\Omega_Y} p_{Y\mid X}(y\mid x) \log_2 p_{Y\mid X}(y\mid x)$$

3) Finally, to evaluate $H(Y)$ and thus the mutual information $I(X,Y) = H(Y) - H(Y\mid X)$, we need the probabilities of the different values of $Y$.

In terms of the transition probabilities and the input probabilities, these read

$$p_Y(y) = \sum_{x\in\Omega_X} p_{Y\mid X}(y\mid x) p_X(x)$$

- why ?
- ... well, think of the meaning of the transition probabilities, and e.g. the example case of BSC

4) Putting all together we can write the mutual information as

$$I(X,Y) = H(Y) - H(Y\mid X)$$

$$= -\sum_{y\in\Omega_Y} p_Y(y) \log_2(p_Y(y))$$

$$+ \sum_{x\in\Omega_X} p_X(x) \sum_{y\in\Omega_Y} p_{Y\mid X}(y\mid x) \log_2 p_{Y\mid X}(y\mid x)$$

where the probabilities of the different values of $Y$ read as given above.
5) Finally, the capacity per channel use is

\[
C_S = \max_{p_x(x)} I(X,Y) \quad \text{[bits per channel use]}
\]

Notice that here, in general, the mutual information expression depends on the input probabilities in two ways

- \( H(Y) \) depends on the input distribution
- \( H(Y | X) \) depends on the input distribution

.. see the expressions on the previous page

However, in some cases where the transition probabilities are symmetric, it may happen that the impact of the input distribution on \( H(Y | X) \) cancels out

- in such cases, maximizing \( I(X,Y) \) over the input distribution means then maximizing directly the output entropy \( H(Y) \)

We’ll illustrate with an example, next.

Notice that, in general, the input distribution can in practice be influenced by the design and optimization of the source and channel coders, as well as by the modulation alphabet design.

---

**Example: binary symmetric channel, BSC**

- \( \Omega_X = \Omega_Y = \{0,1\} \)
- Input probabilities denoted by \( p_X(0) = q, p_X(1) = 1 - q \)
- bit flip probability \( p \)

Based on the previous, the mutual information is now (show it)

\[
I(X,Y) = H(Y) - H(Y | X) = ...
\]

\[
= H(Y) + p \log_2 p + (1 - p) \log_2(1 - p)
\]

Thus here, due to the symmetry of the transition probabilities, the conditional entropy does not eventually depend on the input distribution.

Furthermore, since the output \( Y \) is a binary random variable, its maximum entropy is 1 which is realized when both output values are equally probable, i.e., \( p_Y(0) = p_Y(1) = 0.5 \).

In the above BSC case, it is then easy to show (do it) that

\[
p_Y(0) = p_Y(1) = 0.5
\]

when \( p_X(0) = q = 0.5 \) and \( p_X(1) = 1 - q = 0.5 \).

Thus, we can express the capacity per channel use for the BSC example case finally as (illustration, next page)

\[
C_s = \max_{q} I(X,Y)
\]

\[
= 1 + p \log_2 p + (1 - p) \log_2(1 - p)
\]
Example: binary symmetric channel, BSC, cont'd

\[ C_s = 1 + p \log_2 p + (1 - p) \log_2 (1 - p) \]

The following special cases can be identified:

1. \( p = 1/2 \) \( \Rightarrow C_s = 0 \)
   - Input and output statistically independent, rate is zero

2. \( p = 0 \) or \( p = 1 \) \( \Rightarrow C_s = 1 \)
   - Maximum rate that a BSC can support

Rate \( C_s = 0 \) means that the observation variable is useless, it does not convey any information about the input \( X \):

- Intuitively clear, if you think of the BSC example and the case with \( p = 1/2 \)
- This interpretation of zero rate, however, applies generally

Some complementary notes

Many times people ask the question:

"Even if the bit flip probability is 50% in the channel, e.g., in the previous BSC example, doesn't this still mean that 50% of the bits go through correctly?"

So how come the rate or capacity is zero?"

It is a fair question, but the answer is:

- The rate is zero because the observation variable \( Y \), in that case, does not help the receiver at all, in telling what bit was fed into the channel.
- That is, the receiver can equally well purely guess the transmitted bits and the error rate would still be the same, i.e., 50%.
- Thus, the observation is useless, and the channel does not convey any information at all.
- This way, the rate is zero.

Another thing is that like we noticed in the BSC example, when there is certain symmetry in the transition probabilities \( p_{Y \mid X}(y \mid x) \), the conditional entropy \( H(Y \mid X) \) may not eventually depend on the input probabilities:

- If that is the case, then to maximize \( I(X, Y) = H(Y) - H(Y \mid X) \) over the input probabilities, it is naturally sufficient to maximize the output entropy \( H(Y) \).
- This maximization, in turn, assuming still similar symmetry in the transition probabilities, commonly leads to equally likely input values, i.e., discrete uniform distribution at the source.
  - This is what happened also in our previous BSC example.
- Thus in many cases with symmetrical transition probabilities, equally probable input values will maximize the mutual information.
  - This, however, is not the case if the transition probabilities are not symmetric.
CAPACITY PER CHANNEL USE, case 2: discrete-valued input and continuous-valued output

Next, we take a step closer to the physical world, where additive Gaussian distributed noise is always superimposed to the signal.

Such Gaussian noise distribution is continuous-valued, and thus if we model the receiver observation $Y$ as

$$Y = X + N$$

where $X$ is still the channel input and $N$ refers to Gaussian noise, the observation $Y$ is continuous-valued as well.

In general, the entropy of a continuous-valued random variable, say the noise variable $N$, is commonly expressed as

$$H(N) = \mathbb{E}[-\log_2 f_N(N)] = -\int_{-\infty}^{\infty} f_N(n) \log_2 f_N(n) \, dn$$

In applied mathematics, this is called differential entropy, to strictly-speaking differentiate from the entropy of a discrete random variable

- the reason for this is: different from the discrete random variable case, the (differential) entropy of a continuous random variable can also be negative or tend towards infinity
- this means, strictly speaking, that the exact values of the differential entropy are not necessarily meaningful but comparing different differential entropies is still certainly meaningful

Furthermore, the entropy of continuous random variables can also be meaningfully bounded if the variance is fixed.

Based on applied math literature, if the variance of $N$ is fixed, say $\sigma_N^2$, the differential entropy of $N$ is always upper-bounded as

$$H(N) \leq \frac{1}{2} \log_2 \left( 2\pi e \sigma_N^2 \right)$$
Capacity per channel use, case 2: discrete input and continuous output, cont’d

Now, when the observation variable $Y$ is continuous-valued, the conditional entropy $H(Y|X)$ can be expressed as

$$H(Y|X) = \sum_{x \in \Omega_X} H(Y|X=x)p_X(x)$$

$$= -\sum_{x \in \Omega_X} p_X(x) \int_{\Omega_Y} f_{Y|X}(y|x) \log_2 f_{Y|X}(y|x) \, dy$$

The average mutual information and the channel capacity, in turn, read (formally exactly like earlier)

$$I(X,Y) = H(X) - H(X|Y)$$
$$= H(Y) - H(Y|X)$$

$$C_s = \max_{p_X(x)} I(X,Y)$$

where the probability distribution of $Y$ needed to evaluate $H(Y)$ is given by

$$f_Y(y) = \sum_{x \in \Omega_X} f_{Y|X}(y|x)p_X(x)$$

Again, in general, the input probabilities impact both $H(Y)$ and $H(Y|X)$.

We will illustrate with some concrete examples soon, particularly the additive Gaussian noise case, after we have first treated the most general case – that is continuous-valued input and continuous-valued output.

CAPACITY PER CHANNEL USE, case 3: continuous input and continuous output

This is the most general case, where also the input variable is allowed to be continuous-valued with probability distribution $f_X(x)$, i.e., the input alphabet is not constrained to any discrete structure.

In this case, the conditional entropy reads

$$H(Y|X) = \int_{\Omega_X} \int_{\Omega_Y} f_Y(y|x) \log_2 f_Y(y|x) \, dy \, dx$$

Otherwise, formally, the approach is identical to the previous cases, i.e., the mutual information and the capacity per channel use read

$$I(X,Y) = H(X) - H(X|Y)$$
$$= H(Y) - H(Y|X)$$

$$C_s = \max_{f_X(x)} I(X,Y)$$
CAPACITY OF GAUSSIAN NOISE CHANNEL, continuous input

As a concrete example, we consider now the case where the input and output are related through additive Gaussian noise model of the form

\[ Y = X + N \]

Here, \( X \) refers to the continuous input variable, with variance \( \sigma^2_X \), and \( N \) denotes the Gaussian noise, variance \( \sigma^2_N \), while noise is assumed to be statistically independent of the input (physically plausible).

Directly based on the additive model, it then follows that

\[ H(Y|X) = H(N) \] (why?).

Thus, the mutual information reads

\[ I(X,Y) = H(Y) - H(Y|X) = H(Y) - H(N) \]

where \( H(N) = \frac{1}{2} \log_2 \left( 2\pi e \sigma^2_N \right) \) as discussed earlier.

Then, assuming a given noise variance, it is clear that the above mutual information is maximized when the output entropy \( H(Y) \) is maximized!

On the other hand, the linear signal model with independent variables imply \( \sigma^2_Y = \sigma^2_X + \sigma^2_N \), and thus based on the earlier discussion regarding the differential entropy properties under fixed variance, it follows that

\[ H(Y) \leq \frac{1}{2} \log_2 \left( 2\pi e \sigma^2_Y \right) = \frac{1}{2} \log_2 \left( 2\pi e (\sigma^2_X + \sigma^2_N) \right) \]

where the equality holds if and only if the output \( Y \) is Gaussian distributed (why?).

Based on the linear signal model and assumed Gaussian distribution for the noise, the output \( Y \) can be Gaussian distributed only if the input \( X \) is also Gaussian distributed!

Capacity of Gaussian noise channel, continuous input, cont’d

Thus, we can conclude that in the additive Gaussian noise model, the maximum mutual information (the capacity) is achieved when also the input variable is Gaussian distributed.

The corresponding capacity per channel use reads then

\[ C_s = \frac{1}{2} \log_2 \left( \frac{1 + \frac{\sigma^2_X}{\sigma^2_N}}{1 + \frac{\sigma^2_N}{\sigma^2_N}} \right) \]

where \( \frac{\sigma^2_X}{\sigma^2_N} \) can be interpreted as the signal-to-noise ratio, SNR.

- the result is very intuitive, the larger the SNR the larger is the mutual information
- numerical examples will follow soon

Notice that the above result has been derived assuming real-valued random variables
- I/Q modulation, in turn, facilitates using complex-valued baseband signals and variables in carried modulated systems
- Thus, from the capacity analysis point of view, we can think that there are two parallel channels, when viewed from baseband I and Q inputs at TX to the baseband I and Q observations at RX
- In such cases, if \( \frac{\sigma^2_X}{\sigma^2_N} \) denotes the effective SNR per I and Q channel, and if we assume that the I and Q channels are fed with statistically independent inputs and that the noise variables at the I and Q at RX are independent, the corresponding capacity per complex-valued channel use is commonly expressed as

\[ C_s = \log_2 \left( 1 + \frac{\sigma^2_X}{\sigma^2_N} \right) \]
Capacity of Gaussian noise channel, discrete input

In practical digital communications, like we have learned earlier, the channel input is always a discrete-valued variable
- symbols from selected constellation or symbol alphabet
- some common examples below

Now, after establishing previously that the maximum mutual information is realized, in an additive Gaussian noise channel, when also the input is continuous Gaussian variable, an obvious question then is:
- how much do we lose in capacity by using the practical discrete constellations, instead of Gaussian input?

We'll address that next, for simplicity in case of real-valued variables

<table>
<thead>
<tr>
<th></th>
<th>4-PSK</th>
<th>8-PSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-AMPM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Capacity of Gaussian noise channel, discrete input, cont'd

In case of discrete channel input and additive Gaussian noise, that is for an observation of the form $Y = X + N$ where the input $X$ is discrete and noise $N$ is Gaussian, the mutual information still reads formally

$$I(X,Y) = H(Y) - H(Y | X) = H(Y) - H(N)$$

where $H(N) = \frac{1}{2} \log_2 (2\pi e \sigma_N^2)$. The entropy of the output, $H(Y)$, in turn must be evaluated using the distribution of $Y$.

For $Y = X + N$, the distribution of the output $Y$ reads

$$f_Y(y) = \sum_{x \in \Omega_X} f_{Y|X}(y | x)p_X(x) = \sum_{x \in \Omega_X} \left\{ \frac{1}{\sqrt{2\pi\sigma_N^2}} e^{-(y-x)^2/(2\sigma_N^2)} p_X(x) \right\}$$

which is basically a "Gaussian-mixture" type of distribution, that is, a sum of multiple Gaussian distributions, each centered at one of the discrete input symbols while being then also weighted by the corresponding input symbol probability
- you can easily see this by considering first that if the input symbol is fixed to one particular symbol, what is the corresponding output distribution (it is just a shifted Gaussian, right?)
- then, the total output distribution follows by averaging over the different input symbols

Assuming then, for simplicity, all input symbols to be equally probable, the differential entropy $H(Y) = E[-\log_2 f_Y(Y)]$ is easy to evaluate, if not by pen and paper, then at least numerically using a computer (numerical integration methods).

This is what has been done next, and the results are illustrated on the following pages
- showing also the behavior with continuous-valued Gaussian input, for reference
Capacity of Gaussian noise channel, cont’d

Real-valued constellations:

The essential engineering conclusion is then: the results show that discretizing the channel input yields some loss in the capacity but as long as the input constellation size is chosen properly, relative to the SNR range at which the system operates, the loss is very small, compared to the theoretical capacity with Gaussian input:
- this raises the idea of adaptive modulation – tune the modulation order based on SNR
- low SNR => low-order modulation, high-SNR => high-order mod.

Notice also that the capacity of the discrete alphabets saturates at large SNR to \( \log_2(\Omega_X) \) where \( \Omega_X \) denotes the size of the alphabet:
- this is very natural since with equally probable input symbols, the input entropy \( H(X) = \log_2(\Omega_X) \)
- obviously this is the maximum that can pass through since \( I(X,Y) = H(X) - H(X|Y) \)

Complex-valued constellations:

Here the conclusions are practically identical to the previous slide.

As a side note, the figures also show the SNR points at which an example detection error probability of \( 10^{-5} \) can be obtained, with the different symbol constellations, using a practical minimum distance detector at the receiver and assuming no error control coding:
- we will deal with the error probability analysis and coding aspects later in this course!
All physical transmission media and thus the transmitted and received signals are, eventually, continuous-time and bandlimited, and subject to noise.

For simplicity, we consider a tightly bandlimited baseband channel with a transfer function of the form

\[ B(f) = \begin{cases} 1 & \text{when } |f| \leq W \\ 0 & \text{when } |f| > W \end{cases} \]

where \( W \) denotes the physical bandwidth in Hz.

Thus, assuming that the channel adds white Gaussian noise to the signal and that the receiver can filter out all out-of-band noise, the received signal after RX filtering can be expressed as

\[ Y(t) = X(t) + N(t) \]

where \( X(t) \) denotes the bandlimited communication waveform and \( N(t) \) denotes bandlimited Gaussian noise (noise at passband).

Such a model is commonly called a bandlimited AWGN channel.

Now, if the receiver samples the bandlimited received signal at a rate \( f_s \geq 2W \), we obtain a corresponding discrete-time model of the form

\[ Y_k = X_k + N_k \]

where \( Y_k = Y(kT_s) \), \( X_k = X(kT_s) \), \( N_k = N(kT_s) \). The sample rate requirement of \( f_s \geq 2W \) stems directly from Nyquist sampling theorem.

The capacity of bandlimited continuous-time channels, cont’d

Now based on our earlier results, and assuming real-valued signals for simplicity, the capacity of the system, per channel use, reads

\[ C_s = \frac{1}{2} \log_2 \left( 1 + \frac{\sigma_X^2}{\sigma_N^2} \right) \]

where it is assumed that the distribution of \( X_k \) is also Gaussian (see earlier pages). Furthermore, \( \sigma_X^2 \) and \( \sigma_N^2 \) denote the variances of the discrete-time received signal and noise variables \( X_k \) and \( N_k \).

Then, for the critical sample rate of \( f_s = 2W \), the consecutive samples of the received signal \( Y_k = X_k + N_k \) are statistically independent, and thus the capacity of the bandlimited continuous-time channel, in bits per second, is given by

\[ C = 2W \times C_s = W \log_2 \left( 1 + \frac{\sigma_X^2}{\sigma_N^2} \right) \]

This means nothing but summing \( C_s = \frac{1}{2} \log_2 \left( 1 + \frac{\sigma_X^2}{\sigma_N^2} \right) \) bits per channel use over \( 2W \) independent channel uses per second.

This is a very fundamental result for bandlimited noisy continuous time channels, and is known as the Shannon-Hartley capacity law:

- direct interpretations are: bandwidth impacts capacity linearly while the SNR impacts the capacity logarithmically
- though we come back to these interpretations, soon

Notice that even though the sample rate in the receiver may easily be, in practice, larger than \( 2W \) (i.e., \( f_s > 2W \)), this does not bring any benefit in information theoretic sense:

- this is because for \( f_s > 2W \), the consecutive samples are already correlated and thus the channel uses are not independent any more
Example capacities for different bandwidths and SNR’s

Below, we give some example numerical values by directly evaluating the capacity law for different values of $W$ and SNR

$$C = W \log_2 \left(1 + \frac{S}{N}\right)$$

<table>
<thead>
<tr>
<th>$W$</th>
<th>SNR</th>
<th>Capacity (approx.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 kHz</td>
<td>30 dB = 1000</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>200 kHz</td>
<td>0 dB = 1</td>
<td>200 kbps</td>
</tr>
<tr>
<td>200 kHz</td>
<td>-10 dB = 0.1</td>
<td>27.5 kbps</td>
</tr>
<tr>
<td>20 MHz</td>
<td>30 dB = 1000</td>
<td>200 Mbps</td>
</tr>
<tr>
<td>20 MHz</td>
<td>0 dB = 1</td>
<td>20 Mbps</td>
</tr>
<tr>
<td>20 MHz</td>
<td>-10 dB = 0.1</td>
<td>2.75 Mbps</td>
</tr>
<tr>
<td>1 GHz</td>
<td>30 dB = 1000</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>1 GHz</td>
<td>0 dB = 1</td>
<td>1 Gbps</td>
</tr>
<tr>
<td>1 GHz</td>
<td>-10 dB = 0.1</td>
<td>140 Mbps</td>
</tr>
</tbody>
</table>

Some complementary notes

We derived the fundamental capacity law, strictly speaking, assuming real-valued variables and physical baseband bandwidth of $W$.

However, the very same law applies also to the I/Q carrier-modulated system, since

- if the physical RF bandwidth is $W$, then the baseband equivalent covers the frequencies $-W/2 ... W/2$, that is, the physical bandwidth at baseband is only $W/2$
- thus, the critical sample rate to obtain independent channel uses is only $f_s = W$
- however, like we discussed earlier, the capacity per channel use for complex I/Q variables is of the form $C_s = \log_2 \left(1 + \frac{\sigma_X^2}{\sigma_N^2}\right)$
- thus, as a whole, the same law of $C = W \log_2 \left(1 + \frac{S}{N}\right)$ still applies

Thus, in summary, the fundamental capacity law of

$$C = W \log_2 \left(1 + \frac{S}{N}\right)$$

applies for both baseband and RF systems, and the bandwidth $W$ is always the physical bandwidth of the system!
About the impact of bandwidth on capacity

Like was shown previously, the bandlimited AWGN channel capacity reads

\[ C = W \log_2 \left( 1 + \frac{S}{N} \right) \]

where \( W \) denotes the physical bandwidth available for communication, measured in Hz, \( S \) denotes the useful signal power and \( N \) the inband noise power, both at receiver.

Denoting the double-sided noise spectral density by \( N_0 [\text{Watts/Hz}] \), the inband noise power reads directly \( N = 2WN_0 \).

Thus by substituting \( N = 2WN_0 \), we can express the channel capacity as

\[ C = W \log_2 \left( 1 + \frac{S}{2N_0W} \right) \]

From these expressions, it follows that:

- increasing the bandwidth \( W \) yields linear increase in capacity but only if the SNR is constant (see the upper expression)
- however, since increased bandwidth always implies increased noise power, the capacity increase is eventually not linear (lower expression), if the useful received power \( S \) is fixed
- for a fixed bandwidth \( W \), increasing the SNR increases the capacity logarithmically (see the upper expression)

About the impact of bandwidth on capacity, cont’d

The engineering conclusion regarding the impact of the bandwidth is:

In order to capitalize efficiently the capacity increase through increased bandwidth, also the received signal power should be increased to balance the increased noise bandwidth such that the SNR stays at a reasonable level

- this way you can truly capitalize the linear increase in capacity through increased bandwidth

Thus, the capacity analysis shows that two basic resources, bandwidth and power, should not be treated independently. They are always bound to each other, through the noise power that depends on the bandwidth.
Capacity under interference

Many practical systems, particularly cellular networks, are co-channel interference limited

- in cellular networks, this is stemming from the frequency reuse

Assuming that the interference is Gaussian distributed, and independent of the actual noise and useful signal entering the receiver, the previous capacity law can be directly applied as long as we replace the noise power by the sum of the noise and interference powers

That is, the capacity reads

\[ C = W \log_2 \left( 1 + \frac{S}{N + I} \right) \]

where \( I \) refers to the inband/co-channel noise power at receiver.

This is commonly written as

\[ C = W \log_2 \left( 1 + \text{SINR} \right) \]

where \( \text{SINR} \) refers to the signal to interference plus noise ratio.

CHANNEL CAPACITY UNDER FADEING

Next, we address the channel capacity aspects incorporating also the effects of time-selective and frequency-selective fading

- time-selective fading:
  - the useful signal level varies at receiver
  - leads to the concept of instantaneous capacity (capacity per given fading state) and ergodic capacity (average capacity over different fading states)
  - fundamental aspects, particularly in mobile communications (e.g. cellular networks) due to Doppler phenomenon

- frequency-selective fading
  - different frequencies of the signal fade differently
  - thus the received SNR or SINR is actually frequency dependent, or at least can be quantified separately for different frequencies/sub-bands of the received signal
  - thus also the capacity is different, per sub-band or subcarrier, while the total channel capacity is then sum of the capacities of the sub-bands
  - fundamental aspect, again in broadband mobile communications (e.g. cellular networks) due to delay spread
Here are address the capacity of a noisy bandlimited channel that is also impacted by a time-selective fading process.

We consider first the capacity per channel use, and model the receiver observation as

\[ Y = hX + N \]

where \( X \) refers to the channel input variable, with variance \( \sigma_X^2 \), \( N \) denotes additive Gaussian noise, with variance \( \sigma_N^2 \), while \( h \) denotes the fading process whose variance is denoted by \( \sigma_h^2 \).

- all variables are assumed to be mutually statistically independent which is physically plausible
- most common example for the fading process is the so-called Rayleigh fading for which \( |h| \) is Rayleigh distributed and thus \( |h|^2 \) is exponentially distributed
  - numerous other fading distributions also exist but Rayleigh fading is the most common one

Now, considering first the case that the value of the fading process is fixed, this model is identical to the earlier Gaussian noise case except for the received SNR definition which now reads

\[ SNR_h = \frac{|h|^2}{\sigma_X^2} \]

- This is commonly called the instantaneous received SNR since it describes the received SNR for the given fading state \( h \)
- Under Rayleigh fading, \( |h|^2 \) is exponentially distributed and thus also the instantaneous SNR is exponentially distributed

Based on our earlier developments, the input distribution that maximizes the mutual information between the transmitter and receiver, for a given fading state, is Gaussian distribution under which the instantaneous capacity per channel use reads

\[ C_{s,h} \approx \log_2 \left( 1 + SNR_h \right) = \log_2 \left( 1 + \frac{|h|^2}{\sigma_X^2} \right) \]

- this is the instantaneous capacity per channel use, i.e., the capacity per channel use for a given fading state \( h \)

Taking then the band-limitation constraint into account, as earlier, the corresponding instantaneous capacity in bits/s reads

\[ C_h = W \log_2 \left( 1 + SNR_h \right) = W \log_2 \left( 1 + \frac{|h|^2}{\sigma_X^2} \right) \]

where \( W \) denotes the physical bandwidth.
Channel capacity under time-selective fading, cont’d

Then, the so-called **ergodic Shannon capacity** is defined as the statistical average of the instantaneous capacity, expressed formally as

\[
C = E[C_h] = W E[\log_2 (1 + SNR_h)] = W E[\log_2 \left(1 + \frac{|h|^2}{\sigma_s^2} \right) \frac{\sigma_s^2}{\sigma_N^2}]
\]

where \(E[.\]\) refers to expectation over the fading distribution.

For Rayleigh fading, \(|h|^2\) is exponentially distributed and the above expectation can always be easily solved numerically (for given \(W\), \(\sigma_s^2\) and \(\sigma_N^2\)).

There is also a well-known upper bound of the form

\[
C = E[C_h] = W E[\log_2 (1 + SNR_h)] \\
\leq W \log_2 \left(1 + E[\frac{|h|^2}{\sigma_s^2} \frac{\sigma_s^2}{\sigma_N^2}]\right) \\
= W \log_2 \left(1 + \frac{E[|h|^2]\sigma_s^2}{\sigma_N^2}\right) \\
= W \log_2 \left(1 + \frac{\sigma_h^2\sigma_s^2}{\sigma_N^2}\right) \\
= W \log_2 \left(1 + SNR_{ave}\right)
\]

where \(SNR_{ave} = E[SNR_h] = \sigma_h^2\sigma_s^2 / \sigma_s^2\) denotes the average received SNR while the approximation itself stems from the well-known Jensen’s inequality from applied math

- this upper bound applies independently of the fading distribution!

The ergodic capacity upper bound of the form

\[
C \leq W \log_2 \left(1 + SNR_{ave}\right)
\]

is intuitive

- capacity under fading is upper-bounded by the corresponding capacity of noise only channel if the average received SNRs are the same
- hence, fading reduces the capacity

A graphical illustration below, showing both the exact ergodic capacity under Rayleigh fading (obtained through numerical averaging) and the upper bound (applicable for any fading distribution)

- we may possibly address this issue further in the classroom and/or Matlab exercises.
Channel capacity under time-selective fading, cont'd

In addition to the ergodic capacity, also the **concept of outage capacity** is commonly used in the context of fading channels.

Outage capacity is based on defining a minimum instantaneous received SNR, $\text{SNR}_{h,\text{min}}$, and the corresponding minimum capacity per channel use

$$C_{h,\text{min}} \simeq \log_2 \left( 1 + \text{SNR}_{h,\text{min}} \right)$$

When the true instantaneous received SNR is larger than or equal to $\text{SNR}_{h,\text{min}}$, the receiver can always decode the signal correctly since the true instantaneous capacity is larger than or equal to $\text{SNR}_{h,\text{min}}$.

On the other hand, it is assumed that when the instantaneous received SNR is below $\text{SNR}_{h,\text{min}}$, receiver cannot anymore decode the received signal correctly, and hence it is said that the receiver is in outage.

The probability of such outage to take place is, by definition,

$$P_{\text{outage}} = \text{Prob.}(\text{SNR}_h < \text{SNR}_{h,\text{min}})$$

In case of Rayleigh fading channel, instantaneous SNR is exponentially distributed with mean value of $E[\text{SNR}_h] = \text{SNR}_{\text{ave}} = \frac{\sigma^2_{\text{IN}}}{\sigma^2_X}$ and thus the above probabilities stem directly from the corresponding cumulative distribution function of the exponential distribution, written as

$$P_{\text{outage}} = \text{Prob.}(\text{SNR}_h < \text{SNR}_{h,\text{min}}) = 1 - e^{-\frac{1}{\text{SNR}_{\text{ave}}}}$$

Example outage probabilities for $\text{SNR}_{\text{ave}} = 0 \text{ dB}, 10 \text{ dB}, 20 \text{ dB}$ are plotted next page vs. the required minimum received SNR. The second figure then shows the outage capacity vs. outage probability.
Next, we consider the capacity analysis under an additional assumption that the transmitter has knowledge about the fading process:

- thus, the transmitter can do power adaptation based on the fading state

We first recall the previous definition for the average capacity, rewritten as

$$C = E[C_h] = WE[ \log_2 (1 + SNR_h) ]$$

where $E[.]$ refers to expectation over the fading distribution, and $SNR_h = |h|^2 / \sigma_N^2$.

Now, in case the transmitter is adapting the TX power based on the fading state, the question naturally is: what is the best way to adapt the power?

To find the answer to this question, we first redefine the average capacity as

$$C = E[C_h] = WE[ \log_2 \left( 1 + \frac{|h|^2 P_{tx}(|h|^2)}{\sigma_N^2} \right) ]$$

where $P_{tx}(|h|^2)$ denotes now the TX power that in general depends on the fading state $h$ (or $|h|^2$ like written above).

Thus, the problem is to maximize the above capacity, i.e., find power adaptation policy $P_{tx}(|h|^2)$ such that the above capacity is maximized, under an average transmit power constraint:

$$C_{max} = \max_{P_{tx}(|h|^2)} WE[ \log_2 \left( 1 + \frac{|h|^2 P_{tx}(|h|^2)}{\sigma_N^2} \right) ]$$

subject to $E[ P_{tx}(|h|^2) ] \leq P_{ave}$

The previous problem is a nonlinear optimization problem with a linear constraint, and can be solved e.g. through differentiating an appropriate Lagrangian function against the power policy $P_{tx}(\cdot)$ and setting the derivative equal to zero:

- don’t worry, this is just math, the final result is interesting

By doing the above, we’ll obtain the following power allocation policy

$$P_{tx}(|h|^2) = \begin{cases} 
C - \frac{\sigma_N^2}{|h|^2} & \text{when } |h|^2 \geq g_{th} \\
0 & \text{when } |h|^2 < g_{th}
\end{cases}$$

where $g_{th} = \sigma_N^2 / C$ denotes a threshold for the instantaneous fading state $|h|^2$ below which no power is allocated at all (i.e., nothing is transmitted), while $C$ denotes a constant, which can be solved from the average transmit power constraint as

$$E[ P_{tx}(|h|^2) ] = \int_{g_{th}}^{\infty} (C - \frac{\sigma_N^2}{g}) f(g) dg = P_{ave} \text{, or}$$

$$C \int_{\sigma_N^2/C}^{\infty} f(g) dg = P_{ave} + \frac{\sigma_N^2}{C} \int_{\sigma_N^2/C}^{\infty} \frac{1}{g} f(g) dg$$

where $f(\cdot)$ denotes the probability distribution of $|h|^2$ and for notational convenience $g = |h|^2$ is used as the integration variable.

- these latter expressions are nothing but writing out explicitly the average TX power constraint $E[ P_{tx}(|h|^2) ] = P_{ave}$
- notice that there is no closed-form solution for $C$, but it can always be solved numerically
Channel capacity under time-selective fading, cont’d

The corresponding maximum capacity is obtained by directly substituting the power allocation policy to the earlier capacity formula, written here as:

\[
C_{\text{max}} = \max_{P_{tx}(|h|^2)} WE\left[ \log_2 \left( 1 + \frac{|h|^2 P_{tx}(|h|^2)}{\sigma_N^2} \right) \right] \\
= W \int_{\sigma_N^2/C}^{\infty} \log_2 \left( \frac{gC}{\sigma_N^2} \right) f(g) dg
\]

The engineering conclusions are:

- the power adaptation policy on the previous slide implies that there is a threshold for the instantaneous fading process below which nothing is transmitted:
  - for these fading states, the instantaneous rate is zero; transmitter does not even try to send anything
- for all other fading states, the transmitter allocates power according to the inverse of the squared instantaneous fading:
  - strong instantaneous channel => lots of power
  - weak instantaneous channel => much less power
  - so from the ergodic capacity maximization perspective, it makes sense to allocate most of the available power (under the average power constraint) to those instants when the channel is strong, and thus not even trying to “equalize” the fading deeps through added power (exactly the opposite)!
- this is commonly referred to as the water-filling principle in the literature, see next slide for a visual illustration

An illustration of the water filling principle is given below for an example fading realization shown in the top part of the figure:

- we will investigate concrete examples in the exercises (classroom or Matlab)

\[
P_{\text{ts}}(|h|^2) = \begin{cases} 
C - \frac{\sigma_N^2}{|h|^2} & \text{when } |h|^2 \geq g_{\text{th}} \\
0 & \text{when } |h|^2 < g_{\text{th}}
\end{cases}
\]
Next we address the capacity analysis of noisy and bandlimited channels under frequency-selective fading

- an example illustration above, again key issue in broadband wireless systems due to multipath delay spread
- here we assume that the channel response does not change over time

For capacity analysis purposes, we split the overall bandwidth $W$ into many narrow subbands, such that the fading response is approximatively constant within the sub-band

Notations in the continuation are

- sub-band index: $k$
- sub-band width: $\Delta f$
- TX power density at sub-band $k$: $P_k$
- channel response at sub-band $k$: $H_k$
- noise power density $N_0$

Channel capacity under frequency-selective fading, cont’d

Now, stemming directly from our earlier developments, the capacity of the sub-band $k$ reads

$$C_k = \Delta f \log_2 \left( 1 + SNR_k \right)$$

$$= \Delta f \log_2 \left( 1 + \frac{\Delta f P_k |H_k|^2}{\Delta f N_0} \right) = \Delta f \log_2 \left( 1 + \frac{P_k |H_k|^2}{N_0} \right)$$

Total capacity is then the sum of sub-band capacities, over the bandwidth $W$, which reads

$$C = \sum_k C_k = \Delta f \sum_k \log_2 \left( 1 + \frac{P_k |H_k|^2}{N_0} \right)$$

In the limiting case, with $\Delta f \to 0$, we get

$$C = \int_W \log_2 \left( 1 + \frac{P(f)|H(f)|^2}{N_0} \right) df$$

In the case that the transmitter does not know anything about the channel response, the only reasonable way to allocate the total transmit power $P_{TOT}$ is to divide it equally to all sub-bands, i.e.,

$P(f) = P_{TOT} / W$

This yields the following capacity of the form

$$C = \int_W \log_2 \left( 1 + \frac{P_{TOT} |H(f)|^2}{WN_0} \right) df$$
Channel capacity under frequency-selective fading, cont’d

On the other hand, if the transmitter does know the channel response, then the question is: how to distribute the total power to the different frequencies, such that the capacity is maximized, under the total power constraint?

Thus, our capacity maximization problem is of the form

\[
C_{\text{max}} = \max_{P(f)} \int W \log \left( 1 + \frac{P(f)|H(f)|^2}{N_0} \right) df
\]

subject to total power constraint

\[
\int W P(f) df \leq P_{\text{TOT}}
\]

Also this is a non-linear optimization problem with linear constraint, and can be solved similarly (using Lagrangian etc) to the earlier time-domain power allocation problem.

This yields essentially the following power allocation policy over frequency

\[
P(f) = \begin{cases} 
K - \frac{N_0}{|H(f)|^2} & \text{when } |H(f)|^2 \geq G_{\text{th}} \\
0 & \text{when } |H(f)|^2 < G_{\text{th}} 
\end{cases}
\]

where \(G_{\text{th}} = N_0 / K\) denotes a threshold for the frequency-selective channel response below which no power is allocated (i.e., nothing is transmitted), while \(K\) denotes again a constant, which can be solved from the average transmit power constraint as

\[
W_{\text{act}} K = P_{\text{TOT}} + \int W_{\text{act}} \frac{N_0}{|H(f)|^2} df
\]

where \(W_{\text{act}}\) refers to those frequencies at which \(|H(f)|^2 \geq G_{\text{th}} = N_0 / K\)

- notice that also this cannot be solved in closed-form for the constant \(K\) since its valued is implicitly impacting the set of frequencies over which the integral is calculated through the threshold \(|H(f)|^2 \geq G_{\text{th}} = N_0 / K\)
- thus it has to be solved numerically, in practice

The corresponding maximum capacity is then obtained by directly substituting the obtained power allocation policy to the capacity formula, written as

\[
C_{\text{max}} = \max_{P(f)} \int W \log \left( 1 + \frac{P(f)|H(f)|^2}{N_0} \right) df
\]

\[
= \int W_{\text{act}} \log \left( \frac{K|H(f)|^2}{N_0} \right) df
\]
The engineering conclusions are:

- the frequency-domain power adaptation policy on the previous slides implies that there is a threshold \( G_{th} = N_0 / K \) for the frequency-selective channel response values below which nothing is transmitted
  - for these sub-bands where the response is below the threshold, the rate is zero; transmitter does not even try to send anything at these sub-bands
- for all other sub-bands, the transmitter allocates power according to the inverse of the squared channel response
  - strong sub-band \( \Rightarrow \) lots of power
  - weak sub-band \( \Rightarrow \) much less power
  - so from the total capacity maximization perspective, it makes sense to allocate most of the available power (under the average power constraint) to those sub-bands where the channel is strong!
- this is commonly called the principle of "water-filling in frequency", and is illustrated next page
  - the first part of the figure shows an example frequency selective channel response while the lower part then illustrates the transmit power allocation
  - we’ll seek to illustrate also this, through a concrete example, in the exercises
Some complementary notes

Also the most generic case of time-varying frequency-selective fading channel and it’s ergodic/average capacity can be addressed
- without and with channel state information in the transmitter
- if the transmitter has channel state information, it leads to joint water-filling in both time and frequency

Furthermore, transmitter power allocation to maximize the ergodic or average capacity is only one approach to use channel state information at TX

One example alternative approach is the so-called channel inversion principle
- allocate the TX power, either in time or frequency, so that constant instantaneous SNR, per time or frequency unit, is observed at RX
  - the larger the fading deep, the larger the TX power (which is basically the opposite of the earlier approach)
- this leads to a constant rate following from the constant received SNR
- the ergodic/average capacity with this approach is, however, lower

Some complementary notes, cont’d

In most of the practical systems, the exact water-filling is not adopted
- this is already due to the fact knowing e.g. the exact fading state of the channel is not feasible
- though in many systems, devices report the channel state back to the base-station, but the resolution in time and frequency is somewhat limited
- also, there are hardware related constraints that do not allow for aggressive adaptation of power in time or frequency

On the other, practical systems do contain some elements that have similarity to the previous concepts and results
- avoiding the deepest fading notes in both time and frequency is indeed the target also in practice
  - this is realized through time- and frequency domain scheduling and bandwidth allocation, such that devices and scheduled to transmit and receive only at those time-frequency slots where the channel is reasonably good
- also, even though the instantaneous power may not follow the fading characteristics, the fact that modulation and coding schemes are adapted, in both time and frequency, according to the channel state leads to instantaneous rates that are following the channel state characteristics
3. BASEBAND DIGITAL COMMUNICATION

Bits, Symbols, and baseband waveforms

The starting point in the baseband digital transmission is the utilization of pulse amplitude modulation (PAM) for transmission of binary bit sequences or generally multilevel symbol sequences.

A multilevel symbol is obtained when, e.g., 4 bits are combined into one higher-level symbol. In this case, \(2^4 = 16\) different symbol levels are required to represent all possible bit combinations. Generally, \(B\) bits can be represented using \(2^B\) different levels.

The number of used symbol levels (i.e., the number of bits / symbol) is selected based on the requirements of the application and transmission channel so that different symbol levels can still be reliably distinguished in the receiver (in the presence of noise and other distortions).

When several bits are combined into one symbol, the used symbol rate (symbols/second, sometimes also called “baud rate”) can be decreased. This affects directly the required bandwidth as we will see later on.

Simple toy example:

- **bit sequence:** 0 1 0 0 1 1 0 0 ... bits/s
- **symbol sequence:** -3A 9A ... symb/s

- **binary signal:**
- **16-level signal:**

Bit rate, symbol rate and symbol alphabet size (\(M\)) are always related as:

\[ R_{\text{bit}} = \log_2(M) \times R_{\text{sym}} \quad \text{[bits/s = bits/symbol} \times \text{symbols/s]} \]

Bit to symbol mapping, simple examples

Suppose we have a 4 level symbol alphabet, i.e., \(M = 4\), and the symbol values are for example \(-3, -1, +1, +3\)

Then, we can represent all possible bit combinations of length \(\log_2(4) = 2\) and the mapping could e.g. look like:

\[
\begin{align*}
0 0 & \rightarrow -3 & 0 1 1 0 1 1 1 0 0 0 ... \\
0 1 & \rightarrow -1 & 1 1 & + 3 & + 1 & + 3 & - 3 ... \\
1 1 & \rightarrow +1 \\
1 0 & \rightarrow +3
\end{align*}
\]

Then, we can represent all possible bit combinations of length \(\log_2(4) = 2\) and the mapping could e.g. look like:

\[
\begin{align*}
0 0 & \rightarrow -7 & 0 1 1 0 1 1 1 0 0 0 ... \\
0 0 1 & \rightarrow -5 \\
0 1 1 & \rightarrow -3 \\
0 1 0 & \rightarrow -1 \\
1 1 0 & \rightarrow +1 \\
1 1 1 & \rightarrow +3 \\
1 0 1 & \rightarrow +5 \\
1 0 0 & \rightarrow +7
\end{align*}
\]

Notice that in the above mapping, the bit combinations of the neighboring symbol values differ by only 1 bit

- this is called Gray coding or Gray mapping
- this is sensible, since if the receiver makes an error in distinguishing between different symbol levels (due to e.g. noise), then it is likely that error happens between neighboring levels
- we will talk about this much more, later in this course

Another example, \(M = 8\):

\[
\begin{align*}
0 0 0 & \rightarrow -7 & 0 1 1 0 1 1 1 0 0 0 ... \\
0 0 1 & \rightarrow -5 \\
0 1 1 & \rightarrow -3 \\
0 1 0 & \rightarrow -1 \\
1 1 0 & \rightarrow +1 \\
1 1 1 & \rightarrow +3 \\
1 0 1 & \rightarrow +5 \\
1 0 0 & \rightarrow +7
\end{align*}
\]
**PAM principle and pulse shapes**

In pulse amplitude modulation (PAM), the symbols are transmitted to the channel using pulses such that the pulse amplitude represents the symbol value (hence the name PAM).

Therefore, the total signal \( x(t) \) transmitted to the channel, corresponding to a sequence of symbols \( a_k \), is composed of a sequence of pulses where different pulses are weighted with the corresponding symbol values expressed as:

\[
x(t) = \sum_k a_k p(t - kT)
= ... + a_0 p(t) + a_1 p(t - T) + ...
\]

Here \( T \) is the **symbol interval**, the **symbol rate** is \( f_{sym} = 1/T \), and \( p(t) \) is the **fundamental pulse shape**.

In baseband PAM, the spectral contents of the transmit signal \( x(t) \) is around zero frequency, meaning that the used pulse shape \( p(t) \) is of lowpass type.

- such baseband signals are naturally not directly applicable in wireless/radio comms, so later we generalize the developments to include also carrier modulation.

In general, it is important that pulses representing consecutive symbols do not interfere with each other in the reception, in particular at those moments where the receiver is interpreting the amplitudes of the individual pulses.

- symbol interval \( T \) is the difference between two neighboring symbols, and thus between **two neighboring pulses**.

In the ideal case, the following condition is thus fulfilled (why?):

\[
p(t) = \begin{cases} 1 & \text{when } t = 0 \\ 0 & \text{when } t = \pm T, \pm 2T, ... \end{cases}
\]

So called Nyquist criterion to avoid inter-symbol interference.

The previous Nyquist criterion for zero inter-symbol interference, ISI, can be practically implemented in two different ways:

1) Using short pulses that are not overlapping in time
   - thus the pulse width is limited to the symbol interval \( T \)
   - Simple implementation, however, the bandwidth is not the smallest possible
   - Example: a rectangular pulse duration identical to the symbol interval => sinc type of spectrum, not strictly bandlimited at all (there will be a graphical example soon)
   - this goes commonly under the term **line-coding**

2) Using clearly longer pulses that are overlapping in time, but designing the pulses such that the above condition is still fulfilled
   - The used bandwidth can be minimized, substantially better spectral efficiency compared to short pulses
   - However, the implementation is more complex (e.g., symbol synchronization in the receiver, see next page example)
   - This is called as **Nyquist pulse shaping** and it is a very essential element in developing modern communications systems.

We illustrate these two principles next with a simple example.
PAM principle and pulse shapes (cont’d)

Example: Illustration of PAM signal composed of individual pulses using the rectangular pulse and a little bit smoother (and longer) pulse.

- Short rectangular pulse:

- Smoother and longer pulse:
PAM principle and pulse shapes (cont’d)

Additional illustration where the total signals, with two different pulse-shapes but the very same symbol stream, are overlaid:

\[ x(t) = \sum_{k} a_k p(t - kT) = ... + a_0 p(t) + a_1 p(t - T) + ... \]

A note about terminology, and “amplitude” in particular

In this context, it is common that the term “amplitude” can include both positive and negative values
- like in our previous examples

On the other hand, in the context of bandpass signals, the concept of amplitude (and envelope) are always understood as positive-only quantities
- this commonly causes some confusion to many people
- … and I also agree that it is a little confusing, in particular when we soon combine carrier modulation on top of the baseband waveform
- so I’ll make complementary notes about this issue in that context, again
- this is, however, the common (slightly confusing) terminology that is commonly used, in literature and engineering work
Spectral contents of baseband PAM signals

In general, the spectral contents of a digital PAM signal of the form

\[ x(t) = \sum_k a_k p(t - kT) \]

depends on the symbol sequence properties and the used pulse shape (quite obvious, I guess).

Now, let’s assume that

- the symbol sequence \( a_k \) is a discrete random signal with power spectral density \( G_a(e^{j2\pi fT}) \)
- the Fourier transform of the used pulse shape is \( P(f) \)

In this case, the power spectral density of a digital PAM signal can be expressed as

\[ G_x(f) = \frac{1}{T} |P(f)|^2 G_a(e^{j2\pi fT}) \]

Since the PAM signal contains both discrete time and continuous-time elements, the mathematical derivation of the above result is not exactly trivial (but not that difficult either, try it for fun).

On the other hand, the result is quite intuitive from the transmit filter interpretation point of view that will be discussed later on

- PAM signal generation by feeding the symbol sequence into a transmitter filter whose impulse response is the pulse-shape \( p(t) \)
- more on this later

Spectral contents of baseband PAM signals (cont’d)

In general, one key requirement is that the spectrum of the transmitted signal must always be fitted to the properties of the channel.

Thus, the above result offers two different tools to control the PAM signal spectrum, i.e.

- \( G_a(e^{j2\pi fT}) \), controlling the symbol sequence spectrum
- \( P(f) \), optimizing the pulse spectrum
Example spectra of two different pulses

- Spectrum of a square pulse of length $T$
  ![Pulse Spectrum](image1.png)

- Spectrum of a longer and smoother Nyquist pulse
  ![Pulse Spectrum](image2.png)

Line coding vs. Nyquist pulse shaping

Based on the previous discussions, there are two alternative ways to create a digital PAM signal and shape its spectral contents:

1. **Line coding**
   - Rectangular pulses, or some simple variations, are used to produce the continuous-time waveform
     - A broad sinc type spectrum (see previous page)
   - DC level can be removed by building the signal appropriately
   - Controlled correlation is created into the symbol sequence, in bit to symbol mapping, in order to shape the power spectrum
   - Often used with binary transmission and simple systems with focus on very simple hardware solutions
   - In pure line coding, optimizing the bandwidth efficiency is not the main objective

2. **Nyquist pulse shaping**
   - Basically assumes that consecutive symbols are uncorrelated
     - The power spectrum of the transmitted signal is fully defined by the Fourier transform of the pulse shape
   - The pulse shape is optimized so that the signal bandwidth is minimized
     - Consecutive pulses are overlapping in time (pulse duration is typically in the order of 3-10 symbol periods)
     - Taking the Nyquist criterion into account in the optimization

In modern systems, the Nyquist pulse shaping is much more common, since bandwidth and spectral efficiency optimization is a key concern.

Nevertheless, a good communications engineer should also have the basic knowledge about line coding basics.
LINE CODING  (... only briefly)

Objectives of line coding

- Elementary transmit signal spectrum control and shaping, simple hardware implementation

- Removing so called baseline wander (DC level variations) in AC coupled wireline systems

- Avoiding symbol timing synchronization difficulties in case that there are long sequences of ‘1’s or ‘0’s in the transmission.

- Monitoring the system behavior during the normal operation is basically possible by using appropriate line codes (monitoring for line code “violations” at the receiving end).

Classification and terminology of line codes

Several different line codes have been introduced and developed over the years. The used terminology also varies somewhat.

In the following, we consider merely the binary line coding case, that is one pulse is always only carrying one source bit.

( More elaborate higher order symbol mapping cases are then considered in the context of Nyquist pulse-shaping. )

Binary line codes can be classified depending on the used signal levels as follows:

<table>
<thead>
<tr>
<th>Possible symbol values</th>
</tr>
</thead>
<tbody>
<tr>
<td>unipolar:</td>
</tr>
<tr>
<td>+a, 0</td>
</tr>
<tr>
<td>polar (or antipodal):</td>
</tr>
<tr>
<td>+a, -a</td>
</tr>
<tr>
<td>bipolar (or pseudoternary):</td>
</tr>
<tr>
<td>+a, 0, -a</td>
</tr>
</tbody>
</table>
Examples of different line codes

<table>
<thead>
<tr>
<th>Bit stream</th>
<th>1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unipolar NRZ-L (On-Off signaling)</td>
<td>+V</td>
</tr>
<tr>
<td>Bipolar NRZ-L</td>
<td>0</td>
</tr>
<tr>
<td>NRZ-M</td>
<td>-V</td>
</tr>
<tr>
<td>NRZ-S</td>
<td>0</td>
</tr>
<tr>
<td>RZ</td>
<td>0</td>
</tr>
<tr>
<td>AMI</td>
<td>0</td>
</tr>
<tr>
<td>Biphase-L (Manchester)</td>
<td>-V</td>
</tr>
<tr>
<td>Biphase-M</td>
<td>+V</td>
</tr>
<tr>
<td>Biphase-S</td>
<td>-V</td>
</tr>
<tr>
<td>CMI</td>
<td>+V</td>
</tr>
<tr>
<td>HD83</td>
<td>V</td>
</tr>
<tr>
<td>3B4B</td>
<td>+V</td>
</tr>
</tbody>
</table>

Binary antipodal codes

During each symbol period, one of the following pulse shapes is transmitted (as is, if the transmitted bit is ‘1’, or as negative, if the transmitted bit is ‘0’...or vice versa)

![Pulse shapes](image)

If return-to-zero, RZ, or non-return-to-zero, NRZ, pulses are to be used in AC coupled channel, it must be taken care of that there are equal amount of positive and negative valued pulses.

With biphase code (=Manchester code) the DC level of the pulse is zero. As a result, also the DC level of the transmitted signal is zero.

- A zero crossing appears in the middle of each symbol period (helps in synchronization).

- The required bandwidth is larger (about two times) compared to the basic NRZ code.

  - This can be seen so that the biphase signaling is achieved by exploiting the NRZ pulse shape and
    - using a double symbol rate, and after each bit, transmitting the bit’s complement or...
    - performing elementary carrier modulation with a rectangular wave with double frequency (compared to symbol rate)

Such bi-phase code is simple and adequate in many cases where hardware and design simplicity is emphasized.
**Twinned Binary Code, TBC, line code**

TBC is a bipolar or pseudo-ternary line code, with a symbol alphabet \{-1, 0, +1\}, where the transmit symbol \(a_k\) is formed as a direct difference between two neighboring bits, \(b_k\) and \(b_{k-1}\), as:

\[
a_k = b_k - b_{k-1}
\]

This allows for elementary transmit signal spectrum shaping, through \(G_B(e^{j2\pi fT})\), as depicted below.

Because the transformation \(a_k = b_k - b_{k-1}\) is basically linear filtering in the bit domain, the spectral density of the transmit symbols reads

\[
G_A(e^{j2\pi fT}) = G_B(e^{j2\pi fT})\left|1 - e^{-j2\pi fT}\right|^2 = 4G_B(e^{j2\pi fT})\sin^2\left(\pi fT\right)
\]

where \(G_B(e^{j2\pi fT})\) is the spectral density of the original bit sequence.

Thus, \(G_A(e^{j2\pi fT})\) contains an additional shaping through \(4\sin^2\left(\pi fT\right)\)
which implies, e.g., that the spectrum is explicitly zero at zero frequency.

Principal coder and decoder block-diagrams are sketched below (here \(D\) corresponds to a delay of one bit duration, while the ternary slicer has decision regions of \(\pm1/2\)).

\[
\begin{array}{c}
b_k \\
\uparrow \\
\downarrow D \\
|| \\
\uparrow \\
\downarrow a_k \\
\uparrow \\
\downarrow \text{TERNARY SLICER} \\
\uparrow \\
\downarrow \hat{b}_k \\
\end{array}
\]

**Example**

Input: 0 1 1 1 1 1 1 1 0 0 0 0 0 0
Output: 1 0 0 0 0 0 0 -1 0 0 0

---

**Alternate Mark Inversion, AMI, line code**

The previous TBC code is eventually not very practical, as errors would accumulate badly (why?).

Alternate Mark Inversion (AMI) line code is another example of bipolar or pseudo-ternary codes, which can be seen as a further evolution from the previous TBC code such that the error accumulation problem is substantially reduced

- hence, there is similar spectral shaping characteristics as in the TBD code; zero and zero frequency

The line coding rule for the AMI code can be simply expressed as:

\[
\begin{align*}
0 & \rightarrow 0 \\
1 & \rightarrow +1 \text{ / } -1 \text{ (in alternating manner)}
\end{align*}
\]

**Example** (below +1's and -1's shown as just +'s and –'s , for simplicity):

Binary input: 0 1 1 1 1 1 1 0 1 0 1
AMI coded: 0 + - + - + - 0 + 0 -

If the basic NRZ square pulse is then adopted, a long sequence of '1's is now observed as an alternating square wave.

AMI code thus removes the synchronization problems introduced by the long sequence of '1's but not the synchronization problem caused by long sequences of '0's.

In more advanced line codes also this can be avoided. A good example is the block line codes described in the following page (e.g., HDB3).

Notice that the AMI line coding principle allows for some elementary system monitoring or error detection

- if the receiver observes two consecutive +'s or –'s, something has gone wrong since such cases are not allowed in the AMI code
**Block line codes**

In this type of a code, blocks of \( k \) bits are converted into blocks of \( n \) symbols that are then transmitted into the channel with NRZ or the like pulses. The size of the alphabet is \( L \).

To ensure that for every input block there is a unique line code block it must hold that

\[
2^k \leq L^n
\]

One starting point for the implementation is the AMI code, in which we add some symbol timing related additional structure into the long zero sequences to make the synchronization easier.

**The fundamental idea:** If the AMI coded block includes only zeros, then instead, we transmit some three level sequence with the same length which

- includes one or more + and/or - symbols to assist synchronization
- is a **forbidden sequence in the AMI code**, and therefore, **can be recognized in the decoder and converted back into the zero sequence**

For instance, HDB3 code, commonly used in classical PCM systems, is based on this principle

- Block of four zeros (0000) is replaced with a block B00V or 000V where
  - B is a + or - symbol fulfilling the AMI criterion
  - V is a + or - symbol not fulfilling the AMI criterion (violation)

Another example is the B6ZS code

- Six zero block (000000) is replaced with block B0BV0V

---

**BASEBAND DIGITAL COMMUNICATION BASED ON NYQUIST PULSE SHAPING**

**Coder or Bit-to-Symbol mapper**

Converts the incoming bit stream into a symbol sequence where the symbols are taken from a specific symbol alphabet. Some examples:

1) Binary alphabet, the symbol sequence is basically the same as the bit sequence or the alphabet is say \( \{-1, +1\} \)

2) Two consecutive bits are described using the alphabet \( \{-3, -1, +1, +3\} \) (see also the earlier examples)

For an alphabet size \( M \), the bit rate and symbol rate are always related through:

\[
R_{bit} = \log_2 (M) \times R_{sym}
\]

In general, the coder may also introduce redundancy, for error control purposes (channel coding, error control coding), which we will address extensively in the later part of this course.
Transmit filter and transmit signal

Transmit filter

Converts the discrete symbol sequence into a continuous-time signal. The impulse response of this filter $g(t)$ is now the transmitted pulse shape.

Thus, the transmitted signal is given by

$$S(t) = \sum_{m=-\infty}^{\infty} A_m g(t - mT)$$

where $A_m$ is the transmitted symbol at time instant $mT$, and $R_{sym} = 1 / T$ is the symbol rate.

Consequently, the waveform is composed of pulses scaled with the symbol values.

In case of Nyquist pulse-shaping, the pulses are long and consecutive pulses overlap in time.

![Symbol Stream](image1)

Channel and received signal

Channel

Typically modeled with a linear filter (modeling linear distortion) with impulse response $b(t)$ and additive noise $N(t)$ (usually normally distributed, especially thermal noise)

- In mobile systems, the channel also varies as a function of time, and therefore, also the filter models have to be time-dependent

Received signal

The received signal can now be expressed as

$$R(t) = b(t) * S(t) + N(t)$$

$$= \int_{-\infty}^{\infty} b(\tau)S(t - \tau)d\tau + N(t)$$

$$= \int_{-\infty}^{\infty} b(\tau) \sum_{m=-\infty}^{\infty} A_m g(t - mT - \tau) d\tau + N(t)$$

$$= \sum_{m=-\infty}^{\infty} A_m h(t - mT) + N(t)$$

Here $h(t)$ is the received pulse shape which is the convolution (why?) of the transmit filter and channel impulse responses, written as

$$h(t) = \int_{-\infty}^{\infty} b(\tau)g(t - \tau) d\tau = b(t) * g(t)$$

- in general, the channel always distorts the transmitted pulses
Elements of the receiver

Receiver tasks are generally more challenging than those of the transmitter:
- Detection of transmitted bits reliably from the received noisy and distorted signal

Basic receiver functionalities are explained below.

Receiver filter

There is always a filtering stage in the receiver. The fundamental task of the receiver filter is to:
1) attenuate noise and interference outside the transmission band
   While doing so, it also
2) affects the effective pulse shape (why?)

In general, the signal after the receiver filter with impulse response $f(t)$ is given by

$$Q(t) = f(t) * R(t) = \ldots = \sum_{m=-\infty}^{\infty} A_m p(t - mT) + U(t)$$

where

$$p(t) = g(t) * b(t) * f(t)$$

is the overall pulse shape and $U(t) = f(t) * N(t)$ describes the filtered noise at the output of the receiver filter.

Sampling

In sampling, the continuous-time signal $Q(t)$ after the receiver filter is sampled to create the corresponding discrete time signal $Q_k = Q(kT)$.

In ideal case, the samples are taken at the moments in which the samples correspond best with the transmitted symbols (i.e., the effects of other symbols are minimized).

For this, the symbol timing recovery is essential – especially with Nyquist pulse-shaping where the individual pulses are long.

Symbol timing recovery

Defines the correct timing and sampling time for the received pulses. There are often specific components in the transmitted signal that make the synchronization easier (e.g. periodically repeating known pilot or reference signals).

(For carrier modulated systems, discussed later, also the carrier synchronization is important, in addition to timing synchronization).
Elements of the receiver (cont’d)

Detection (decision making)
In detection, decisions $\hat{A}_k$ about the transmitted symbols $A_k$ are made using the received samples $Q_k$.

These received samples contain always noise (see previous slides), and possibly also distortion due to the channel.

Conventionally, the decisions are made based on some decision thresholds.

Example: Let’s consider a 4-PAM alphabet $\{-3, -1, +1, +3\}$. In this case, assuming that the noise is zero mean, it is intuitive to set the decision thresholds at the middle of the neighboring symbol values:

```
-3  -1  +1  +3
```

Here, we have actually applied the so called minimum distance detection principle which will maximize the probability of correct decisions (and hence minimize the probability of error) with certain assumptions (e.g., noise distribution type).

- With symmetric and zero mean noise distribution (such as zero-mean Gaussian thermal noise), this is the so-called maximum likelihood (ML) detector
- Basically, statistical decision making (detection theory) is a large field of applied mathematics which we will address much more extensively in this course, a little later

Decoding
Maps the detected symbol sequence back into the bit stream according to the used alphabet and bit-to-symbol mapping that was adopted in the transmitter.

If the bit-to-symbol mapping and coding block in the transmitter was introducing redundancy, for error control purposes, the redundancy is utilized by the receiver in the decoding process

- these aspects will be studied extensively in the latter part of the course

Other notes regarding receiver
- in practice, the channel will always distort the pulses observed in the receiver
- due to this, there will be interpulse interference, or intersymbol interference, ISI
- to reduce the effects of the ISI, and channel distortion in general, receivers commonly adopt some form of channel equalization
  - discrete time linear filtering seeking to “undo” the effects of the channel, that is to remove the ISI
  - since in mobile systems the channel is time-variant, also the equalizer needs to be tracking such time-variations; adaptive equalization methods are important
  - such channel equalizer principles and methods will be treated extensively in this course, a little later
Inter-pulse or inter-symbol interference, ISI

Let’s study two consecutive symbols with values \( A_0 = 1, A_1 = 2 \).

The corresponding example pulses and their overlap in time is illustrated in the figure:

\[
a_0 g(t) + a_1 g(t - T)
\]

\[
a_0 g(t)
\]

\[
a_1 g(t - T)
\]

In the receiver, the sampling is performed at relative time instants \( t = mT \).

Now, if the pulses after the receiver filter obey the Nyquist criterion, consecutive pulses do not interfere with each other, and therefore, the inter-symbol interference (ISI) is zero.

It is worth noticing that this will be achieved only if the sampling synchronization (symbol timing recovery) is perfectly acquired and that the receiver filter output pulse-shape \( p(t) \) truly is a Nyquist pulse

- \( p(t) = g(t) \ast b(t) \ast f(t) \)
- thus, no matter how beautiful pulse is adopted in the transmitter, receiver always observes a pulse that has been shaped by the channel and the RX filter
- we will discuss this issue soon in more details, together with the connection between the pulse bandwidth and the symbol rate

Overall received signal and system model

Based on the previous, the overall received signal reads

\[
Q(t) = f(t) \ast R(t) = \ldots = \sum_{m=-\infty}^{\infty} A_m p(t - mT) + U(t)
\]

where

\[
p(t) = g(t) \ast b(t) \ast f(t)
\]

is the overall pulse shape and \( U(t) = f(t) \ast N(t) \) describes the filtered noise at the output of the receiver filter.

Assuming that the actual channel noise \( N(t) \) is white noise with spectral density \( N_0 \), the spectral density of the filtered noise reads (why?)

\[
S_U(f) = N_0 |F(f)|^2
\]

- if the actual channel noise \( N(t) \) is Gaussian distributed, also the filtered noise \( U(t) = f(t) \ast N(t) \) is Gaussian distributed while the above spectral density holds independently of the distribution
Overall received signal and system model, cont’d

When combined with symbol rate sampling, the received symbol rate samples read

\[
Q_k = Q(kT) = \sum_m A_m p(kT - mT) + U(kT) \\
= \sum_m A_m p_{k-m} + U_k \\
= A_k * p_k + U_k
\]

In the previous model, \( U_k \) describes the noise samples while \( p_k = p(kT) \) denotes the sampled version of the overall pulse shape \( p(t) = g(t) * b(t) * f(t) \).

The spectral density of the noise samples \( U_k \) reads

\[
S_U(e^{2\pi fT}) = N_0 \frac{1}{T} \sum_{m=-\infty}^{\infty} \left| F\left(f - \frac{m}{T}\right) \right|^2
\]

Now, in general, the total pulse shape \( p(t) = g(t) * b(t) * f(t) \) is not a Nyquist pulse, due to the channel distortion, and thus the values \( p_k = p(kT) \) for \( k \neq 0 \) are non-zero which corresponds to ISI.

To see this more clearly, we simply write out the above expression as

\[
Q_k = \sum_m A_m p_{k-m} + U_k \\
= ... + A_{k+1} p_{-1} + A_k p_0 + A_{k-1} p_1 + ... + U_k
\]

- the ISI strength depends on the values of \( p_k = p(kT), \ k \neq 0 \) compared to \( p_0 \)

Hence, the overall baseband PAM system can be essential modeled through the following discrete-time equivalent system model of the form

\[
\text{BITS} \rightarrow \text{CODER} \rightarrow \text{DISCRETE-TIME EQUIVALENT CHANNEL} \rightarrow \text{DECODER} \rightarrow \text{BITS}
\]

where

- the sampled total effective pulse shape \( p_k = p(kT) \) is commonly called the discrete-time equivalent channel, and also many times the ISI profile since it describes the structure of the ISI in the system
- the discrete time noise samples \( U_k \) are Gaussian distributed (assuming the physical channel noise is Gaussian) with spectral density

\[
S_U(e^{2\pi fT}) = N_0 \frac{1}{T} \sum_{m=-\infty}^{\infty} \left| F\left(f - \frac{m}{T}\right) \right|^2
\]

- \( N_0 \) is the spectral density of the physical channel noise, assumed white noise
- \( F(f) \) denotes the frequency response of the receiver filter

We will use this model in many of the following further developments, particularly

- when deriving optimal decision rules for making reliable decisions about the transmitted symbols in the receiver under noise
- when analyzing the symbol error probability and bit error probability of different symbol alphabets under noise
- when deriving different receiver signal processing mechanisms, primarily channel equalizers and sequence detector, to mitigate or deal with the ISI
PULSE SHAPE AND BANDWIDTH CONSIDERATIONS

The spectrum of the transmitted signal must, in general, be fitted to the available channel bandwidth $W$. This can be modeled by assuming the following perfectly band-limited channel of the form:

$$B(f) = \begin{cases} 1 & |f| < W \\ 0 & |f| \geq W \end{cases}$$

A transmit pulse that would fit to this frequency window perfectly has a similar rectangular spectrum of the form:

$$G(f) = \begin{cases} \frac{1}{2W} & |f| < W \\ 0 & |f| \geq W \end{cases}$$

The corresponding pulse shape in time domain is

$$g(t) = \frac{\sin(2\pi W t)}{2\pi W t} = \text{sinc}(2\pi W t)$$

Consequently, the ideal pulse is a sinc-function with zero crossings at the integer multiples of $1/(2W)$

- Now, if the symbol period is $T = 1/(2W)$, then there is no interaction or interference between the neighboring pulses, although the pulses will be considerably overlapping in time.

From the condition $T = 1/(2W)$, it then follows that

$$W = \frac{1}{2T}$$

- For a given symbol rate $1/T$, this is also the minimum bandwidth which enables to avoid the intersymbol interference
- Very important result; will be proven in the next few pages

Pulse-shape filter design

The previous sinc-pulse illustrates the principle well, and is also a theoretical minimum bandwidth pulse, but it is not a practical solution (due to its infinite length and slowly decaying shape). Hence, we will discuss next the pulse-shape filter design from more practical perspective.

Time-domain requirements

Requirement: inter-symbol interference (ISI) is zero

$$p(0) = 1$$

$$p(mT) = 0, \ m = \pm1, \pm2,...$$

The shape of the pulse is otherwise free.

Frequency-domain requirements

The ideal bandlimited sinc-pulse is not a practical solution. Practical pulses include typically 0-100% excess bandwidth compared to the sinc-pulse. Consequently, the overall bandwidth is (see also next page)

$$W = \frac{1}{2T} + \alpha \frac{1}{2T}$$

where $\alpha = 0.1, ... , 1$ is the so called roll-off factor.

In practice, the pulse shape filter $P(f)$ has a symmetric transition band with respect to the theoretical minimum bandwidth $1/(2T)$, see next pages for a proof of this.
Nyquist criterion in frequency domain for zero ISI

From the previously presented time domain criterion it directly follows that (notice the close relation to the sampling theorem):

\[
\sum_{k=-\infty}^{\infty} p(kT) \delta(t - kT) = \delta(t), \quad \text{i.e.,}
\]

\[
p(t) \sum_{k=-\infty}^{\infty} \delta(t - kT) = \delta(t)
\]

The corresponding frequency-domain criterion, obtained through Fourier transform, can be expressed as

\[
P(f) * \frac{1}{T} \sum_{m=-\infty}^{\infty} \delta(f - m \frac{1}{T}) = 1, \quad \text{i.e.,}
\]

\[
\frac{1}{T} \sum_{m=-\infty}^{\infty} P(f - m \frac{1}{T}) = 1
\]

The final expression is the famous Nyquist criterion in frequency-domain to avoid the inter-symbol interference. Now, it follows that:

- The lowest bandwidth for reaching 0-ISI is indeed \( W = 1/2T \). This can be achieved with the previously showed ideal bandlimited pulse. Hence, the time domain pulse shape is a sinc-function.
- Bandlimited pulse shape filters have symmetric transition bands as sketched already on the previous page.

Pulses/filters that fulfill the Nyquist criterion are called Nyquist pulses/filters. Some arbitrary examples in frequency domain are:

Graphical illustrations of the obtained requirement given below:

(a) Here pulse bandwidth \( W < 1/2T \) (i.e., \( 1/T > 2W \)); thus the pulse cannot fulfill the Nyquist criterion, no matter what is the exact shape of the pulse (or its spectrum).

(b) Here pulse bandwidth \( W = 1/2T \) (i.e., \( 1/T = 2W \)); Nyquist criterion is fulfilled only if the pulse is the sinc pulse (whose rectangular spectrum is assumed in the figure). Thus, the sinc pulse is the unique minimum bandwidth pulse.

(c) Here pulse bandwidth \( W > 1/2T \) (i.e., \( 1/T < 2W \)); Nyquist criteria can be fulfilled, as long as the pulse spectrum is symmetric relative to \( 1/2T \) (like in the figure). There are generally an infinite amount of different pulses like this, with one practical example being the so-called raised-cosine pulses (more on them soon).
Pulse shape filtering in baseband PAM system

In general, avoiding inter-symbol interference is essential in the sampling process in the receiver.

The observed pulse shape \( p(t) = g(t) * b(t) * f(t) \) depends on the transmitter filter \( g(t) \), the receiver filter \( f(t) \), and the channel \( b(t) \). Now the corresponding transfer function is \( P(f) = G(f)B(f)F(f) \).

The objective is that this cascade of the three transfer functions, i.e.,

\[
P(f) = G(f)B(f)F(f)
\]

fulfils the Nyquist criterion.

The transfer function of the channel is usually unknown and it cannot be affected by the system designer.

Transmit and receive filters, on the other hand, are subject to the system designer. Thus, the following two solutions are commonly available:

1) Pulse shaping mostly in the transmitter, while the receiver filter then approximates an ideal lowpass filter whose bandwidth is \( (1 + \alpha)/2T \).

2) Pulse shaping is splitted equally between the transmitter and receiver, in the form of so-called square-root Nyquist filters (concrete examples a little later). This is performance wise somewhat better option compared to 1) since the receiver filter attenuates noise more efficiently (details will come a little later in this course).

In both cases, it is only the cascade of TX and RX filters, that is \( G(f)F(f) \), that fulfills the Nyquist criterion

- Thus, for practical frequency selective channel response \( B(f) \), there is ISI in the receiver since \( P(f) = G(f)B(f)F(f) \) does not fulfill the Nyquist criterion
- This will be then compensated for by a discrete-time channel equalizer

Raised-cosine pulses

One famous and widely applied class of pulse-shape filters is the so called raised-cosine pulses.

The idea in these pulses is to control the effective length of an ideal sinc-pulse using a window function.

Here the effective pulse duration is decreased by increasing the excess bandwidth \( \alpha \). In this case the oscillation of the pulse attenuates faster but the bandwidth increases.

\[
p(t) = \frac{\sin(\pi t / T)}{\pi t / T} \left[ \frac{\cos(\alpha \pi t / T)}{1 - (2\alpha \pi t / T)^2} \right] = \text{sinc}(t / T) \left[ \frac{\cos(\alpha \pi t / T)}{1 - (2\alpha \pi t / T)^2} \right]
\]

The Fourier transform of the pulse is:

\[
P(f) = \begin{cases} 
  T & ; |f| \leq \frac{1 \small{-} \alpha}{2T} \\
  \frac{T}{2} \left\{ 1 \small{-} \cos \left( \frac{\pi T}{\alpha \left( |f| \small{-} \frac{1 \small{+} \alpha}{2T} \right) \right) \right\} & ; \frac{1 \small{-} \alpha}{2T} \leq |f| \leq \frac{1 \small{+} \alpha}{2T} \\
  0 & ; |f| \geq \frac{1 \small{+} \alpha}{2T} 
\end{cases}
\]

\[
p(t) = \frac{\sin(\pi t / T)}{\pi t / T} \left[ \frac{\cos(\alpha \pi t / T)}{1 - (2\alpha \pi t / T)^2} \right] = \text{sinc}(t / T) \left[ \frac{\cos(\alpha \pi t / T)}{1 - (2\alpha \pi t / T)^2} \right]
\]

The Fourier transform of the pulse is:

\[
P(f) = \begin{cases} 
  T & ; |f| \leq \frac{1 \small{-} \alpha}{2T} \\
  \frac{T}{2} \left\{ 1 \small{-} \cos \left( \frac{\pi T}{\alpha \left( |f| \small{-} \frac{1 \small{+} \alpha}{2T} \right) \right) \right\} & ; \frac{1 \small{-} \alpha}{2T} \leq |f| \leq \frac{1 \small{+} \alpha}{2T} \\
  0 & ; |f| \geq \frac{1 \small{+} \alpha}{2T} 
\end{cases}
\]
Raised-cosine pulses, cont’d

Examples of raised cosine pulse spectra with 3 different roll-off factors $\alpha = 1$, $\alpha = 0.5$, $\alpha = 0.1$ (i.e., 100%, 50%, 10% excess bandwidth).

Square-root raised-cosine pulses

Many of the Nyquist filters, also the raised-cosine filters, have also the so called square root Nyquist versions

- Here the idea is that an individual pulse does not fulfill the Nyquist criterion but two consecutive filters in cascade do.
- I.e., one square root filter in the transmitter and one in the receiver, like was discussed on the previous pages.

Expressions:

$$p_s(t) = \frac{4\alpha \frac{t}{T} \cos([1 + \alpha] \pi \frac{t}{T}) + \sin([1 - \alpha] \pi \frac{t}{T})}{\pi \frac{t}{T} [1 - (4\alpha \frac{t}{T})^2]}$$

$$P_s(f) = \begin{cases} \sqrt{T} & ; |f| \leq \frac{1 - \alpha}{2T} \\ \frac{T}{2} \left(1 - \cos \left(\frac{\pi T}{\alpha} \left|\frac{f}{2T} + \frac{1 + \alpha}{2T}\right|\right)\right) & ; \frac{1 - \alpha}{2T} \leq |f| \leq \frac{1 + \alpha}{2T} \\ 0 & ; |f| \geq \frac{1 + \alpha}{2T} \end{cases}$$

Design and analysis in Matlab, e.g., using the “rcosdesign” function

- for both ordinary and square-root raised cosine pulses
- sidenote: Matlab’s built-in “rcosdesign” function is not very well optimized for creation of square-root pulses, especially if the pulse length is relatively short, such that two of such filters in cascade really yield a Nyquist pulse. Hence, you may need to use a relatively long pulse when experimenting with it.
Example of square-root raised-cosine pulses

Achievable bit rate

For a given bandwidth

\[ W = (1 + \alpha) \frac{1}{2T} \]

and symbol alphabet size \( M \), the physical layer core bit rate reads

\[ R_{bit} = \log_2(M) \times \frac{1}{T} \]
\[ = \log_2(M) \times \frac{2W}{1 + \alpha} \]

Important observation is that for a given bandwidth \( W \)

- bit rate can be increased by increasing the symbol alphabet size
  - larger alphabets are, however, more sensitive to noise, distortion and interference, so there are no free lunches in this respect
  - we will analyze explicitly the sensitivity of different symbol alphabets against noise, a little later
- higher values of roll-off \( \alpha \) decrease the bit rate

In most practical systems, the roll-off factor is in the order of 20-40%

- example: baseband waveform pulse-shape filtering in the WCDMA/UMTS/HSPA mobile networks done with 22% roll-off
Achievable bit rate and comparison to Shannon capacity

As we discussed earlier in the course, the theoretical maximum information rate (the Shannon capacity) of any communication system is

\[ C = W \log_2(1 + SNR) \]

where \( W \) is the bandwidth and SNR refers to the signal to noise ratio.

On the other hand, for baseband PAM, we just established that the bit rate reads (zero excess bandwidth, \( \alpha = 0 \), assumed below, for simplicity)

\[ R_{\text{bit}} = \log_2(M) \times 2W = W \log_2(M^2) \]

Thus, it can be deduced that

- the bandwidth \( W \) impacts both expressions in a similar way
  - large bandwidth implies larger bit rate
- the modulation (alphabet) size \( M \) is implicitly linked to SNR/SINR
  - larger modulations need larger SNR (we will establish this also through the error rate analysis, soon)
  - in principle there is a limit of the form
    \[ M^2 \leq 1 + SNR \quad \Rightarrow \quad M \leq \sqrt{1 + SNR} \]
- engineering conclusion is the following: larger modulation implies larger bit rate as long as the SNR is at proper range for the particular modulation such that RX can reliable decode the signal

Discrete-time pulse-shape filtering

In practice, the pulse-shape filtering is carried out using discrete-time / digital filtering, in both TX and RX sides

This means that

- at TX, the sample rate must be increased in the process (which is in general called interpolation)
- at RX, the signal is sampled at higher than symbol rate

The principle is illustrated below (at TX side as a concrete example), while also an example of a 4x oversampled discrete-time square-root raised cosine filter is shown (both impulse and amplitude responses)
Shortly on symbol rate sampling vs. aliasing

Like we have now learnt, the physical bandwidth of a baseband PAM signal is generally of the form

\[ W = (1 + \alpha) \frac{1}{2T} \]

Thus, if thinking the classical Nyquist sampling theorem, the minimum sample rate for such signals should be

\[ (f_s)_{\text{min}} = 2W = (1 + \alpha) \frac{1}{T} \]

However, in our basic PAM receiver concepts, we are assuming symbol rate sampling, i.e., \( f_s = 1/T \)

Thus, the obvious questions are:

- What the heck, did something go wrong? Is there aliasing involved? What's going on here?

The answers are:

- No need to worry, everything is good :).

- That is, the symbol rate sampling in PAM receivers is only seeking to obtain information about the amplitudes of the individual pulses – not, per se, to discretize the whole waveform such that there would not be e.g. aliasing (which is the basic idea behind sampling theorem)

- Thus, assuming that \( \alpha > 0 \), there is indeed aliasing involved in symbol rate sampling since \((1 + \alpha)(1/T) > 1/T\) but this is not relevant as we are only looking after the amplitudes of the individual pulses

Shortly on symbol rate sampling vs. aliasing, cont'd

On the other hand, it is also good to acknowledge that the previous reasoning assumes ideal symbol timing synchronization

This may not always be realistic in practice, and thus in general, it is useful to sample the received signal at a rate larger than the symbol rate

- By sampling e.g. at a rate \( f_s = 2 \times (1/T) \) (2 samples per pulse/symbol) or, \( f_s = 4 \times (1/T) \) (4 samples per pulse/symbol), we avoid all aliasing and thus the symbol timing adjustment can be done digitally
  - through appropriate fractional delay filtering or other digital interpolation mechanisms

- With such higher sample rate, then also the actual pulse shape filtering can be done digitally (like we already concluded a few slides back), which applies to both TX and RX
Dealing with ISI in practice – a glimpse

In practice, even if beautiful square-root Nyquist pulse-shaping filters are deployed in transmitter and receiver, the channel is anyway distorting the pulses (like already discussed in modeled in greater details)

- Causes intersymbol interference in reality

There are two alternative solutions to tackle this in the receiver

- **Channel equalizers**
  - At symbol level, ISI acts as linear discrete-time filtering
    => use discrete-time inverse filtering to remove the ISI
  - Both linear and nonlinear (e.g. decision feedback) equalization solutions exists
  - Also different equalizer optimization criteria, e.g. zero-forcing and minimum mean-squared error

- **Sequence detection**
  - Instead of removing the ISI explicitly with discrete-time filtering, take it into account in the detection
  - Means essentially fitting all the possible transmit sequences, with ISI included, to the observed sequence
  - Fairly complex compared to equalizers
  - Practical implementations though e.g. Viterbi algorithm

- We will treat both of these techniques excessively in this course

Eye diagram

An eye diagram consists of several overlapping symbol level synchronized waveform traces.

It is also assumed that symbols are random and independent, in which case all possible symbol combinations are present.

Eye diagrams can be measured with an oscilloscope or simulated using computers. They are useful when analyzing the system functionality during the development phase. For example, ISI and sensitivity to timing synchronization errors can be seen in the eye diagram.

Examples on the next pages illustrate the concept well.

Properties:

- ISI decreases the vertical eye opening

- The greater is the vertical eye opening, the more immune the system is to noise (noise would be seen as random fluctuation on top of the curves)

- The ideal sample time is when the vertical opening has its largest value

- The smaller is the horizontal opening, the more sensitive the system is for timing synchronization
Eye diagrams

The effect of the excess bandwidth (raised-cosine pulses, 2-level PAM) is illustrated well in the eye-diagram:

It can be seen that increasing the excess bandwidth improves the eye diagram by making the horizontal eye opening larger. In the extreme case (0% excess band, i.e., the sinc-pulse) the horizontal eye opening goes practically to zero

- Thus, the sinc-pulse is infinitely sensitive to timing errors, while large excess bandwidths correspond to good robustness against reasonable timing errors

In the below, another example with 4-level PAM with 25% excess bandwidth and raised-cosine pulse-shape is given:
4. SINGLE-CARRIER LINEAR DIGITAL I/Q MODULATION

In carrier modulation, in general, the baseband signal is upconverted to around a desired carrier frequency.

Or from digital communications point of view:

In carrier-modulated digital communications, we deploy bit-carrying waveforms whose spectral content is located in the desired part of the radio spectrum.

Contents in this part:
- Complex quadrature (I/Q) modulation and complex constellations: QAM, PSK
- Relation of bandwidth, alphabet size and bit rate in carrier modulated case
- Discrete-time baseband equivalent model of the overall system

Complex (I/Q) modulation

Many of the carrier modulation methods in digital communication are based on

- Utilization of a complex symbol alphabet
- Complex quadrature modulation (I/Q modulation)

There are several alternative complex alphabets available (QAM, PSK, ...). First, we study the principles at general level where the exact structure of the alphabet is not yet essential.

The quadrature I/Q modulation is based on the fact that sine and cosine waves can in principle be modulated independently using two parallel baseband waveforms.

The beef is thus:

- The carrier modulator input includes a continuous complex-valued baseband signal with appropriate Nyquist pulse shaping deployed according to the previous discussion
  - allows for the use of complex-valued symbol alphabets
- Real and imaginary parts of the signal are then used to modulate cosine and sine waves.
- NB: The purpose of the pulse shape filtering is the same as earlier: bandlimit the baseband waveform

Recap:

\[ x_{bp}(t) = A(t) \cos(\omega_c t + \phi(t)) \]
\[ = x_i(t) \cos(\omega_c t) - x_q(t) \sin(\omega_c t) \]
\[ = \text{Re}[x_{bp}(t)e^{j\omega_c t}] \]

where \( x_{bp}(t) = x_i(t) + jx_q(t) \).
Complex (I/Q) modulation

In practice, \( g(t) \), the transmit filter, is always a real-valued pulse, from which it follows that:

\[
x(t) = \sqrt{2} \cos(\omega_c t) \sum_{m=-\infty}^{\infty} \text{Re}[a_m] g(t - mT)
\]

\[
-\sqrt{2} \sin(\omega_c t) \sum_{m=-\infty}^{\infty} \text{Im}[a_m] g(t - mT)
\]

Hence, the real and imaginary parts of the baseband signal modulate the cosine and sine components of the carrier.

Although this is the practical implementation principle, also the complex notation is used a lot in the continuation.

NB1: The term bandpass PAM/PSK/QAM is quite often used in case of the I/Q modulation.

NB2: In more implementation oriented studies, also the term Cartesian transmitter is many times used (note also that e.g. the PA is excluded in above figure).
Bandpass PAM/PSK/QAM receiver

Two equivalent structures for demodulation:

1) Based on complex signals:

\[ y(t) \to e^{-j\alpha t} \to \text{FILTER} \sqrt{2} f(t) \to \text{SAMPLER} q(t) \to \text{SLICER} \to \text{DECODER} \to \text{BITS} \]

2) Based on parallel real-valued signals:

\[ y(t) \to \cos(\alpha t) \\text{IN-PHASE SIGNAL} \to \text{FILTER} \sqrt{2} f(t) \to \text{SAMPLER} \to \text{SLICER} \to \text{DECODER} \to \text{BITS} \]

\[ y(t) \to -\sin(\alpha t) \\text{QUADRATURE SIGNAL} \to \text{FILTER} \sqrt{2} f(t) \to \text{SAMPLER} \to \text{SLICER} \to \text{DECODER} \to \text{BITS} \]

Spectral interpretation using complex signals:

- \[ S(f) \]
- \[ Y(f) = X(f) \]
- \[ R(f) \]
- \[ Q(f) \]

Bandpass PAM/PSK/QAM receiver (cont’d)

To illustrate the whole transmission chain let’s assume for now:

- Ideal channel: no noise, distortion or interference \( y(t) = x(t) \)
- The receiver oscillator is perfectly synchronized with the received signal’s phase and frequency
- The receiver filter \( f(t) \) is a lowpass filter which removes the spectrum components around the frequency \( \pm 2f_c \). In addition, it contributes to the pulse shape and removes noise and interference outside the used frequency band.
- Sampling at symbol rate with ideal timing synchronization
- The overall pulse shape experienced at the output of the receiver filters fulfills the Nyquist criterion, i.e., no inter-symbol interference.
  \[ \rightarrow \text{original symbols are observed}, \quad q_k = a_k \]
- ... Complete modeling of the received signal, under noise and frequency-selective channel will follow soon

(Again, coefficients \( \sqrt{2} \) have been included to match the signal levels in different parts of the transmission chain. In practice this means just proper normalization of the signal levels in different stages).

Notice that the input of the slicer is complex valued and it cannot be generally replaced with two real valued slicers (compare to the earlier baseband transmission)

Filter \( f(t) \) removes the spectrum components around the frequency \( \pm 2f_c \), and also, the noise and interference outside the useful signal band (see spectral illustration on previous slide), while also affecting on the overall pulse shape.

Notice that the implementation of practical radio receivers include also aspects like selectivity-filtering, amplification, and tunability, etc… (the whole radio architecture perspective)

- however, to understand and illustrate the transmission of carrier modulated PAM/PSK/QAM signals these are not essential, and therefore, we exploit a simplified model here in this regard
Alternative receiver structures

In the upper structure, filtering is performance prior to the I/Q downconversion, using a complex bandpass filter. This is an alternative way to do I/Q demodulation as a whole.

In the lower structure, also the order of the sampling and I/Q demodulation has been changed. This corresponds to bandpass sub-sampling, at symbol rate, where controlled aliasing is used to bring the signal closer to baseband. Remaining possible frequency offset is then corrected for using digital I/Q downconversion.

Discrete-time pulse-shape filtering, again

Similar to the earlier discussion, again discrete-time pulse-shape filtering can be adopted, at oversampled rate, at both TX and RX

- carried out now in both I and Q branches
Complex Alphabets or Constellations

In I/Q modulated PAM/PSK/QAM, the symbol values are complex numbers.

If we assume that the transmit filter $g(t)$ is real-valued (practically always true) and we denote the complex symbols using their amplitudes and phases as $a_m = r_m e^{j\phi_m}$, the modulated signal can be re-written as

$$x(t) = \sqrt{2} \Re \left\{ \sum_{m=-\infty}^{\infty} a_m g(t - mT) \right\}$$
$$= \sqrt{2} \Re \left\{ \sum_{m=-\infty}^{\infty} e^{j\omega t} r_m e^{j\phi_m} g(t - mT) \right\}$$
$$= \sqrt{2} \sum_{m=-\infty}^{\infty} r_m \cos(\omega t + \phi_m) g(t - mT)$$

Consequently, it is possible to think the I/Q modulated waveform so that

- the symbol amplitude and phase within a certain symbol period defines the amplitude and phase of the modulated carrier
- transitions between consecutive symbol durations depend on pulse shaping (see the example two slides forward)

The utilized complex symbols (the symbol alphabet) can be illustratively presented with a constellation figure, see next page.

Often used constellations are, e.g.,

- Quadrature Amplitude Modulation, QAM: constellation points are distributed on a uniformly spaced grid
- Phase Shift Keying, PSK: constellation points are distributed uniformly on a circle, information is in the modulated signal phase

Also many other constellations are available, but the above are the de-facto solutions adopted in practice.

Example complex constellations

4-PSK (quadrature PSK, QPSK)

- Alphabet size $2^B = 4$, each symbol represents $B = 2$ bits.
- Symbols: $A_m = be^{j\phi_m}$; $\phi_m \in \{0, \pi/2, \pi, 3\pi/2\}$
- Information is in the phase of the carrier-modulated waveform
- 45deg rotated version (compared to fig below) is also commonly adopted; in this case, identical to 4-QAM

16-QAM

- Alphabet size $2^B = 16$, each symbol represents $B = 4$ bits.
- Symbols: $A_m = a_{m,T} + j a_{m,Q}$; $a_{m,T}, a_{m,Q} \in \{\pm c, \pm 3c\}$
- Information is in both the amplitude and phase of the carrier-modulated waveform

More generally:

- $M$-PSK: $2^B$ constellation points uniformly distributed on a circle (equal phase difference between the neighboring points)
- $M$-QAM: $2^B$ square-shape ($2^{B/2} \times 2^{B/2}$) constellation (uniformly spaced “grid”)
Examples of carrier-modulated pulses with 16-QAM

- Individual I/Q modulated pulses (left) and the composite waveform (right); pulse-shaping with square-pulse

Symbol Stream: $-1-3i -1-1i 3+3i 1+3i -3+1i$

Relative Time in Symbol Periods

- Individual I/Q modulated pulses (left) and the composite waveform (right); pulse-shaping with raised-cosine pulse (25% roll-off)

Symbol Stream: $-1-3i -1-1i 3+3i 1+3i -3+1i$

Relative Time in Symbol Periods

16-QAM example, different component signals, square pulse

$$x(t) = \cos(\omega_c t) \sum_{m=\infty}^{\infty} \Re[a_m] g(t - mT) - \sin(\omega_c t) \sum_{m=\infty}^{\infty} \Im[a_m] g(t - mT)$$

$$= \sum_{m=\infty}^{\infty} r_m \cos(\omega_c t + \phi_m) g(t - mT)$$

rectangular pulse $g(t)$

symbol sequence: $-1-3j, -1-j, 3+3j, 1+3j, -3+j$

Baseband Waveform, I branch

Baseband Waveform, Q branch

Carrier Modulated I Component

Carrier Modulated Q Component

Final Carrier Modulated Waveform
**16-QAM example, different component signals, RC pulse**

\[ x(t) = \cos(\omega_c t) \sum_{m=-\infty}^{\infty} \text{Re}[a_m]g(t - mT) - \sin(\omega_c t) \sum_{m=-\infty}^{\infty} \text{Im}[a_m]g(t - mT) \]

\[ = \sum_{m=-\infty}^{\infty} r_m \cos(\omega_c t + \phi_m)g(t - mT) \]

raised-cosine pulse \( g(t) \)

symbol sequence: \(-1 - 3j, -1 - j, 3 + 3j, 1 + 3j, -3 + j\)

---

**Some further notes**

The term Quadrature Amplitude Modulation, QAM, (e.g. the previous 16-QAM whose constellation is on the right), is slightly misleading when thinking about the carrier modulated I and Q components separately.

This is because the symbol levels along the RE and IM axes contain both positive and negative values (e.g. \(-3, -1, +1, +3\)), and thus naturally then also the baseband I and Q waveforms contain also positive and negative values (see previous slide example).

Thus, from the perspective of a general bandpass signal of the form \( A(t) \cos(\omega_c t + \phi(t)) \):

- the carrier modulated I and Q component waveforms, when considered separately, contain actually both amplitude and phase modulation
  - negative values of the baseband I and Q waveforms correspond to 180deg phase shift, relative to the unmodulated \( \cos(\omega_c t) \) and \( \sin(\omega_c t) \) waves
- ... but when finally superimposed, the total carrier modulated signal can be interpreted so that the complex symbol amplitude and phase modulate the carrier amplitude and phase, respectively

Notice that all this discussion and interpretations are valid for arbitrary M-QAM, e.g.

- 16-QAM, 64-QAM, 256-QAM, 1024-QAM, ...
Noisy constellation and detection

Because of noise already, the received samples $Q_k$ are not perfectly matching the constellation points. Now, if we plot noisy samples in the complex domain, the result is commonly of the form:

![Complex constellation](image1)

Generally, the objective in decision making is then to select or decide that specific symbol (from the used alphabet) which the received complex sample $Q_k$ would most likely be representing.

Intuitively the most reasonable selection is the symbol $\hat{A}_k$ which minimizes the distance $|Q_k - \hat{A}_k|$:
- this is again the maximum likelihood, ML, detector which minimizes the error probability (under certain assumptions)
- we will talk about different detectors much more later in this course

This way, decision regions are formed around the constellation points. A particular region includes all those points that are closer to that specific symbol compared to any other symbols in the alphabet.

Here are examples of decision regions for the 4-PSK and 16-QAM:

![Decision regions](image2)

Bit-to-Symbol mapping and Gray code

Also with complex constellations, it is sensible to do the bit to symbol mapping such that the neighboring symbols differ by only one bit:
- if the receiver "mixes up" between different symbols, it is likely to happen between the neighboring ones
- in such cases, even though the symbol is misinterpreted, only a single bit out of $\log_2(M)$ is decoded incorrectly

Examples below, for QPSK and 16QAM:

![Gray code example](image3)
Achievable bit rate and comparison to Shannon capacity, revisited

Again, the theoretical maximum information rate of any system is

\[ C = W \log_2(1 + \text{SNR}) \]

where \( W \) is the bandwidth and SNR refers to the signal to noise ratio.

On the other hand, for I/Q modulated system, we recently established that the bit rate reads (zero excess bandwidth, \( \alpha = 0 \), assumed below, for simplicity)

\[ R_{\text{bit}} = \log_2(M) \times W = W \log_2(M) \]

Thus, it can be again deduced that

- the bandwidth \( W \) impacts both expressions in a similar way
  - large bandwidth implies larger bit rate
- the modulation (alphabet) size \( M \) is implicitly linked to SNR/SINR
  - larger modulations need larger SNR (we will show this explicitly a little later in the course)
  - in principle there is now a limit of the form
\[ M \leq 1 + \text{SNR} \]

- engineering conclusion is, again: larger modulation implies larger bit rate as long as the SNR is at proper range for the particular modulation such that receiver can reliably decode the received signal
- we will explicitly analyze the symbol error rate and bit error rate of different complex alphabets against noise, a little later

Overall received signal and system model

We can next easily derive a model for the continuous-time received baseband signal with the following steps, in the I/Q modulated case:

- I/Q modulated TX signal (recap):
\[ x(t) = \sqrt{2} \Re \left[ e^{-j\omega_c t} \sum_m A_m g(t - mT) \right] \]

- Noisy and distorted received signal at receiver input:
\[ y(t) = b(t) \ast x(t) + n(t) \]
  - \( b(t) \) denotes the bandpass channel impulse response
  - \( n(t) \) denotes additive channel noise, assumed white Gaussian

- I/Q demodulated baseband received signal:
\[ Q(t) = \sqrt{2} f(t) \ast \left( e^{-j\omega_c t} y(t) \right) = \sum_m A_m p(t - mT) + Z(t) \]

where \( p(t) \) denotes the total effective received pulse-shape while \( Z(t) \) denotes downconverted and filtered noise.
Overall received signal and system model, cont’d

In the previous received signal expression, the total received pulse-shape \( p(t) \) is given by

\[
p(t) = g(t) \ast b_{LP}(t) \ast f(t)
\]

where \( g(t) \) and \( f(t) \) denote the TX and RX pulse shape filters while \( b_{LP}(t) \) is the so-called low-pass equivalent or baseband equivalent channel impulse response which is the impulse response corresponding to the frequency selective multipath channel frequency response shifted to baseband

- illustrated below
- because the frequency-response is, in general, not symmetric around zero frequency, this baseband equivalent impulse response, and thus the whole effective received pulse-shape is complex-valued

The downconverted and filtered noise \( Z(t) = Z_i(t) + jZ_q(t) \), in turn, reads

\[
Z(t) = \sqrt{2}f(t) \ast \left( e^{-j\omega_C t}n(t) \right)
\]

Thus, if the physical channel noise \( n(t) \) is white Gaussian noise, with spectral density \( N_0 \), then it is relatively easy to show that the following properties hold for the downconverted and filtered noise \( Z(t) = Z_i(t) + jZ_q(t) \):

- the spectral density of \( Z(t) \) is of the form (why?)
  \[
  S_Z(f) = 2N_0 |F(f)|^2
  \]
- the real and imaginary parts (the I and Q components) of \( Z(t) \) are Gaussian distributed (why?) and they both have the variance (power) of the form
  \[
  \sigma^2 = N_0 \int_{-\infty}^{\infty} |f(t)|^2 dt = N_0 \int_{-\infty}^{\infty} |F(f)|^2 df = E[Z_i(t)]^2 = E[Z_q(t)]^2
  \]
- the corresponding total noise power is \( E[Z(t)]^2 = 2\sigma^2 \)
  - notice that \( \sigma^2 \) is also the noise power in the corresponding baseband PAM system, for reference
- the real and imaginary parts of the noise are also mutually uncorrelated, and thus based on the Gaussian distribution also statistically independent

The above properties can be proven in a straight-forward manner based on the definitions of spectral density, correlation, etc.

These properties are important for us, later, when e.g. analyzing the symbol error and bit error probabilities of different modulations as well as in devising optimal decision rule for RX.
Overall received signal and system model, cont’d

Finally, when combined with symbol rate sampling, the corresponding discrete-time received signal samples read

\[ Q_k = Q(kT) = \sum_m A_m p(kT - mT) + Z(kT) \]

\[ = \sum_m A_m p_{k-m} + Z_k \]

\[ = A_k * p_k + Z_k \]

where \( Z_k = Z(kT) \) denotes sampled noise while \( p_k = p(kT) \) denotes the sampled total pulse-shape.

Again, in general, the total pulse shape \( p(t) = g(t) * h_{LP}(t) * f(t) \) is not a Nyquist pulse, due to the channel distortion, and thus the values \( p_k = p(kT) \) for \( k \neq 0 \) are non-zero which corresponds to ISI.

Similar to earlier, we can write out the above expression again as

\[ Q_k = \sum_m A_m p_{k-m} + Z_k \]

\[ = ... + A_{k+1} p_{-1} + A_k p_0 + A_{k-1} p_1 + ... + Z_k \]

- the ISI strength depends on the values of \( p_k = p(kT), \ k \neq 0 \) compared to \( p_0 \)
- notice that since \( p(t) = g(t) * h_{LP}(t) * f(t) \) is generally a complex-valued pulse, also the ISI coefficients \( p_k = p(kT) \) are complex-valued.
Overall received signal and system model, cont’d

We will use this model, and particularly the received discrete-time signal model of the form

\[ Q_k = A_k \cdot p_k + Z_k \]

in many different occasions in the continuation, e.g.

- when deriving optimal decision rules for making reliable decisions about the transmitted symbols in the receiver under noise
  - this is where we will need the information, e.g., about the noise statistics
- when analyzing the symbol error probability and bit error probability of different symbol alphabets under noise
  - again we will need the information about the noise statistics
- when deriving different receiver signal processing mechanisms, primarily channel equalizers and sequence detector, to mitigate or deal with the ISI
  - this is where we capitalize on the complete received signal model, i.e., the structure of the ISI as well as the noise statistics

Example: baseband received signal and ISI

As elaborated on the previous parts of the notes, a frequency-selective multipath radio channel will distort the pulses badly.

To begin with, we’ll illustrate through a simple example where

- 16-QAM data is transmitted over the channel at \( R_{\text{sym}} = 10 \text{ MHz} \) with square-root raise cosine pulse-shape filtering in both TX and RX
- without a frequency-selective channel, the pulse-shape at the RX filter output would be a beautiful raised-cosine Nyquist pulse
- however, the example frequency-selective channel will distort the pulse-shape badly and create intersymbol interference, ISI
- notice that the fading response is non-symmetric around zero frequency, when interpreted for I/Q downconverted RX signal
Example: baseband received signal and ISI, cont’d

Non-symmetric channel response causes the received pulse-shape to be complex-valued, and in general a non-Nyquist pulse as shown below

- creates intersymbol interference, ISI

The corresponding symbol rate samples are illustrated below, without any additive noise

- clearly, ISI is substantial distortion or degradation in the symbol level signal structure which needs to be accounted for in the receiver

Dealing with ISI in the receiver, like we already concluded in the context of baseband transmission, can be addressed through

- Discrete-time equalization filters, or
- Sequence detection
- Details will come later in this course

Notice that in the carrier-modulated case, all detection processing (e.g. discrete-time equalization filter) is complex-valued

- Physically this is stemming from the fact that, in general, the bandpass channel response does not have any symmetry wrt. the carrier center-frequency (see next slide)
- Hence from the baseband signal perspective (signal after downconversion in the receiver) this is complex filtering
Example: impact of noise on 16-QAM constellation

In addition to the ISI, naturally also the additive channel noise is limiting the receiver performance

- below are some examples of the received signal constellation, with 16-QAM, and with different noise variances (here only noise, no ISI)

\[ \sigma^2 = N_0 \int_{-\infty}^{\infty} |f(t)|^2 \, dt = N_0 \int_{-\infty}^{\infty} |F(f)|^2 \, df \]

- naturally, the larger is the noise variance (power), the larger is the degradation

5. ERROR PROBABILITY ANALYSIS AND RELATED ASPECTS

Next we address the reliability of symbol and bit detection in the receiver under noise, primarily focusing on I/Q modulated PAM/PSK/QAM

- what is the probability of correctly (or incorrectly) detecting a bit or symbol under noise?
- and how large SNR different symbol constellations need to reach a certain target error probability?

Based on our previous modeling, the received discrete-time sample sequence \( Q_k \) is, in general, of the form (see the previous slides)

\[ Q_k = A_k * p_k + Z_k \]

where \( A_k \) denotes the transmitted symbols, \( p_k \) is the ISI profile, and \( Z_k \) is filtered and sampled channel noise.

- noise \( Z_k \) is Gaussian, its real and imaginary parts are statistically independent and both of them have the variance (power)

\[ \sigma^2 = N_0 \int_{-\infty}^{\infty} |f(t)|^2 \, dt = N_0 \int_{-\infty}^{\infty} |F(f)|^2 \, df \]

- as a special case, we have the baseband PAM system model where all variables are real-valued

In the following, we analyze the probability of bit errors (\( \hat{b}_k \neq b_k \)) and symbol errors (\( A_k \neq A_k \)), under a simplifying assumption that

- there is no ISI, i.e., we focus on the impact of noise only and thus

\[ Q_k = A_k + Z_k \]
Concepts of symbol error, bit error and packet/block error

In general, one can analyze error and error probabilities in various different ways:

- **symbol error probability**: average probability that a transmitted symbol (from a given constellation) is detected incorrectly
- **bit error probability**: average probability that a transmitted bit is detected incorrectly
- **packet or block error probability**: average probability that a packet or block of detected bits contain one or more bit errors

All these are useful in quantifying and understanding the reliability of communications.

Bit errors and corresponding packet errors are most useful and valuable from the end-to-end communication and application perspective.

Furthermore, once we introduce error control coding later in the course, we can also separate between uncoded and coded system bit error probabilities

- here, however, we do not assume any error control coding, yet

In general, symbols are the internal means of any communication system to represent and transfer information, hence we start our analysis from quantifying and measuring symbol error probabilities

- many times in literature, symbol error probability is also called symbol error rate (terminology wise)

---

Noisy constellation and the minimum distance detection principle

Like we already concluded earlier, the received symbol rate samples do not exactly match with the ideal symbol values

- principal examples for 4-PSK and 16-QAM below:

![Noisy constellation example](image)

Based on our received signal modeling and noise analysis, the received signal samples form Gaussian distributed "clouds" around the ideal symbol values.

The detector should make reliable decisions about the transmitted symbols, i.e., choose such symbol from the constellation as the symbol decision that the received sample \( \hat{Q}_k \) most likely corresponds to.

Intuitively, a good way to make decisions is to choose such symbol \( \hat{A}_k \), which minimizes the distance or squared distance \( |Q_k - \hat{A}_k|^2 \)

- this is the so called minimum distance detector

- we will show later that this is also the optimum way to make decisions under certain assumptions, i.e., it is the detector that minimizes the detection error probability

These optimality aspects are addressed later, while in the following we

- analyze the detection error probability for PAM, QAM and PSK type of constellations assuming the minimum distance detector
Minimum distance principle and detection boundaries

Minimum distance detector imposes decision thresholds and corresponding decision regions around the ideal constellation points, i.e., the subsets of real and/or complex planes in which all points are closer to the considered symbol than to any other symbol.

The following figures show the decision regions for 4-PSK and 16-QAM:

Now, a symbol error corresponds to a situation where the received sample $Q_k$ is not anymore located within the decision region of the correct transmitted symbol.

- thus, a symbol error occurs which corresponds to one or more bit error, depending on which bit combination the erroneous symbol corresponds to.

In the following, we analyze the symbol error probability such that

- first, the individual symbol error probabilities of all different constellation points are calculated (different points in the constellation have, in general, different sensitivity to noise).
- then, the actual average symbol error probability is obtained by averaging these over the different symbol values.

Recap: Gaussian distribution and cumulative probabilities

In the following symbol error probability analysis, we end up commonly calculating cumulative tail probabilities of Gaussian distributions.

These can be conveniently expressed using the so-called $Q$ function, $Q(k)$, which is defined for a normalized Gaussian distribution $f_X(x)$ (variance $\sigma^2 = 1$ and mean $m = 0$) as follows:

$$ Q(k) = P(X > k) = \int_k^\infty f_X(x)dx = \frac{1}{\sqrt{2\pi}} \int_k^\infty e^{-x^2/2}dx $$

The values of the Q function can be trivially obtained numerically (using e.g. Matlab).

For a Gaussian variable $X$ with mean $m$ and variance $\sigma^2$, the tail probabilities can be then easily obtained as

$$ P(X > x') = Q\left(\frac{x' - m}{\sigma}\right) $$

As a side note, for larger values of $k$, say $k > 3$, there are also different approximations, e.g.,

$$ Q(k) \approx \frac{1}{\sqrt{2\pi}} \int_k^\infty e^{-x^2/2}dx \approx \frac{1}{\sqrt{2\pi} \cdot \frac{1}{2}} e^{-k^2/2} $$
Symbol error probability, basic binary case (2-PAM)

This is the most elementary case where the size of the constellation is $M=2$, and the two symbols are real-valued, say $-a$ and $+a$. The decision threshold of the minimum distance rule is at 0 (why?).

The distributions of the received sample $Q_k = A_k + Z_k$ are shown below, for both possible values of the transmit symbol $A_k$. The figures also show the Gaussian tail probabilities corresponding to the symbol errors.

$$Q_k = -a + Z_k : \quad f_{Q_k|A_k}(q_k | -a) \quad P(error \mid A_k = -a) = P(Q_k > 0) = P(Z_k > a) = Q(a/\sigma)$$

$$Q_k = +a + Z_k : \quad f_{Q_k|A_k}(q_k | +a) \quad P(error \mid A_k = +a) = P(Q_k < 0) = P(Z_k < -a) = Q(a/\sigma)$$

The symbol error probabilities thus read (below $d = 2a$):

$$P(symbol \ error) = P(symbol \ error \mid A_k = +a) = P(symbol \ error \mid A_k = -a) = Q\left(\frac{a}{\sigma}\right) = Q\left(\frac{d}{2\sigma}\right)$$

Symbol error probability, binary case (2-PAM), cont’d

Based on the previous expressions, the error probability depends on
- distance between the symbol values, $d$, and
- noise variance (power) $\sigma^2$

This is intuitively plausible, right?

Example: Let’s evaluate the numerical values of the error probability, in an example case of $a = 1$ and for different noise variance values of $\sigma^2 = \{1, 0.1, 0.01, 0.001\}$. Matlab allows calculating the $Q(.)$ function values through e.g. the so called erfc function (see help erfc)

```matlab
>> % Define Q-function using erfc (see help erfc)
>> % --------------------------------------------
>> function q = quu(x)
>>     q = 0.5*erfc(x/sqrt(2));  % Test it for symbol error probability of binary PAM
>> % --------------------------------------------------
>> a = 1;  % symbols +a, -a
>> var = [1 1e-1 1e-2 1e-3];  % noise variance
>> sigma = sqrt(var);  % noise std
>> >> P = quu(a./sigma);  % error probabilities
>> P(1) ans = 0.158655253931457
>> P(2) ans = 7.827011290012758e-004
>> P(3) ans = 7.619853024160593e-024
>> P(4) ans = 8.979163924003197e-220
```
Symbol error probability, binary case (2-PAM), cont’d

In a more general case of binary symbols but assuming that the two symbol values are arbitrary complex numbers, say \(a_i\) and \(a_j\), the received sample distributions look like (in the graphical illustration, the other symbol value is assumed to be 0):

If the noise distribution in the received sample is Gaussian along both the Re and Im axes, with variance \(\sigma^2\), and the noise Re and Im components are independent, the error probability reads

\[
P(\text{symbol error}) = P(\text{symbol error when } a_i \text{ transmitted})
\]

\[
= P(\text{symbol error when } a_j \text{ transmitted})
\]

\[
= Q\left(\frac{d}{2\sigma}\right)
\]

where \(d = |a_i - a_j|\) denotes the Euclidian distance between the symbols.

This is a handy result in the continuation, in some cases, to evaluate pairwise error probabilities, as

\[
P(\text{detector prefers } a_i \text{ when } a_j \text{ transmitted}) = Q\left(\frac{d}{2\sigma}\right)
\]

Symbol error probability, 4-PAM

In this case, we have \(M=4\) level real alphabet \([-3a, -a, +a, 3a]\).

Let’s first consider that the transmit symbol \(A_k = -a\). Then the distribution of the received sample \(Q_k\) is (again, \(d = 2a\)):

\[
Q_k = -a + Z_k:
\]

\[
Q(d/2\sigma) \quad f_{Q_k|A_k}(q_k | -a) \quad Q(d/2\sigma)
\]

Thus, the error probability can be expressed as

\[
P(\text{symbol error} | A_k = -a) = P(Q_k > 0 \text{ or } Q_k < -2a)
\]

\[
= P(Z_k > a \text{ or } Z_k < -a)
\]

\[
= Q\left(\frac{d}{2\sigma}\right) + Q\left(\frac{d}{2\sigma}\right)
\]

\[
= 2Q\left(\frac{d}{2\sigma}\right)
\]

Stemming from the symmetry of the situation, this applies also in the case of \(A_k = +a\).
Symbol error probability, 4-PAM, cont’d

Similarly, when the transmit symbol $A_k = -3a$, the distribution of the received sample $Q_k$ is of the form $(d = 2a)$:

$$Q_k = -3a + Z_k$$

Now, the error probability reads

$$P(\text{symbol error} | A_k = -3a) = P(Q_k > -2a) = P(Z_k > a) = Q\left(\frac{d}{2\sigma}\right)$$

Again, due to symmetry, this applies also when $A_k = +3a$.

Thus, as a summary, with 4-PAM, we have:

$$P(\text{symbol error} | A_k = \pm a) = 2Q\left(\frac{d}{2\sigma}\right)$$

Finally, assuming that all symbols are equally likely, the average symbol error probability of 4-PAM reads

$$P(\text{symbol error}) = \frac{2}{4} \times 2Q\left(\frac{d}{2\sigma}\right) + \frac{2}{4} \times Q\left(\frac{d}{2\sigma}\right) = 1.5Q\left(\frac{d}{2\sigma}\right)$$

Symbol error probability, generic M-PAM case

We can trivially generalize the previous 4-PAM derivations to a generic M-PAM alphabet of the form

$$\{\pm a, \pm 3a, \pm 5a, \ldots, \pm (M - 1)a\}$$

as (why?)

$$P(\text{symbol error}) = \frac{M - 2}{M} \times 2Q\left(\frac{d}{2\sigma}\right) + \frac{2}{M} \times Q\left(\frac{d}{2\sigma}\right) = 2\frac{M - 1}{M} Q\left(\frac{d}{2\sigma}\right)$$

Example: 8-PAM

$$\Rightarrow P(\text{symbol error}) = \frac{14}{8} Q\left(\frac{d}{2\sigma}\right) = 1.75Q\left(\frac{d}{2\sigma}\right)$$
Symbol error probability, 4-PSK (QPSK)

One form of QPSK constellation is \{b, jb, -b, -jb\}

To simplify the analysis, we consider a $45^\circ$ rotated version of this constellation, i.e., \{\pm b / \sqrt{2} \pm jb / \sqrt{2}\} (see the illustration below).

This has no impact on the final result, while the also shows the symbol error probability of 4-QAM.

Based on our earlier assumptions, the Re and Im components of the noise are independent, and both have variance $\sigma^2$.

Let’s first consider that $A_k = -b / \sqrt{2} - jb / \sqrt{2}$, for a correct decision is made if the noise sample $Q_k = A_k + Z_k$ is within the grey-shaded box in the above figure. In this case, the probability of correct decision is

\[
P(\text{correct decision} \mid A_k = -b / \sqrt{2} - jb / \sqrt{2}) = P(\text{Re}[Z_k] < d / 2 \text{ and } \text{Im}[Z_k] < d / 2)
\]

\[
= P(\text{Re}[Z_k] < d / 2) \times P(\text{Im}[Z_k] < d / 2)
\]

\[
= \left(1 - Q\left(\frac{d}{2\sigma}\right)\right)^2
\]

\[
= 1 - 2Q\left(\frac{d}{2\sigma}\right) + Q^2\left(\frac{d}{2\sigma}\right)
\]

Symbol error probability, 4-PSK (QPSK), cont’d

The corresponding symbol error probability is then given by

\[
P(\text{symbol error} \mid A_k = -b / \sqrt{2} - jb / \sqrt{2}) = 1 - P(\text{correct decision} \mid A_k = -b / \sqrt{2} - jb / \sqrt{2})
\]

\[
= 2Q\left(\frac{d}{2\sigma}\right) - Q^2\left(\frac{d}{2\sigma}\right)
\]

Due to the symmetry of the 4-PSK constellation in the complex plane, all symbol values have identical noise sensitivity, and the average symbol error probability of 4-PSK reads directly

\[
P(\text{symbol error}) = 2Q\left(\frac{d}{2\sigma}\right) - Q^2\left(\frac{d}{2\sigma}\right)
\]

Latter approximate form applies when $Q\left(\frac{d}{2\sigma}\right)$ is small. In most practical cases, this is a good approximation (we’ll illustrate this a little later, in a concrete manner).
Symbol error probability, 16-QAM

We consider the 16-QAM alphabet of the form \( \{ \pm c, \pm 3c \} + j \{ \pm c, \pm 3c \} \), shown also below:

\[
\begin{align*}
\text{Im}[A_k] &\quad \begin{array}{c}\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\quad \text{Re}[A_k]\end{array} \\
\text{d}\left\{\begin{array}{c}\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\quad \text{d}\end{array}\right\}
\end{align*}
\]

\[d = 2c\]

In terms of noise sensitivity, there are basically three different kinds of symbol values:

- **inner points**: \( \{ \pm c \} + j \{ \pm c \} \)
- **corner points**: \( \{ \pm 3c \} + j \{ \pm 3c \} \)
- **other points on the side**: \( \{ \pm c \} + j \{ \pm 3c \} \text{ ja } \{ \pm 3c \} + j \{ \pm c \} \)

The symbol error probabilities of these different types of symbol values are expressed in the following.

1) With the inner symbols (e.g., \( A_k = -c - jc \)) we can write:

\[
P(\text{correct decision} \mid A_k \text{ inner point}) = P(-d/2 < \text{Re}[Z_k] < d/2) \times P(-d/2 < \text{Im}[Z_k] < d/2)
\]

\[
= \left(1 - 2Q\left(\frac{d}{2\sigma}\right)\right)^2
\]

\[
\Rightarrow P(\text{symbol error} \mid A_k \text{ inner point}) = 1 - P(\text{correct decision} \mid A_k \text{ inner point})
\]

\[
= 4Q\left(\frac{d}{2\sigma}\right) - 4Q^2\left(\frac{d}{2\sigma}\right)
\]

\[
\approx 4Q\left(\frac{d}{2\sigma}\right)
\]

2) For the corner symbol values (e.g., \( A_k = -3c - j3c \)), we can write:

\[
P(\text{correct decision} \mid A_k \text{ corner point}) = P(\text{Re}[Z_k] < d/2) \times P(\text{Im}[Z_k] < d/2)
\]

\[
= \left(1 - Q\left(\frac{d}{2\sigma}\right)\right)^2
\]

\[
\Rightarrow P(\text{symbol error} \mid A_k \text{ corner point})
\]

\[
= 1 - P(\text{correct decision} \mid A_k \text{ corner point})
\]

\[
= 2Q\left(\frac{d}{2\sigma}\right) - Q^2\left(\frac{d}{2\sigma}\right)
\]

\[
\approx 2Q\left(\frac{d}{2\sigma}\right)
\]
Symbol error probability, 16-QAM, cont’d

3) For other points (e.g., $A_k = -3c - je$), we can write:

$$P(\text{correct decision} | A_k \text{ not inner or corner point}) = P(\text{Re}[Z_k] < d/2) \times P(-d/2 < \text{Im}[Z_k] < d/2)$$

$$= (1 - Q\left(\frac{d}{2\sigma}\right))(1 - 2Q\left(\frac{d}{2\sigma}\right))$$

$$\Rightarrow P(\text{symbol error} | A_k \text{ not inner or corner point})$$

$$= 1 - P(\text{correct decision} | A_k \text{ not inner or corner point})$$

$$= 3Q\left(\frac{d}{2\sigma}\right) - 2Q^2\left(\frac{d}{2\sigma}\right)$$

$$\cong 3Q\left(\frac{d}{2\sigma}\right)$$

Assuming finally that all symbol values are equally likely, the average symbol error probability of 16-QAM reads:

$$P(\text{symbol error}) = \left[4Q\left(\frac{d}{2\sigma}\right) - 4Q^2\left(\frac{d}{2\sigma}\right)\right] \times \frac{4}{16}$$

$$+ \left[2Q\left(\frac{d}{2\sigma}\right) - Q^2\left(\frac{d}{2\sigma}\right)\right] \times \frac{4}{16}$$

$$+ \left[3Q\left(\frac{d}{2\sigma}\right) - 2Q^2\left(\frac{d}{2\sigma}\right)\right] \times \frac{8}{16}$$

$$= 3Q\left(\frac{d}{2\sigma}\right) - 2.25Q^2\left(\frac{d}{2\sigma}\right)$$

$$\cong 3Q\left(\frac{d}{2\sigma}\right)$$

Again, the last approximate form applies well when $Q(d/2\sigma)$ is small.

Symbol error probabilities, some intermediate conclusions

Using the previous results, we can summarize the following:

2-PAM: $P(\text{symbol error}) = Q\left(\frac{d}{2\sigma}\right)$

4-PAM: $P(\text{symbol error}) = 1.5Q\left(\frac{d}{2\sigma}\right)$

4-PSK: $P(\text{symbol error}) \approx 2Q\left(\frac{d}{2\sigma}\right)$

16-QAM: $P(\text{symbol error}) \approx 3Q\left(\frac{d}{2\sigma}\right)$

In all these results, the symbol error probability depends on the minimum distance, $d$, and the noise variance (power) $\sigma^2$.

This is a generally applicable fact and observation, and conforms well with intuition also.

Furthermore, for given noise power $\sigma^2$, the error probability is approximatively the same with different constellations (roughly) if the minimum distance $d$ is the same.
Symbol error probability, general M-QAM case

We next establish the symbol error probability of generic M-QAM.

We assume that the value of $M \in \{4, 16, 64, 256, 1024, \ldots\}$, which means that the constellation is a true square, and thus you can view it as a $\sqrt{M} \times \sqrt{M}$ grid in complex plane.

I'll skip the derivation details (they are identical to 16-QAM case, when it comes to the individual symbol error probabilities), while the final result reads

$$P(\text{symbol error}) =$$

$$[4Q\left(\frac{d}{2\sigma}\right) - 4Q^2\left(\frac{d}{2\sigma}\right)] \times \frac{M - 4 - 4 \times (\sqrt{M} - 2)}{M} + [2Q\left(\frac{d}{2\sigma}\right) - Q^2\left(\frac{d}{2\sigma}\right)] \times \frac{4}{M} + [3Q\left(\frac{d}{2\sigma}\right) - 2Q^2\left(\frac{d}{2\sigma}\right)] \times \frac{4 \times (\sqrt{M} - 2)}{M}$$

As a concrete example, let’s consider 1024-QAM ($M=1024$):

$$P(\text{symbol error}) = 3.875Q\left(\frac{d}{2\sigma}\right) - 3.754Q^2\left(\frac{d}{2\sigma}\right)$$

Notice that when the constellation size ($M$) increases, the relative share of the inner points increases.

In the limiting case, with $M \to \infty$, we can write

$$P(\text{symbol error}) \to 4Q\left(\frac{d}{2\sigma}\right) - 4Q^2\left(\frac{d}{2\sigma}\right)$$

Union Bound analysis method

In some cases, e.g. M-PSK with $M > 4$, deriving exact symbol error probability expressions in analytic form becomes tedious.

In such cases, we can easily get good approximations using a method called union bound.

Suppose we consider an arbitrary set of points in the complex plane, of size $M$. Suppose the transmit symbol is one of these points, say $A_k = a_i$. Then, let’s denote with $E_{ij}$ the case that the received sample is closer to another symbol $a_j$ than $a_i$. In this case, we can first write

$$P(\text{symbol error} | A_k = a_i) = P(E_{i1} \cup E_{i2} \cup \cdots \cup E_{i,i-1} \cup E_{i,i+1} \cup \cdots \cup E_{iM})$$

Union bound forms now an upper bound for the error probability as:

$$P(\text{symbol error} | A_k = a_i) \leq \sum_{j=1}^{M} P(E_{ij}) = \sum_{j=1}^{M} Q\left(\frac{d_{ij}}{2\sigma}\right)$$

where $d_{ij} = |a_i - a_j|$.

For example, in the case of 4-PSK, this union bound yields

$$P(\text{symbol error}) \leq 2Q\left(\frac{d}{2\sigma}\right) + Q\left(\frac{d}{\sqrt{2}\sigma}\right)$$

which is a good approximation, with reasonably low error probabilities for the exact result (obtained earlier).

In practice, the union bound can be adopted such that only the nearest neighbors and the corresponding pairwise error probabilities are considered in the summation.
Union Bound analysis method, cont’d

Example: 8-PSK, and suppose the symbol values read
\[ \{ \pm b, \pm jb, \pm be^{j\pi/4}, \pm be^{j3\pi/4} \} \]

With the union bound, and considering only the nearest neighbours, we can then trivially write:

\[ P(\text{symbol error}) \simeq 2Q \left( \frac{d}{2\sigma} \right) \]

where the distance between the neighbors \( d = b\sqrt{2} - \sqrt{2} \) (easy to verify from the 8-PSK constellation points).

Bit error probability and Gray code

Next, we address the analysis of the actual bit error probability which is the most fundamental measure of the reliability of any digital communication system.

For a given symbol constellation and symbol error probability, the corresponding bit error probability depends essentially on the mapping between bits and symbols.

With reasonable SNR values, when symbol errors occur, they take place with high probability between the neighboring symbols.

Thus, it is intuitively plausible to do the bit-to-symbol mapping such that the bit combinations of the neighboring symbols differ as little as possible.

A specific form is the so-called Gray code (mapping) where the nearest neighbors differ by only one bit:

\[ \text{Re}[A_i] \quad \text{Im}[A_i] \]

With such mappings, it is then easy to write (an approximation of) the bit error probability as:

\[ P(\text{bit error}) \approx \frac{1}{\log_2(M)} P(\text{symbol error}) = \frac{1}{B} P(\text{symbol error}) \]

when the constellation size \( M = 2^B \) (\( B \) bits per symbol).
Bit error probability and Gray code, cont’d

The previous expression is a good approximation for the average bit error probability assuming Gray coding/mapping between the bits and symbols.

On the other hand, it is also possible to express the exact bit error probability, as follows:

- go through the symbol constellation, symbol by symbol, and express the mutual symbol error probabilities between all different symbol pairs
  - these can be expressed using the Q function, by using similar analysis principles to what we did earlier, but now really evaluating the mutual pairwise symbol error probabilities
- calculate then also how many true bit errors are realized in the different pairwise symbol error case
- finally, weight and sum all these together
- ..it takes a little effort but is doable, for sure

Block error or packet error probability

All practical systems communicate bits in blocks or packets

- thus, the block error or packet error probability is a very essential measure

The error detection methods (which we discuss at the latter part of the course), can then be used to detect whether the received bit packet contains errors or not
- if there are errors, then the receiver can e.g. ask the transmitter to resend the packet (automatic repeat request, ARQ)

For a packet of $L$ bits and symbol constellation size of $M = 2^B$, the $L$ bits correspond to $L/B$ symbols, and thus the block error probability can be expressed as

$$P(\text{block error}) = 1 - [1 - P(\text{symbol error})]^L/B$$

$$\approx \frac{L}{B} P(\text{symbol error})$$

The latter form is accurate in particular when the symbol error probability is reasonably small

- this stems from Taylor series expressions, truncated to the linear term (that is, in general, $(1 - x)^n \approx 1 - nx$ when $x$ is small)

Example:

Suppose

$L = 1000$, $B=4$, $P(\text{symbol error}) = 10^{-6}$

Then

$$P(\text{block error}) \approx 2.5 \times 10^{-4}.$$
Impact of SNR

Intuition: The error probability in the detection should depend on the signal-to-noise ratio, SNR, of the detector input signal.

This is addressed, in more details, in the following.

We will use our simple detector input signal model of the form

\[ Q_k = A_k + Z_k \]

to first define the SNR, and then express the earlier derived symbol error probability expressions for different constellations using the SNR measure.

In other words, the earlier Q(.) function based results depend on \( d / 2\sigma \) thus we just need to find the dependence \( d / 2\sigma \leftrightarrow \text{SNR} \) for different considered constellations

- this is what we do next

Symbol error probabilities vs. SNR

Based on our simple received signal model of the form

\[ Q_k = A_k + Z_k \]

the detector input SNR can now be defined as

\[ \text{SNR} = \frac{E[|A_k|^2]}{2\sigma^2} \]

Here, the useful signal power is the symbol power \( E[|A_k|^2] \), while the total noise power is \( 2\sigma^2 \) (as earlier, \( \sigma^2 \) denotes the noise power, per Re and Im component, thus the total noise power is \( \sigma^2 + \sigma^2 = 2\sigma^2 \)).

On the next slide, we have expressed the symbol error probabilities as functions of the above SNR measure

- that is, we have mapped the argument \( d / 2\sigma \) of the earlier Q(.) function based expressions to the SNR measure

\[ \frac{d}{2\sigma} \leftrightarrow \frac{E[|A_k|^2]}{2\sigma^2} \]

We also show three different versions of the error probabilities: exact, approximate, and rough. For example for 4-PSK, these are:

- exact expression: \( 2Q(\sqrt{\text{SNR}}) - Q^2(\sqrt{\text{SNR}}) \)
- approximation: \( 2Q(\sqrt{\text{SNR}}) \)
- rough approximation: \( Q(\sqrt{\text{SNR}}) \)

Rough approximations mean that the error probabilities of different constellations would be (approximatively) the same if the minimum distances are the same

General observation is that larger constellations need larger SNR, which is also intuitively clear – see next page for more details.
Symbol error probabilities vs. SNR, cont’d

Following table shows the results after the $d / 2 \sigma \leftrightarrow \text{SNR}$ mapping for the three example constellations (2-PAM, 4-PSK, 16-QAM):

<table>
<thead>
<tr>
<th>symbol values</th>
<th>2-PAM</th>
<th>4-PSK</th>
<th>16-QAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>min. distance $d$</td>
<td>$2a$</td>
<td>$b\sqrt{2}$</td>
<td>$2c$</td>
</tr>
<tr>
<td>$E[</td>
<td>A</td>
<td>^2]$ exact $P_e$</td>
<td>$Q\sqrt{2} \times \text{SNR}$</td>
</tr>
<tr>
<td>approx. $P_e$</td>
<td>$Q\sqrt{2} \times \text{SNR}$</td>
<td>$2Q \sqrt{\text{SNR}}$</td>
<td>$3Q \sqrt{\text{SNR}}/5$</td>
</tr>
<tr>
<td>rough $P_e$</td>
<td>$Q\sqrt{2} \times \text{SNR}$</td>
<td>$Q \sqrt{\text{SNR}}$</td>
<td>$Q \sqrt{\text{SNR}}/5$</td>
</tr>
</tbody>
</table>

These are illustrated graphically below:

![Graph showing SNR vs. error probability for 2-PAM, 4-PSK, and 16-QAM](image)

Notice that with reasonably low error probability values, also the rough approximations are surprisingly accurate, when thinking about the differences along the SNR axis.

Symbol error probabilities vs. SNR, cont’d

Some observations and conclusions, when interpreted at SNR axis:

- 4-PSK is roughly 3 dB worse than 2-PAM
- 16-QAM is roughly 7 dB worse than 4-PSK
- Rough approximations are pretty accurate, at lower error probs.

With the rough approximations, different constellations can be compared very easily:

- e.g., 4-PSK vs. 16-QAM, in order to have identical error probabilities, it should hold that (see the previous page):
  \[ Q\left(\sqrt{\text{SNR}_{4-PSK}}\right) = Q\left(\sqrt{\text{SNR}_{16-QAM}}/5\right) \]
  \[ \Rightarrow \text{SNR}_{16-QAM} = 5 \times \text{SNR}_{4-PSK} \]

- that is, 16-QAM needs 5-times larger SNR compared to 4-PSK, which when converted to dBs is $10 \log_{10}(5) = 7$ dB (this can be observed from the previous page figure)

On the other hand, this type of analysis does not take into account that for fixed bandwidth and thus for fixed noise power, different modulations yield different bit rates.

... Or in an alternative form: if the target bit rate is fixed, then larger constellation needs lower symbol rate, and hence lower bandwidth, and therefore the noise powers in different systems are different.

The message thus is:

- difference in the needed SNRs does not necessarily automatically correspond to difference in the needed received signal power
- we explore this further in the classroom exercises
- furthermore, when we consider the concept of bit-energy a little later, the difference in the bit-rates facilitated by different constellations will be automatically taken into account
ERROR PROBABILITY ANALYSIS UNDER FADING

Our previous error probability analyses assume a very simple received signal model of the form

\[ Q_k = A_k + Z_k \]

- in other words, the received symbol rate samples are only distorted by additive noise

In practice, there is also ISI and thus \( Q_k = A_k * p_k + Z_k \).

On the other hand, in mobile communication systems, one essential ingredient is also time-selective fading.

If we neglect ISI but incorporate time-selective fading, the received symbol rate samples can be modeled as

\[ Q_k = p(k)A_k + Z_k \]

- here \( p(k) \) denotes the effective time-selective fading experienced in the system

Depending on the propagation environment, this fading process can be described through various alternative statistical models

- most common example is the so-called Rayleigh fading

- in this case, the Real and Imaginary parts of the complex fading gain \( p(k) \) are both Gaussian distributed and statistically independent, which means that the amplitude fading process \( |p(k)| \) has Rayleigh distribution while the squared envelope \( |p(k)|^2 \) is exponentially distributed (Chi-squared distribution)

- (we talked about these shortly already in the context of fading channel capacity analysis)

The severity of the fading, over time, is then commonly described by the Doppler spread, which is impacted by the carrier frequency and mobility

Next, we analyze shortly the symbol error probability of a minimum distance detector under time-selective fading and noise, i.e., for the received signal model of the form

\[ Q_k = p(k)A_k + Z_k \]

We start by first evaluating the instantaneous symbol error probability, i.e., assume that the value of the complex fading gain \( p(k) \) is fixed. For the most elementary case of 2-PAM, we can then directly write

2-PAM:

\[ P(\text{symbol error} | p(k) \text{ fixed}) = Q(\sqrt{2 \times SNR_p}) \]

where

\[ SNR_p = |p(k)|^2 \frac{E[|A_k|^2]}{2\sigma^2} \]

denotes the so-called instantaneous SNR (SNR for given fading state).
Symbol error probability under time-selective fading, cont’d

Then the actual average symbol error probability over the different fading states can be obtained by averaging the instantaneous error probabilities for given fading distribution.

For the example case of Rayleigh fading, \( |p(k)|^2 \) and thus the instantaneous SNR \( (SNR_p) \) is exponentially distributed, which we write

\[
SNR_p \sim \frac{1}{E[SNR_p]} e^{-SNR_p/E[SNR_p]} = \frac{1}{SNR_{ave}} e^{-SNR_p/\text{SNR}_{ave}}
\]

where

\[
\text{SNR}_{ave} = E[SNR_p] = E[|p(k)|^2] = \frac{E[|A|^2]}{2\sigma^2} = \sigma_p^2 \text{SNR}_{awgn}
\]

denotes the corresponding average SNR and \( \sigma_p^2 = E[|p(k)|^2] \) is the fading variance.

Then we can write for 2-PAM (as a concrete example)

\[
P(\text{average symbol error})
= E\left(Q\left(\sqrt{2 \times SNR_p}\right)\right)
= \int_0^\infty Q\left(\sqrt{2 \times SNR_p}\right) \frac{1}{SNR_{ave}} e^{-SNR_p/\text{SNR}_{ave}} dSNR_p
= \ldots
= \frac{1}{2} \left(1 - \sqrt{\frac{\text{SNR}_{ave}}{1 + \text{SNR}_{ave}}}\right)
\]

This expression is closed-form, and is surprisingly simple, isn’t it?

It shows that the error probability for Rayleigh fading depends only on the average SNR!

Assuming that \( \sigma_p^2 = E[|p(k)|^2] = 1 \), and thus that the average SNRs are the same in the systems without and with fading, the corresponding error probabilities with 2-PAM are illustrated below:

It is clear that fading has a substantial impact on the average symbol error probability.

Intuitively, the lowest fading deeps dominate

- thus one way to improve the situation and fight against fading is to adopt some form of diversity
- different multiantenna based TX and RX diversity methods are covered in the course ELT-43106 Multicarrier and Multiantenna Techniques

Error probabilities of other constellations (4-PSK, 16-QAM, etc) can be analyzed in a similar manner. Ready expressions available in the literature. Also, there are lots of ready results for fading distributions other than Rayleigh as well.
**SHORTLY ON CONSTELLATION OPTIMIZATION**

Based on our earlier analysis, and using the rough approximation, the symbol error probabilities of different constellations are all of the form

\[ P(\text{symbol error}) \approx Q\left(\frac{d}{2\sigma}\right) \]

Thus, if one wishes to try to minimize the error probability, it means that the minimum distance, \( d \), between the neighboring points in the complex plane should be maximized.

However, this is always implicitly tied to the average symbol power \( E \left| A_k \right|^2 \), hence there are no free lunches.

One can thus view the constellation optimization as designing the points in the complex plane such that the minimum distance, \( d \), is maximized under the constraint of

- fixed average symbol power \( E \left| A_k \right|^2 \), or
- fixed peak symbol power \( \max \left| A_k \right|^2 \) relative to the average

These aspects are shortly discussed in the following

- though it is true and useful to note that the gains observed through the optimization are commonly only fractions of a dB, compared to the standard PSK or QAM type of constellations

---

**Most common constellations, QAM**

**QAM, \( B=\log_2(M) \) even**

<table>
<thead>
<tr>
<th>( M )</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>64</td>
<td>6</td>
</tr>
</tbody>
</table>

Here, the I and Q components are both mutually independent \( \sqrt{M} \) signals.

- detection can be done in the receiver, independently for the I and Q components
- detector implementation is simple in this sense

---

**QAM, \( B=\log_2(M) \) odd**

<table>
<thead>
<tr>
<th>( M )</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>128</td>
<td>7</td>
</tr>
</tbody>
</table>

Here the I and Q components are not independent anymore, and thus the detection cannot be done independently for the I and Q.
Most common constellations, PSK

- Constellation points are located on a circle, thus the information is only the phase of the RF modulated signal.
- Envelope of the actual RF modulated signal is also more flat, compared to the QAM type of signals while its exact characteristics depend on the applied pulse-shape.
- Detection needs to be done for the complex received samples, as the I and Q are obviously related.

Other PSK related constellations

- These are extensions from the basic PSK such that there can be multiple different circles at which the constellation points are located.
- Information is thus already in both the envelope and phase of the RF modulated signal.
Hexagonal constellations

The name "hexagonal" comes from the fact that the decision regions are of hexagonal shape.

This type of constellations basically yield maximum minimum distance between the neighbors under a given average symbol power.

On the other hand, performance improvements in error probability vs. SNR, compared to e.g. QAM, are only fractions of dBs and thus these have not been really systematically adopted in commercial systems (at least so far).

Excursion: constellation of an old modem standard CCITT v.29

...what kind of decision boundaries would you get for this one?
6. COMPLEMENTARY LINEAR DIGITAL MODULATIONS

Differential PSK (DPSK)

DPSK is a variation of the basic PSK principle where the transmitted bits or symbols impact the phase change of the modulated carrier, between two consecutive symbol durations, instead of the true absolute phase used in PSK.

**Example:** differential binary PSK
- bit 0 => carrier phase remains the same
- bit 1 => carrier phase is changed 180 degrees

This has the benefit that there is no need for accurate phase synchronization in the receiver (called non-coherent detection).

On the other hand, the differential principle can be sensitive to error propagation or error accumulation.

Offset QAM/PSK

In offset modulation, the idea is to reduce the peak-to-average power ratio, PAPR, of the modulate waveform by imposing a deliberate delay of half the symbol period between the I and Q branches.

This means that also the receiver will impose an additional half a symbol period delay between the samplers in the I and Q branches.

The delay between the I and Q branch baseband waveforms reduce the TX signal PAPR because now the I and Q cannot reach zero value at the same time

- this is illustrated in the following using a scatter diagram which means plotting the baseband complex waveform in complex plane
- here, it is really the waveform that is plotted, not just the values at the symbol instants (so this is kind an eye diagram but in complex plane)

Complementary linear digital modulations, cont’d

Offset QAM/PSK transmitter and scatter diagrams:

Carrierless amplitude/phase (CAP) processing principle

This is a special case of general I/Q modulated PAM/PSK/QAM in which the carrier frequency $f_c$ is an integer multiple of the symbol rate $1/T$, i.e., $f_c = K \times (1/T)$.

In such special case, the I/Q modulated signal can be generated without an explicit I/Q mixer, using modulated pulse-shaping filters.

This is depicted on the following page.
Complementary linear digital modulations, cont'd

In terms of math:

\[
x(t) = \text{Re}[e^{j\omega t} \sum_m A_m g(t - mT)] \\
= \cos(\omega_c t) \sum_m \text{Re}[A_m] g(t - mT) - \sin(\omega_c t) \sum_m \text{Im}[A_m] g(t - mT) \\
= \ldots \\
= \sum_m \text{Re}[A_m] g_I(t - mT) - \sum_m \text{Im}[A_m] g_Q(t - mT)
\]

where \( g_I(t) = g(t) \cos(\omega_c t) \) and \( g_Q(t) = g(t) \sin(\omega_c t) \), and as already stated the center-frequency \( f_c = K \times (1/T) \).

In this structure, the I/Q modulation and the basic pulse-shaping are combined to one, in the form of modulated bandpass filters in the I and Q branches.

Similarly, on the RX side, I/Q demodulation and ordinary lowpass filtering can be combined into bandpass filters, followed by symbol rate sampling.

Practical limitation, if using digital filters, is that the CAP processing principle can only be applied up to certain center frequencies (not to GHz scale RF frequencies)

- applied e.g. in fast subscriber line techniques (VDSL standard)

\[\pi/4\text{ QPSK}\]

Here the idea is to have multiple differently phase-rotated versions of the basic QPSK constellation, and at any individual symbol instant, the one that keeps the transmit signal PAPR the lowest is adopted in the bit to symbol mapping.

More specifically: The baseline constellation is phase-rotated by 45° (\(\pi/4\)) between the consecutive symbol instants, which means an additional phase change of ±45° between the consecutive symbol instants, which in turn means that the envelope will not go to zero, and thus PAPR is reduced.

This technique is applied, e.g., in digital mobile communication networks.

\[\pi/2\text{ BPSK}\]

Similarly there exists also a modified version of the basic BPSK, called \(\pi/2\) BPSK, where two binary constellations \(\{-1, +1\}\) and \(\{-j, +j\}\) are alternated. This has a similar impact that the envelope won't go to zero at any time point (and thus the PAPR is reduced).
Complementary linear digital modulations, cont’d

Digital VSB and SSB

Digital VSB and SSB signals can be obtained by
- first creating a baseband PAM signal, with real-valued constellation
- then imposing appropriate sideband filtering to this signal

In case of SSB (see lecture notes of the Comm Theory course), the sideband filtering can be carried out using a Hilbert transformer
- this Hilbert filtering, as well as the basic pulse shape filtering can both be done digitally, as long as the signal is oversampled properly

In case of VSB, the approach is similar except for the fact that the sideband filtering is done so that deliberately a small vestigial sideband is retained
- thus, as a whole, the vestigial sideband is shaped according to the VSB principle (by the sideband filter) while the transition band of the other (main) sideband is shaped by the Nyquist pulse-shape filtering principle (by the pulse shape filter)

Notice that for a given RF bandwidth of $W_{BP}$, SSB and VSB allow for using (almost) double the symbol rate compared to I/Q modulated PAM/PSK/QAM (why?)
- however, since, the symbol constellation is limited to be real-valued, the spectral efficiency per real domain is essentially the same

Digital VSB is applied e.g. in US digital TV broadcast networks.

7. MODULATION EFFICIENCY ASPECTS

Next we address shortly how to measure and quantify modulation efficiency, in terms of bandwidth efficiency and energy efficiency.

Intuitively, these mean
- **bandwidth efficiency**: how efficiently the given modulation method is utilizing the given bandwidth
  - quantified commonly through the concept of spectral efficiency which means the achieved bit rate over the given bandwidth
  - measured in bits/s/Hz
- **energy efficiency**: how efficiently the given modulation method is utilizing the given power or energy
  - can be quantified, e.g., with the needed energy per bit to reach a certain target bit rate under a given reliability, i.e., symbol error probability or bit error probability, in the communication task
  - (can be) measured in J/bit

These are addressed in the following, while we start by defining the concepts of bit energy or energy per bit, and bit energy to noise density ratio ($E_b/N_0$).
Concept of bit energy and bit energy to noise density ratio

One common measure to quantify and measure the ratio of noise and useful signal in the receiver, in addition to the basic signal to noise ratio (SNR), is the so called bit energy to noise density ratio defined as

$$\frac{E_b}{N_0}$$

where
- $E_b$ refers to received useful signal energy per bit [J/bit]
- $N_0 / 2$ refers to double-sided noise spectral density [W/Hz]

Notice that while we are otherwise denoting the two-sided noise spectral density by $N_0$ in these lecture notes and in this course, it is indeed common in the literature to use the notation $N_0 / 2$ in this context, and thus this is what we do also here.

Now, we can express and relate $E_b / N_0$ ratio to the basic SNR as:
- Inband noise power $N$, assuming ideal filtering, is
  $$N = \frac{N_0}{2} 2W = N_0 W$$
- Useful signal power $S$ can be expressed as a product of the bit energy $W$ and bit rate $R_b$ as (why?)
  $$S = E_b R_b$$

Bit energy to noise density ratio, cont’d

Thus, the $S / N$ and $E_b / N_0$ ratios are directly related through

$$\frac{S}{N} = \frac{E_b R_b}{N_0 W} = \frac{E_b}{N_0} \times \frac{R_b}{W}$$

or

$$\frac{E_b}{N_0} = \frac{S}{N} \times \frac{W}{R_b}$$

Compared to ordinary $S / N$ ratio, $E_b / N_0$ ratio takes into account the bit rate or spectral efficiency obtained through different modulations.

Thus, in this sense, $E_b / N_0$ ratio can be viewed to be a more fair or fundamental measure, e.g., in comparing the bit error or symbol error probabilities of different modulations
- there is, however, nothing wrong in using the basic SNR in such comparisons
- you just have to be careful, in thinking which characteristics are fixed when doing the comparisons and when interpreting the results

When applied directly to I/Q modulated PAM/PSK/QAM (and assuming zero excess bandwidth, for simplicity), we can write

$$\frac{E_b}{N_0} = \frac{S}{N} \times \frac{W}{R_b} = \frac{S}{N} \times \frac{1}{T} \times \frac{1}{\log_2(M)} \times \frac{1}{T}$$

$$= \frac{S}{N} \times \frac{1}{\log_2(M)}$$

where $M$ is the size of the constellation
- this allows you, e.g., to revise and redraw the symbol error probability results on slide 161 (the Q(.) function things) using the $E_b / N_0$ ratio as the argument of the Q(.) functions and the measure on the x-axis (instead of the basic SNR)
- this will shown in these notes as well, a few slides onward

Communication theory wise, bit energy is the most fundamental power/energy measure.
SPECTRAL EFFICIENCY

Spectral efficiency is defined as the achieved bit rate over the used physical bandwidth, as:

\[ \frac{R_b}{W} \text{ bits/s/Hz} \]

This is one of the most fundamental performance measures of all communication systems

- enhancing the spectral efficiency is always one of the key targets in the R&D work

When applied to baseband PAM and I/Q modulated PAM/PSK/QAM, we can easily deduce the following

- **baseband PAM**
  \[
  \frac{R_b}{W} = \frac{\log_2(M) \frac{1}{T}}{(1 + \alpha) \frac{1}{2T}} = \frac{2 \log_2(M)}{1 + \alpha} \text{ bits/s/Hz}
  \]

- **I/Q modulated PAM/PSK/QAM**
  \[
  \frac{R_b}{W} = \frac{\log_2(M) \frac{1}{T}}{(1 + \alpha) \frac{1}{T}} = \frac{\log_2(M)}{1 + \alpha} \text{ bits/s/Hz}
  \]

where $M$ refers to the constellation size and $\alpha$ denotes the excess bandwidth factor

Spectral efficiency, cont’d

Thus, the maximum achievable spectral efficiencies are of the form (with $\alpha$ set to zero)

- **baseband PAM**: $2 \log_2(M)$ bits/s/Hz
- **I/Q modulated PAM/PSK/QAM**: $\log_2(M)$ bits/s/Hz

In general, like the above results show, it is true that for a given constellation size $M$, the baseband system is more spectrally efficient.

Some complementary comments

- in practical systems, spectral efficiency can be measured in many different protocol layers
  - physical layer bit rate vs. bit rate at higher protocol layers
- also, even at the physical layer, the core physical layer bits are commonly composed of coded information bits, physical layer reference signals, control channels, other frame structure related overhead, etc.
  - thus, one can measure the physical layer spectral efficiency also such that the overhead due to coding, reference signals, frame structure, etc., is first excluded from the bit rate
- in our basic expressions and analysis, however, we do not differentiate between different types of bits at physical layer
Spectral efficiency, simple numerical examples

**Example 1: Old phoneline modems**

As an example we consider a 28.8 kbits/s modem, with bandwidth of 3.6 kHz (voice bandwidth in old phoneline systems was in the order of 3-4 kHz). The basic spectral efficiency is thus

8 bits/s/Hz

**Example 2: LTE downlink, 20 MHz carrier bandwidth, single-stream transmission**

The basic physical layer bit rates in LTE cellular radio network, with 20 MHz carrier bandwidth and different assumed data modulations (constellations) are

- QPSK: 36 Mbps
- 16QAM: 72 Mbps
- 64QAM: 108 Mbps
- 256QAM: 144 Mbps

The corresponding physical layer spectral efficiencies are thus

- 1.8 bits/s/Hz
- 3.6 bits/s/Hz
- 5.4 bits/s/Hz
- 7.2 bits/s/Hz

Notice that the above bit rates, and thus the spectral efficiency numbers, do not differentiate between the redundancy imposed by error control codes, known reference symbols, control channels, and other protocol layers related overhead, from the actual information bits, thus the true spectral efficiencies in terms of the information rates over the used bandwidth are much smaller

ENERGY EFFICIENCY

Energy efficiency measures how efficiently the given modulation method is utilizing the given power or energy

- can be quantified, e.g., with the needed energy per bit (Joules per bit) to reach a certain target symbol error probability or bit error probability in the link, with a given bit rate target

That is, it measures how much energy is needed for the given communication task, i.e., to reach the given bit rate with given reliability

Intuitively, energy efficiency and spectral efficiency are contradictory features

- large symbol alphabet => large spectral efficiency but lower energy efficiency (why ?)
- small symbol alphabet => low spectral efficiency but better energy efficiency (why ?)

This is, generally speaking, true but the exact comparison of two different modulation methods needs to be still done with care

- This is because larger alphabets can facilitate the given bit rate with lower bandwidth

We’ll address this through some examples next, using the earlier modulation alphabets as examples

- 2-PAM
- QPSK
- 16QAM
Energy efficiency, cont’d

To begin with, we show below the previous results for the symbol error probabilities of the example modulation methods, but using now the Eb/N0 ratio in the x-axis, instead of the plain SNR.

- 2PAM and QPSK are very close, partially identical
- 16QAM is roughly 4dB worse, in terms of energy efficiency
- the actual bit energy (Eb) values can also be easily obtained, by multiplying the needed Eb/N0 value by the assumed noise density
  - N0 [dBm] = -174dBm + RX Noise Figure [dB]

Thus, for any given target symbol error probability (reliability target), one can read the required Eb/N0 values of different modulations:
- 2PAM and QPSK are very close, partially identical
- 16QAM is roughly 4dB worse, in terms of energy efficiency
- the actual bit energy (Eb) values can also be easily obtained, by multiplying the needed Eb/N0 value by the assumed noise density

Energy efficiency, cont’d

Below, we repeat the same exercise but use bit error probabilities instead of symbol error probabilities as the reliability measure:
- here we assume Gray coding based bit error probability curves
- conclusions are largely identical

Notice that in general, since the signal power $S = E_b R_b$, any difference in the needed bit energy $E_b$ translates directly to the same difference in the needed signal power when the target bitrate $R_b$ is fixed.
MODULATION EFFICIENCY LIMITS AND SHANNON CAPACITY

Recap: basic Shannon-Hartley capacity law

\[ C = W \log_2 \left( 1 + \frac{S}{N} \right) \]

where \( S \) is the received useful signal power and \( N \) the inband noise power.

From here, we can directly deduce an upper bound for the spectral efficiency as:

\[ \frac{v}{W} = \log_2 \left( 1 + \frac{S}{N} \right) \quad \Leftrightarrow \quad \frac{S}{N} = 2^v - 1 \]

Thus, for a given SNR, this is the maximum spectral efficiency that any system can achieve!

Alternatively, for a target spectral efficiency of \( v \), the second expression gives the minimum SNR that in theory can facilitate the required spectral efficiency.

From above, one can also easily obtain an upper bound for the needed received signal power:

\[ S = N(2^v - 1) \approx N2^v \quad \text{when} \quad v \gg 1 \]

Thus, we see that when one wishes to increase the spectral efficiency by 1 bit/s/Hz, the useful received signal power must be doubled (assuming a fixed noise power).

Maximum Shannon spectral efficiency vs. PAM/PSK/QAM

Like we just concluded, the maximum Shannon spectral efficiency is of the form

\[ v = \frac{C}{W} = \log_2 \left( 1 + \frac{S}{N} \right) \]

On the other hand, the maximum spectral efficiency of baseband PAM and I/Q modulated PSK/QAM systems are (see earlier slides)

\[ 2 \log_2(M) \quad \text{and} \quad \log_2(M) \quad (M = \text{size of the symbol alphabet}) \]

respectively.

Now, one could seemingly argue that by simply increasing the symbol alphabet size \( M \), we could go beyond the Shannon limit, which is in turn should not be possible.

- **Question:** What is wrong with above reasoning?
- **Answer:** Nothing and everything!

That is, it is strictly speaking incorrect to directly compare or relate the maximum Shannon spectral efficiency and \(- \log_2(M)\) type of spectral efficiency of PAM/PSK/QAM.

- **Question:** Why?
- **Answer:** Shannon capacity theorem gives the maximum rate of information that can in theory pass through a bandlimited AWGN channel error free.

On the other hand, PAM/PSK/QAM systems have always a non-zero error probability with any finite SNR (in particular when no error control codes are yet adopted), independently of the physical layer bit rate (refer to our earlier analyses)
Shannon spectral efficiency vs. PAM/PSK/QAM, cont’d

Despite the previously acknowledged philosophical difference, meaningful comparisons can anyway be done as follows:

- set (reasonably small) target error probability for PAM/PSK/QAM and choose the alphabet and bandwidth that yield the target bit rate
- determine the corresponding SNR or Eb/N0 ratio that provides the target bit error probability
- compare this SNR or Eb/N0 ratio to the corresponding lower limit obtained from the Shannon capacity law, such that the Shannon capacity matches the target bit rate
- difference tells how far away we are from the Shannon bound

Example:

- Suppose we want to reach 40 Mbps bit rate over 10 MHz bandwidth in an I/Q modulated PSK/QAM system
- the spectral efficiency is thus 4 bits/s/Hz, and therefore we choose 16QAM
- theoretical minimum SNR, from Shannon law to reach the above spectral efficiency is $2^4 - 1 = 15 = \sim 11.7$ dB
- assuming that we want to achieve a reliability (symbol error prob.) of $10^{-6}$ in the practical 16QAM system, we need an SNR (see the earlier curves) of around 21 dB
- thus, the gap to the Shannon bound is close to 10 dB in this example! (well, 9.3 dB more exactly)
- this gap is something that we can then reduce using the error control codes, discussed in detail in the latter part of the course!

Notice also that the exact gap naturally depends on the set error probability target (here $10^{-6}$), assumed in the comparison.

Maximum Shannon spectral efficiency and Eb/N0 ratio

As deduced earlier, the maximum spectral efficiency of any system is

$$v = \frac{C}{W} = \log_2 \left( 1 + \frac{S}{N} \right)$$

On the other hand, we can relate the SNR and the $E_b / N_0$ ratio as (see earlier slides)

$$\frac{S}{N} = \frac{E_b R_b}{N_0 W} = \frac{E_b}{N_0} \times \frac{R_b}{W}$$

Combining these two, and using the capacity $C$ as the bit rate $R_b$ we get

$$v = \frac{C}{W} = \log_2 \left( 1 + \frac{E_b}{N_0} \frac{C}{W} \right)$$

Thus, the maximum spectral efficiency and the $E_b / N_0$ ratio implicitly bound to each other through

$$v = \log_2 \left( 1 + \frac{E_b}{N_0} v \right)$$

From above, we can directly solve the following:

$$\frac{E_b}{N_0} = \frac{2^v - 1}{v}$$

Thus, if the target spectral efficiency $v$ is fixed, the above expression gives the smallest possible $E_b / N_0$ ratio that can facilitate the targeted spectral efficiency, with error probability zero!
Maximum Shannon spectral efficiency and \( E_b/N_0 \) ratio, cont’d

On the other hand, if the \( E_b/N_0 \) ratio is fixed, then the previous expression of the form

\[
v = \log_2 \left( 1 + \frac{E_b}{N_0} \right)
\]

defines implicitly the maximum achievable spectral efficiency.

Also, stemming from

\[
\frac{E_b}{N_0} = \frac{2^v - 1}{v}
\]

we can define a lower bound for the \( E_b/N_0 \) ratio when there is an infinitely large bandwidth available.

That is, when \( W \to \infty \) and thus \( v \to 0 \), we get (use L'Hospital's rule)

\[
\lim_{v \to 0} \frac{E_b}{N_0} = \lim_{v \to 0} \frac{2^v - 1}{v} = \ldots = \ln(2) \approx 0.693 \equiv -1.6 \text{ dB}
\]

This is the so called Shannon bound for the \( E_b/N_0 \) ratio, i.e., the Shannon limit for the energy efficiency.

No system can operate with lower \( E_b/N_0 \) ratio!

Thus, the lower limit for the bit energy \( E_b \) in any system is

\[
E_b \geq 0.693 \times N_0 \quad \text{[Joules]}
\]

Maximum Shannon spectral efficiency and \( E_b/N_0 \) ratio, cont’d

The following figure illustrates the maximum spectral efficiency as a function of the \( E_b/N_0 \) ratio

- All practical systems operate in the region below the curve
- Grey region: Bit rate would be larger than Shannon limit, spectral efficiency too high (cannot be reached) for the given \( E_b/N_0 \) ratio
- White (dash-dotted) region: Feasible operation region of all practical systems where the bit rate and spectral efficiency are below the Shannon bound.
Maximum Shannon spectral efficiency and $\text{Eb/N0}$ ratio, cont’d

In the other extreme, when the spectral efficiency tends towards infinite (i.e., $v \to \infty$), we get (you can again use L’Hospital’s rule)

$$\lim_{v \to \infty} \frac{E_b}{N_0} = \lim_{v \to \infty} \frac{2^v - 1}{v} = \ldots = \infty$$

Thus, infinitely large spectral efficiency needs infinitely large $E_b / N_0$ ratio – an observation which is not very surprising in the end.

However, this also shows that $\lim_{v \to \infty} v = \infty$ even though the expression

$$v = \log_2 \left( 1 + \frac{E_b}{N_0} v \right)$$

cannot be solved in closed-form. This is because $\log_2(.)$ is a monotonic and increasing function, hence infinite $E_b / N_0$ ratio must imply infinite $v$.

Finally, notice that the sensitivity of the maximum achievable spectral efficiency $v$ vs. the needed $E_b / N_0$ ratio is quite different in different ranges of the $E_b / N_0$ ratio (see the previous page figure)

- at very small spectral efficiencies and $E_b / N_0$ ratios, the relative sensitivity is high
- while at larger spectral efficiency values, the relative sensitivity starts to slow down

8. DIGITAL FREQUENCY MODULATION

Next, we study carrier modulation methods that can be utilized as an alternative approach compared to the I/Q modulated PAM/PSK/QAM systems, primarily digital frequency modulation.

There are different variants of digital frequency modulation, particularly:

- FSK Frequency Shift Keying
- CPFSK Continuous-Phase FSK
- MSK Minimum Shift Keying
- GMSK Gaussian Minimum Shift Keying

These differ significantly from the previously discussed linear digital modulation methods, e.g., Nyquist pulse shaping is not used in a similar way as was done with I/Q modulated PAM/PSK/QAM (except for the GMSK case, kind of)

- or more accurately said: the baseline pulse shape is a rectangular pulse of the same duration as the symbol interval, $T$
- only in case of GMSK, a longer and smoother pulse is adopted
Frequency Shift Keying, FSK, principle

The starting point is the same as before:

- Consecutive bits are represented as higher level symbols
- With an alphabet size of $M$, we can support $\log_2(M)$ bits per symbols

Now, the principle of creating the actual carrier modulated waveform is such that for representing $M$ different symbols we use $M$ different instantaneous frequencies

- (for reference: with carrier modulated PAM/PSK/QAM, different combinations of carrier phase and amplitude were used)
- thus, intuitively, there is large similarity to analog FM
- the envelope of the modulated signal is always constant, information is purely in the zero crossings

The simplest case is the binary FSK, in which 0 and 1 correspond to two different instantaneous frequencies:

\[ \begin{array}{c}
\text{0} \quad \text{1} \\
0 \quad 1 \quad 0 \quad 1 \quad 1
\end{array} \]

\[ T \]

\[ t \]

NB: The bit rate still reads (why?):

\[ R_b = \log_2(M) \times \frac{1}{T} \]

where $M$ is the symbol alphabet size (here the amount of different pulses) and $1/T$ is the symbol rate.

Properties of FSK, overview

FSK advantages:

- Non-coherent detection is possible $\rightarrow$ carrier synchronization is not required (compare to the analog FM)
  $\rightarrow$ Easy to implement
    - In some cases the carrier synchronization can be extremely hard to implement and achieve (e.g. quickly varying carrier phase in the channel), and therefore, non-coherent detection would be the only feasible solution.
- Immune to some non-linearities: no information in the envelope, so even hard limiting can be done, the information lies in the zero crossings (again compare to the analog FM)
  $\rightarrow$ very non-linear power amplifiers (PAs) can be used
  $\rightarrow$ improved power efficiency from the PAs perspective

This can be significant, e.g. in cellular uplink and satellite communications systems
  $\rightarrow$ interestingly, some of the first cellular systems adopted digital frequency modulation, most notably GSM cellular networks

FSK disadvantages:

- To achieve the same bit error rate with the carrier modulated PAM/PSK/QAM typically 3 dB larger S/N ratio is required (we will shows this later in the notes)
- With conventional FSK the spectral efficiency is poor compared to the PAM/PSK/QAM (MSK and CPM already closer to PAM/PSK/QAM).
- Channel equalization is difficult because of the nonlinear nature of the mapping from bits to RF waveform (cf. again analog FM)
- FSK is also somewhat difficult to be analyzed because of the nonlinear nature
Continuous phase property and pulse orthogonality

The phase behavior of the FSK signal at symbol boundaries can be continuous or non-continuous:

Continuous phase is naturally better, since
- The essential bandwidth is in this case smaller (discontinuities produce always high frequencies)
- Performs better in case of non-linearities in the transmission chain
- This leads to the continuous-phase FSK (CPFSK) principle

In FSK a continuous phase can be always achieved when every pulse includes an integer number of its own cycles or periods:

\[ f_i T = K_i, \quad i = 1, 2, \ldots, M \]

where \( K_i \) are integers. Here \( f_1, f_2, \ldots, f_M \) are the pulse frequencies, and \( T \) is the symbol duration which is here the same as the pulse length.

To minimize the bandwidth, it seems reasonable to minimize the frequency differences of the pulses while maintaining the phase continuity. In case of binary FSK the frequencies fulfill the condition:

\[ f_1 T = K_1 \quad f_2 T = K_2 \]

Here the frequency separation is the smallest possible when \(|K_1 - K_2| = 1\), i.e.,

\[ f_2 - f_1 = 1/T \]

More generally, with \( M \) frequencies, we can choose:

\[ f_i - f_{i-1} = 1/T \quad i = 2, \ldots, M \]

Continuous phase property and pulse orthogonality (cont’d)

Another perspective for designing the FSK pulses (their frequencies) is the pulse orthogonality:
- the frequencies can be selected so that the different pulses form a set of orthogonal functions
- in other words, the (cross)correlation between different pulses, say \( g_i(t) \) and \( g_j(t) \), is zero, i.e.

\[ \int_{-\infty}^{\infty} g_i(t)g_j(t)dt = 0 \quad \text{when} \quad i \neq j \]

This makes the detection task at the receiver easier
- Correlation receiver principle can be used to detect the signal:
  - correlate the received signal against all possible \( M \) pulses and choose the one that maximizes the correlation
  - we will talk about this more, soon

The previously presented frequency design principle of the form

\[ f_i - f_{i-1} = 1/T \quad i = 2, \ldots, M \]

produces also orthogonal pulses with a small additional specification for the first used pulse frequency \( f_1 \) (or alternatively, for the center point of two adjacent pulse frequencies)
- we’ll show this explicitly in the classroom exercises
FSK signal spectral characteristics, shortly

Exact spectral analysis of FSK type signals is somewhat challenging.

On the other hand, any individual FSK pulse is nothing but a truncated sinusoidal, whose spectrum is trivial to analyze (right?)

- sinc-shaped spectrum around the oscillating frequency
- the result is illustrated in the figure below (only positive frequencies shown, for simplicity) for one individual pulse
- below, \( w(t) \) is referring to a truncation function (square pulse of width \( T \))

Thus, if we now assume that the frequency separation of the neighboring pulse frequencies, \( |f_i - f_{i-1}| \), is \( 1/T \) (like sketched before), the essential bandwidth used by the total signal composed of a sequence of such pulses is in principle of the form

\[
W \simeq M \times \frac{1}{T}
\]

where \( M \) is the alphabet size (amount of different pulses)

- this is fundamentally different compared to I/Q modulated PAM/PSK/QAM for which \( W = (1 + \alpha) \times \left( \frac{1}{T} \right) \), independently of \( M \)
- for FSK, the bandwidth depends on \( M \)!

FSK, spectral efficiency

Based on the previous results regarding the (approximate) bandwidth and the achievable bit rate, we can directly express the physical layer spectral efficiency of FSK as

\[
\frac{R_b}{W} \simeq \frac{\log_2(M) \times \frac{1}{T}}{M \times \frac{1}{T}} = \log_2(M) \quad \text{bits/s/Hz}
\]

Notice that this is fundamentally different compared to the spectral efficiency of \( \log_2(M) \) obtained with I/Q modulated PAM/PSK/QAM

- spectral efficiency of FSK decreases for larger \( M \)!
Minimum Shift Keying, MSK

In this case, the frequency separation is only half of the baseline FSK, in other words

\[ f_i - f_{i-1} = \frac{1}{2T} \quad i = 2, \ldots, M \]

This, of course, decreases the required bandwidth directly.

Also pulse orthogonality can still be achieved (try it, we can also check it in the classroom exercises), so the performance of, e.g., the correlation receiver is not affected by the reduced frequency differences

- but what about the continuous phase aspect?

Example: Binary MSK

- When directly applied, the MSK-principle produces the following types of pulses:
  - Phase is clearly not continuous anymore
  - But there is an easy work-around, see next page

\[ g_i(t) \]

\[ g_i(t) \]

\[ g_i(t) \]

However, the continuous phase can still be achieved by choosing the sign (±) of the pulse \( g_i(t) \) always in an appropriate manner

- illustrated below
- again this does not change the orthogonality of the pulses
- notice that this creates also additional memory to the transmitted waveform
  - the exact transmitted pulse depends on not only the current bit to be transmitted but also on the previous bits (because of the sign of the previous pulse)
  - this is something that can, in principle, utilized at the receiver side, to enhance the reliability of the detection
MSK, spectral efficiency

Based on the previous, with MSK the frequency separation of the neighboring pulses, \( |f_i - f_{i-1}| \), is \( 1/(2T) \).

Thus, for an alphabet size of \( M \), the overall bandwidth is essentially of the form (approximately)

\[
W \approx M \times \frac{1}{2T}
\]

Thus, the spectral efficiency of MSK reads

\[
\frac{R_b}{W} \approx \frac{\log_2(M) \times \frac{1}{T}}{M \times \frac{1}{2T}} = \frac{2\log_2(M)}{M} \quad \text{bits/s/Hz}
\]

This is still substantially different to I/Q modulated PAM/PSK/QAM, \( (\log_2(M)) \) though with small values of \( M \), the difference is not very large.

Some numerical spectral efficiency values are illustrated below.

<table>
<thead>
<tr>
<th>PAM/PSK/QAM</th>
<th>FSK</th>
<th>MSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log_2(M) )</td>
<td>( \frac{\log_2(M)}{M} )</td>
<td>( \frac{2\log_2(M)}{M} )</td>
</tr>
<tr>
<td>( M = 2 )</td>
<td>1 bits/s/Hz</td>
<td>0.5 bits/s/Hz</td>
</tr>
<tr>
<td>( M = 4 )</td>
<td>2 bits/s/Hz</td>
<td>0.5 bits/s/Hz</td>
</tr>
<tr>
<td>( M = 8 )</td>
<td>3 bits/s/Hz</td>
<td>0.375 bits/s/Hz</td>
</tr>
<tr>
<td>( M = 16 )</td>
<td>4 bits/s/Hz</td>
<td>0.25 bits/s/Hz</td>
</tr>
</tbody>
</table>

Binary MSK, alternative interpretation and implementation

By definition, MSK stems from digital frequency modulation where the frequency difference of the neighboring pulse frequencies is half the symbol period.

It is, however, interesting to observe that in the binary MSK case, one can obtain the very same pulses through a completely alternative path/logic as follows:

- start from basic QPSK waveform (this has nothing to do with frequency modulation) but use
  - symbol interval of \( 2T \)
  - half the cycle of a cosine wave whose frequency is \( 1/(4T) \) as the pulse-shape
- then, delay the quadrature (Q) component by half the symbol period, meaning by \( T \)
- you’ll see that you will get essentially binary MSK pulses, assuming that the bit to symbol mapping is done correctly

This goes under the name offset-QPSK, which we already addressed earlier from the PAPR limitation perspective,

- however, when combined in the QPSK case with the specific pulse shape and proper mapping of the bits to symbols/pulses, it will merge with binary MSK

This is, generally speaking, just a "strange special case" – in general, PSK and MSK are completely different digital modulation methods.
Continuous phase frequency modulation, CPM, in general

Continuous phase frequency modulation can be practically implemented by FM modulating the carrier with a real baseband signal where a proper pulse shape $g(t)$ is used:

$$ S(t) = \sum_{m=-\infty}^{\infty} A_m g(t - mT) $$

$$ X_c(t) = K \cos \left[ 2\pi f_c t + 2\pi f_{\Delta} \int_{-\infty}^{t} S(\tau) d\tau \right] $$

This can be performed using e.g. a Voltage Controlled Oscillator, VCO:

Here the phase continuity is preserved by-design.

The special case where $g(t)$ is a rectangular pulse corresponds to the continuous-phase FSK (CPFSK) case.

In general, other types of pulse shapes besides the rectangular one, can improve the performance with respect to the bandwidth.

- A smoother pulse shape reduces the bandwidth of $S(t)$ and therefore, that of the carrier modulated signal, even though there is no linear mapping between them
- One example, called GMSK, is given next page

Notice that with longer pulses $g(t)$, the instantaneous frequency of the final output signal at any specific time instant does not anymore only depend on the currently signaled bit/symbol but also on the earlier ones

- ISI type of phenomenon, but in controlled manner
- This is made on purpose in order to reduce the bandwidth
- On the other hand, detection becomes more difficult
  - deliberate ISI must be taken into account at RX

Gaussian Minimum Shift Keying, GMSK

This is a special variation of MSK/CPM principle, in which a better pulse shape is used instead of the rectangular one. Here, the pulse shape $g(t)$ is a Gaussian lowpass filter where the term comes from the fact that the pulse shape resembles Gaussian probability density function.

In the following figures $B_0 T$ is the normalized 3dB bandwidth of the pulse/filter. This parameter defines the pulse width and the 3dB bandwidth of the modulated signal. The MSK is given as a special case.

This GMSK principle is applied in the GSM network with $W T_b = 0.3$
CORRELATION RECEIVERS FOR FSK & MSK

In the following, we address more specifically the receiver processing in FSK and MSK type of systems, and particularly the so-called correlation receiver principle to detect the transmitted bits or symbols

- the receiver correlates the received signal against all the different possible frequency modulated pulses, within every symbol duration, and chooses the maximum

We also address then how reliably the correlation receiver works under an additive white Gaussian noise (AWGN) type of channel

- i.e., we determine the symbol error probability of the correlation receiver for FSK and MSK

Notation in the continuation is that the individual FSK or MSK pulses are denoted with

\[ g_1(t), g_2(t), \ldots, g_M(t) \]

where \( M \) refers to the modulation order/alphabet size.

In other words, within any individual symbol duration, one of these pulses has been transmitted and the receiver is trying to figure out that which one.

For notational convenience, we assumed that the pulses have been normalized to have unit energy, i.e.,

\[ \forall m : \int_{-\infty}^{\infty} g_m^2(t) dt = 1 \]

(...this is only to simplify some notations in the continuation and in the analysis, does not impact any insight or conclusions)

Correlation receiver principle

Based on the previous reasoning, the basic structure of the correlation receiver is of the form

\[ R(t) = g_m(t) + N(t) \]

where \( g_m(t) \) is one of the possible pulses \( \{g_1(t), g_2(t), \ldots, g_M(t)\} \).

Correlation receiver calculates now the following correlations

\[ K_i = \int_{-\infty}^{\infty} R(t) g_i(t) dt \quad i = 1, \ldots, M \]

and chooses the one that is largest

- intuitive - isn’t it?
- we’ll address the resulting performance, in terms of the detection error probability, soon
Matched filter concept

One possible way to implement the calculation of the previous correlations is to adopt the so called matched filtering based processing:

Proof:

\[ K_i = \int_{-\infty}^{\infty} R(t)g_i(t)dt = \int_{-\infty}^{\infty} R(\lambda)g_i(\lambda - t)d\lambda \bigg|_{t=0} \]
\[ = \int_{-\infty}^{\infty} R(\lambda)h_i(t - \lambda)d\lambda \bigg|_{t=0} \]
\[ = [R(t) * h_i(t)]_{t=0} \]

where \( h_i(t) = g_i(-t) \) \( i = 1, \ldots, M \) refer to the mirrored pulses.

- in a real implementation, the filters (mirror pulses) would be delayed by \( T \), to be causal, and hence also the sampling is delayed accordingly

Notice that when processing and detecting a train of received pulses, the above structure works directly as long as you take samples from the matched filters’ outputs at integer multiples of \( T \) (instead of just one sample at relative time zero).

Examples of matched filters, binary FSK

Binary FSK, the original pulses:

\[ g_1(t) = \sin(\omega_1 t)w(t) \quad w(t) = \begin{cases} 1 & 0 \leq t \leq T \\ 0 & \text{muuloin} \end{cases} \]
\[ g_2(t) = \sin(\omega_2 t)w(t) \]

The corresponding matched filters are:

\[ h_1(t) = g_1(-t) = \sin(-\omega_1 t)w(-t) \]
\[ h_2(t) = g_2(-t) = \sin(-\omega_2 t)w(-t) \]

Physically realizable matched filters are delayed version of above:

\[ h'_1(t) = h_1(t - T) = g_1(T - t) = \sin(\omega_1(T - t))w(T - t) \]
\[ h'_2(t) = h_2(t - T) = g_2(T - t) = \sin(\omega_2(T - t))w(T - t) \]
Matched filtering cont’d

In general, the matched filtering and correlation receiver concepts are very universal and fundamental processing principles in communications receivers

- here we introduced this only in the FSK context, but later we will revisit very similar type of processing also in case of other digital modulation methods and the corresponding receivers

Believe or not, also e.g. the earlier established receivers for baseband PAM which utilize

- receiver filtering, sampling and slicing

as well as those of I/Q modulated PAM/PSK/QAM adopting

- I/Q downconversion, receiver filtering, sampling and slicing

can be interpreted to be certain forms of correlation receivers and reflect matched filtering based processing.

We will naturally come back to this later

Correlation receiver - signals and noise

Under additive noise in the channel, the correlator outputs read

\[
K_i = \int_{-\infty}^{\infty} R(t)g_i(t)dt = \int_{-\infty}^{\infty} (g_m(t) + N(t))g_i(t)dt
= \int_{-\infty}^{\infty} g_m(t)g_i(t)dt + \int_{-\infty}^{\infty} N(t)g_i(t)dt
= s_m,i + N_i \quad i = 1, \ldots, M
\]

where

\[
s_m,i = \int_{-\infty}^{\infty} g_m(t)g_i(t)dt \quad \text{and} \quad N_i = \int_{-\infty}^{\infty} N(t)g_i(t)dt \quad i = 1, \ldots, M
\]

For the actual detection and the corresponding error probability analysis purposes, we can collect the results of the \(M\) correlations into an \(M\)-dimensional vector \(\mathbf{K} = [K_1, K_2, \ldots, K_M]^T\) and write

\[
\mathbf{K} = \mathbf{S}_m + \mathbf{N}
\]

where, \(\mathbf{S}_m = [s_{m,1}, s_{m,2}, \ldots, s_{m,M}]^T\) is the so called signal vector and \(\mathbf{N} = [N_1, N_2, \ldots, N_M]^T\) is the corresponding noise vector.

Now, with unit energy and orthogonal FSK pulses, the different signal vectors \(\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_M\) are of the form (why ?)

\[
\begin{align*}
\mathbf{S}_1 &= \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, & \mathbf{S}_2 &= \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, & \ldots, & \mathbf{S}_M &= \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}
\end{align*}
\]
Correlation receiver - signals and noise, cont’d

Example: Binary ($M=2$) FSK with unit energy and orthogonal pulses yield

\[
S_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

Example: Ternary ($M=3$) FSK with unit energy and orthogonal pulses yield

\[
S_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad S_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}
\]

The dimensionality of the space spanned by these orthonormal (orthogonal and unit length) signal vectors thus depends directly on the number of the different pulses, $M$.

- in case of PAM/PSK/QAM, on the other hand, the essential signal space is either 1-dimensional (real axis; space of real-valued numbers) or 2-dimensional (real and imaginary axes, complex plane).
- thus, FSK is very much different class of waveforms, also in this sense.

We’ll come back, later in the course, to the more general concepts of signal space, and associated optimum receivers for different types of digitally modulated signals.

- here, our primary purpose is just to be able to quantify the detection error probability of correlation based FSK receiver.

Next, how about the noise $N_i$ at the correlator outputs $K_i$?

\[
K_i = \int_{-\infty}^{\infty} R(t)g_i(t)dt = \int_{-\infty}^{\infty} (g_m(t) + N(t))g_i(t)dt = s_{m,i} + N_i \quad ; i = 1, \ldots, M
\]

where

\[
N_i = \int_{-\infty}^{\infty} N(t)g_i(t)dt \quad ; i = 1, \ldots, M
\]

Assuming, like usually, that the actual channel noise $N(t)$ is
- white, spectral density is constant $N_0$
- and Gaussian distributed

we can easily deduce that the noise variables $N_i$ at the correlator outputs

- are all Gaussian distributed
- all have variance $E[N_i^2] = \sigma^2 = N_0 \quad \forall i$
- all are mutually uncorrelated $E[N_iN_j] = 0 \quad \forall i \neq j$
- \ldots and are therefore also statistically independent

Proof:

\[
E[N_iN_j] = \int_{-\infty}^{\infty} N(t)\text{white} dt = N_0 \int_{-\infty}^{\infty} g_i(t)g_j(t)dt = \ldots
\]
Correlation receiver, error probability

As a concrete example, we consider the binary FSK with unit energy and orthogonal pulses for which the signal vectors read

\[
S_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

Assuming that the actual detector uses again the minimum distance detection rule, the space of the possible correlator output vectors is divided into the following two decision regions:

Based on our noise statistics on the previous slide, and on the error probability analysis principles discussed in the context of PAM/PSK/QAM already, the detection error probability now reads (here \(d = \sqrt{2}\)):

\[
P(\text{symbol error, binary FSK}) = Q\left(\frac{d}{2\sigma}\right) = Q\left(\frac{\sqrt{2}}{2\sigma}\right) = Q\left(\frac{1}{\sqrt{2}\sigma}\right)
\]

As a comparison, for binary PAM (2-PAM) or BPSK, assuming an alphabet \(\{\pm 1\}\) such that the signal energies are comparable, we have

\[
P(\text{symbol error, binary PAM}) = Q\left(\frac{d}{2\sigma}\right) = Q\left(\frac{1}{\sigma}\right) = Q\left(\frac{1}{\sqrt{\sigma^2}}\right)
\]

Thus, based on above, binary FSK is 3dB worse in terms of the needed received signal power, compared to PAM/BPSK, assuming that the noise powers are identical in the two systems.

Correlation receiver, error probability, cont’d

Previous result holds for binary FSK, assuming orthogonal pulses

- now one may ask: what if the pulse frequencies are chosen such that the pulses are not orthogonal?

Let’s again assume binary FSK but now indeed with non-orthogonal pulses \(\{g_1(t), g_2(t)\}\) such that the correlation between the pulses is denoted with \(\rho\).

In this case, the signal vectors read (why?)

\[
S_1 = \begin{bmatrix} 1 \\ \rho \end{bmatrix}, \quad S_2 = \begin{bmatrix} \rho \\ 1 \end{bmatrix}
\]

The situation is illustrated below, in cases with \(\rho > 0\) (left) and \(\rho < 0\) (right).

Now, the detection error probability reads

\[
P(\text{symbol error, non-orthogonal binary FSK}) = Q\left(\frac{d}{2\sigma}\right) = Q\left(\frac{\sqrt{2}(1 - \rho)}{2\sigma}\right) = Q\left(\frac{(1 - \rho)^2}{2\sigma^2}\right)
\]

In the special case with \(\rho = -1\), we have actually transferred from orthogonal frequency modulation to binary phase modulation

- with \(\rho = -1\), the pulses would only have 180 degree phase difference
**M-level orthogonal FSK, error probability**

The symbol error probability can also be analyzed for arbitrary orthogonal M-FSK.

The results are illustrated below (from the course book, p. 95 there is probably a typo) for different values of $M$ as a function of $E_s/N_0$ ratio:

- here $E_s = ST$ refers to symbol energy, i.e., signal power $S$ times the symbol duration $T$

![Graph showing error probability for different values of M as a function of 10 log base 10 of E_s/N_0 ratio.]

**Correlation receivers and MSK**

In case of MSK, there are always two different pulses $\pm g_m(t)$ available, to represent an individual symbol:

- proper pulse sign selected such that phase continuity is maintained (see earlier slides)

On the receiver side, a simple and straightforward (but strictly-speaking sub-optimum) approach is to remove the sign information and process the received signal as follows:

![Diagram illustrating the correlation receiver and MSK process.]

In case of binary MSK, the detection error probability of the above receiver is essentially very similar to binary FSK.

However, the detection error probability can be further reduced by taking the memory in the MSK signal into account:

- some added complexity but better performance
- performance wise, then closer to BPSK
- practical tools in the form of sequence detectors and Viterbi algorithm, to incorporate the memory in the detection as discussed later in the course
9. ML & MAP FRAMEWORKS IN ESTIMATION & DETECTION

In this part of the course, we shortly address the fundamentals of estimation and detection theory and its applications in digital communications, i.e. how to process the received noisy and possibly distorted signal

- to make as reliable decisions of transmitted bits or symbols as possible
- also to estimate certain essential parameters such as channel response, symbol timing or carrier frequency offset

The presentation builds on the general Maximum Likelihood (ML) and Maximum A Posteriori (MAP) frameworks

- First defining what do these mean conceptually
- And then addressing in particular how to devise actual signal processing solutions (practical decision rules, estimators, etc.) in different elementary scenarios

Particularly interesting concrete applications are

- optimal ML and MAP detection of single bit or single symbol in AWGN
- optimal ML sequence detector (MLSD), and the Viterbi algorithm, to handle ISI in practical systems (treated a little later)
- efficient decoding of certain error control codes, particularly convolutional codes and Turbo codes (treated later in the course)

The underlying ML and MAP frameworks build on probabilistic thinking

- Received signal has always random elements (uncertainty, e.g. noise & interference)
- But also the interesting quantities (e.g. transmitted bits and symbols) can be modeled in probabilistic manner
- The (assumed) distribution of the received signal is the key, and how does that depend on the quantity/quantities of interest

---

Terminology: Estimation and Detection

Estimation and detection refer to a process where one seeks to determine the value of some interesting quantity based on

1) Some observation, in our case typically the received signal
2) Other potential side-information (e.g. potential prior probabilities of transmitted symbols)

The distinction between estimation and detection typically comes from the more distinct nature of the quantity of interest:

- **Estimation**: The quantity of interest is of continuous-valued nature (e.g. unknown carrier phase or symbol timing, or the channel impulse response values)
- **Detection**: The quantity of interest is of discrete-valued nature (e.g. transmitted bit or symbol)

In case of detection, one thus “only” needs to decide or make a decision from a finite set of alternative candidate values

- Single bit or symbol (e.g. QPSK, 16QAM, …): binary alphabet, multilevel alphabet
- Finite-length bit sequence or symbol sequence: among the different possible sequences (still finite set)

In general, estimation and detection/decision making have wide applications in all engineering and other fields of science, while we primarily focus here on digital communications systems and devices perspectives
Estimation and Detection (cont’d)

In any elementary digital communication chain, the transmitted bits or symbols are subject to various elementary processes of the following forms:

- **Deterministic processing** (signal generation). This includes e.g. filtering, modulation and coding.
- **Stochastic/random processing** (noise generation). This includes e.g. thermal noise, interference, and fading

In general, the receiver observes $Y$ and uses this, and other possible side-information, to estimate or detect the value of the interesting quantity, say $X$.

The two general frameworks to carry out the above task are:

- **ML** maximum likelihood
- **MAP** maximum a posteriori

We’ll elaborate these in more details below, in different elementary signal generation and noise generation models and scenarios.

Notation: In the continuation, if the interesting quantity or parameter is say $\theta$, the estimator or detector output is denoted with $\hat{\theta}$

- this is what we have been using already for, e.g. symbol decisions earlier

---

Short recap on elementary probabilities and distributions

**Discrete-valued random variables**

- $X$, discrete set of possible values $\Omega_X = \{x_1, x_2, \ldots, x_K\}$;
- $Y$, discrete set of possible values $\Omega_Y = \{y_1, y_2, \ldots, y_L\}$, …
- Probabilities $p_X(x_k), p_Y(y_l), \ldots$
- Joint probabilities $p_{X,Y}(x_k, y_l)$
- Conditional probabilities (Bayes rule) $p_{X|Y}(x_k \mid y_l) = \frac{p_{X,Y}(x_k, y_l)}{p_Y(y_l)}$
- Statistical expectation/mean
  
  $E[X] = \sum_{x \in \Omega_X} x p_X(x)$, $E[X^n] = \sum_{x \in \Omega_X} x^n p_X(x)$, $E[g(X)] = \sum_{x \in \Omega_X} g(x) p_X(x)$

**Continuous-valued random variables**

- $X$, continuum of values $\Omega_X$; $Y$ continuum of values $\Omega_Y$, …
- Probability densities/distributions $f_X(x), f_Y(y), \ldots$
- Joint distribution $f_{X,Y}(x, y)$
- Conditional distribution (Bayes rule) $f_{X|Y}(x \mid y) = \frac{f_{X,Y}(x, y)}{f_Y(y)}$
- Statistical expectation/mean
  
  $E[X] = \int_{\Omega_X} x f_X(x) dx$, $E[X^n] = \int_{\Omega_X} x^n f_X(x) dx$, $E[g(X)] = \int_{\Omega_X} g(x) f_X(x) dx$
Recap: Gaussian distribution

A real-valued scalar random variable $X$ is called a Gaussian (or normal distributed) when the probability density reads

$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-m)^2/2\sigma^2}$$

where $m$ and $\sigma^2$ are the mean and variance.

The Gaussian probability density function is symmetric with respect to its mean value, as illustrated below.

For a complex-valued Gaussian random variable $X = X_I + jX_Q$, assuming uncorrelated and equal-variance I and Q components (mean $m$ and variance $\sigma^2$, for both the I and Q) the corresponding probability density expression is

$$p_X(x) = \frac{1}{2\pi\sigma^2} e^{-(x-m)^2/2\sigma^2}$$

ML AND MAP DETECTION, IN GENERAL

In the following, we will formulate the elementary ML and MAP detectors, in general, and then see to what kind of practical processing these lead to in selected example scenarios.

For arbitrary discrete-valued parameter $\theta$ and continuous-valued observation $y$, the fundamental definitions are

ML-detector:

$$\hat{\theta}_{ML} = \arg\max_{\theta \in \Theta} f_Y(y | \theta)$$

- In words: choose the value of $\theta$ which makes the given observation $y$ most likely or most probable
- $f_Y(y | \theta)$ denotes the conditional distribution of the observation $Y$ for given $\theta$; called also likelihood function

MAP-detector:

$$\hat{\theta}_{MAP} = \arg\max_{\theta \in \Theta} p_{\theta | Y}(\theta | y)$$

- In words: choose the value of $\theta$ which is most probable for given observation $y$
  - Intuitive! (why?)
  - For reference: think how would you formulate the detection problem if you would not have any observation $y$ at all ...
- $p_{\theta | Y}(\theta | y)$ denotes the conditional probability of the parameter $\theta$ for given observation $y$
- These are also called a-posteriori probabilities of $\theta$
  - opposed to prior probabilities $p_\theta(\theta)$

...these look, perhaps, a little abstract but we’ll get to more concrete stuff soon
SINGLE-BIT OR SINGLE-SYMBOL DETECTION

We consider a principal system model where the input is a discrete-valued random variable (bit or symbol) $A$ belonging to alphabet $\Omega_A$.

System output or observation in receiver is another random variable $Y$, generally continuous-valued, which is somehow statistically dependent on $A$ (why?)

This dependence is here described through the conditional distributions (one for each value of $A$)

$$f_{Y|A}(y|a), \quad a \in \Omega_A$$

Then:

ML-detector chooses that symbol $\hat{a}_{ML} \in \Omega_A$ which maximizes the conditional distribution (likelihood) $f_{Y|A}(y|a)$ for given observation $y$:

$$\hat{a}_{ML} = \arg \max_{a \in \Omega_A} f_{Y|A}(y|a)$$

MAP-detector, in turn, chooses that symbol $\hat{a}_{MAP} \in \Omega_A$ which maximizes the a-posteriori probability $p_{A|Y}(a|y)$ for given observation $y$:

$$\hat{a}_{MAP} = \arg \max_{a \in \Omega_A} p_{A|Y}(a|y)$$

The latter one is more intuitive (why?).

It also maximizes the average probability of correction decision (and thus minimizes the average probability of error), shown explicitly soon.

ML and MAP detectors have something in common

Based on the Bayes rule, the conditional probabilities or distributions are generally linked through

$$p_{A|Y}(a|y) = \frac{f_{Y|A}(y|a)p_{A}(a)}{f_Y(y)}$$

In decision making, the denominator is independent of the decision candidates (why?)

- Hence in MAP detection, we can maximize the numerator

Thus we can write:

$$\hat{a}_{ML} = \arg \max_{a \in \Omega_A} f_{Y|A}(y|a)$$

Hence, the difference is that while the ML detector builds fully on the likelihood function (trusts the observation only), the MAP detector deploys also the prior probabilities $p_{A}(a)$ of the symbols.

Many times, on the other hand, the only reasonable assumption may be that all symbols are equally likely

- If this is the case, the ML and MAP detectors give identical results (why?)

- But notice anyway the fundamental difference behind them:
  - relying only on observation vs. deploying also prior information
Average probability of correct decision and MAP detector optimality

Average probability of correct decision reads

\[ \text{Prob.}[\text{correct decision}] = \int_{y \in \Omega_Y} p_{A|Y}(\hat{a} | y)f_Y(y)dy \]

\[ = \int_{y \in \Omega_Y} \text{Prob.}[A = \hat{a} | Y = y]f_Y(y)dy \]

\[ = \int_{y \in \Omega_Y} p_{A|Y}(\hat{a} | y)f_Y(y)dy \]

As the probability density \( f_Y(y) \) is always, by definition, non-negative, the value of the above integral is maximized when the probabilities \( p_{A|Y}(\hat{a} | y) \) are maximized.

This is, by definition, what MAP detector does.

Hence, we can say:

MAP detector maximizes the average probability of correct decisions and thus minimizes the average detection error probability.

Like already concluded previously, if all symbols are equally likely, then ML detector makes identical decisions

- In that case, also ML detector minimizes the error probability

Single symbol detection in AWGN channel, \( Y = A + N \)

Observation is now assumed to be of the form \( Y = A + N \) where \( N \) denotes additive Gaussian noise variable, variance \( \sigma^2 \), mean 0.

Example: 2-PAM alphabet, \( A \in \{-a,+a\} \). Then likelihood functions read

\[ f_{Y|A}(y|+a) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(y-a)^2}{2\sigma^2}\} \]

\[ f_{Y|A}(y|-a) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(y+a)^2}{2\sigma^2}\} \]

ML-detector thus corresponds to a threshold-device with the threshold at the intersection of likelihood functions (here origin)

- This corresponds to the intuitive minimum distance principle

MAP-detector deploys weighted likelihood functions with symbol prior probabilities as weights. If, e.g., \( p_A(+a) > p_A(-a) \), they look like:

MAP-detector is thus also a threshold-device but the threshold is generally different/shifted compared to ML detector

- Intuition?
Single symbol detection in AWGN channel (cont’d)

Another example: 4-PAM alphabet, $A \in \{-3a,-a,+a,+3a\}$.

ML-detector likelihood functions and the corresponding decision regions and thresholds are illustrated below

- Leads again to the minimum distance principle

[Graph showing likelihood functions and decision regions]

The corresponding weighted likelihoods (for some example priors from the hat) of the MAP-detector, and the corresponding decision regions and thresholds are as below

- Here illustrated for selected example priors
- Notice again the shifts in thresholds compared to the ML detector

[Graph showing weighted likelihood functions and decision regions]

ML and MAP detectors under additive Gaussian noise

Based on previous examples, ML detector in AWGN corresponds to the minimum distance detector (independently of noise variance) while the MAP detector decision regions depend in general on noise variance and prior probabilities of different symbol values (see previous example).

More formally, we can write these as (assuming real-valued variables):

- **ML**
  
  $$
  \hat{a}_{ML} = \arg\max_{a \in \Omega_a} f_{Y|A}(y|a) 
  = \arg\max_{a \in \Omega_a} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y-a)^2}{2\sigma^2}\right\}
  = \ldots
  = \arg\min_{a \in \Omega_a} (y-a)^2
  $$

- **MAP**
  
  $$
  \hat{a}_{MAP} = \arg\max_{a \in \Omega_a} f_{Y|A}(y|a) p_A(a)
  = \arg\max_{a \in \Omega_a} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y-a)^2}{2\sigma^2}\right\} p_A(a)
  = \ldots
  = \arg\min_{a \in \Omega_a} (y-a)^2 - 2\sigma^2 \ln(p_A(a))
  $$

This thus formally establishes the minimum distance detector (ML), while for MAP also the prior probabilities $p_A(a)$ and noise variance $\sigma^2$ play a role.
**Shorty on log-likelihood functions**

Many times, it’s more convenient to play with logarithm of likelihood function (ML), i.e.

\[
\ln\{ f_{Y|A}(y|a) \}
\]

or the corresponding logarithm of weighted likelihood function (MAP), i.e.

\[
\ln\{ f_{Y|A}(y|a) p_A(a) \}
\]

Why?

Because many expressions become simpler, and on the other hand logarithm is a monotonic function and thus not change anything related to which decision candidate will maximize the functions for any given observation.

So many times the detectors are written as:

\[
\hat{a}_{ML} = \arg \max_{a \in \Omega_A} f_{Y|A}(y|a) = \arg \max_{a \in \Omega_A} \ln\{ f_{Y|A}(y|a) \}
\]

\[
\hat{a}_{MAP} = \arg \max_{a \in \Omega_A} f_{Y|A}(y|a) p_A(a) = \arg \max_{a \in \Omega_A} \ln\{ f_{Y|A}(y|a) p_A(a) \}
\]

**Concept of log-likelihood ratio in binary decision problems**

In case of binary detection problems, say \( \Omega_A = \{ -1, +1 \} \), the detectors can also be expressed through the so called log-likelihood ratios or weighted log-likelihood ratios as

\[
\hat{a}_{ML} = +1 \quad \text{if} \quad \ln\{ \frac{f_{Y|A}(y|+1)}{f_{Y|A}(y|-1)} \} \geq 0
\]

\[
\hat{a}_{ML} = -1 \quad \text{if} \quad \ln\{ \frac{f_{Y|A}(y|+1)}{f_{Y|A}(y|-1)} \} < 0
\]

and

\[
\hat{a}_{MAP} = +1 \quad \text{if} \quad \ln\{ \frac{f_{Y|A}(y|+1)p_A(+1)}{f_{Y|A}(y|-1)p_A(-1)} \} \geq 0
\]

\[
\hat{a}_{MAP} = -1 \quad \text{if} \quad \ln\{ \frac{f_{Y|A}(y|+1)p_A(+1)}{f_{Y|A}(y|-1)p_A(-1)} \} < 0
\]
ML and MAP detection in case of binary symmetric channel, BSC

One of the most elementary system models for bit flips or bit errors, from the purely binary signals perspective, is the so called binary symmetric channel (BSC) illustrated below for which $\Omega_A = \{0,1\}$.

Good example where we use such model is in the context of error correction codes in the latter part of the course.

Now, the ML detector compares the likelihood values:

$$
\begin{align*}
\text{If } y = 0: & \quad p_{Y|A}(0|0) = 1 - p, \quad p_{Y|A}(0|1) = p \\
\text{If } y = 1: & \quad p_{Y|A}(1|0) = p, \quad p_{Y|A}(1|1) = 1 - p
\end{align*}
$$

Assuming that $p < 0.5$ (and thus $1 - p > p$), the ML detector in BSC is trivial:

$$\hat{a}_{ML} = y$$

ML and MAP detection in BSC, cont’d

MAP detector, on the other hand, compares the weighted likelihood values:

$$
\begin{align*}
\text{If } y = 0: & \quad p_{Y|A}(0|0) p_A(0) = (1 - p)q, \quad p_{Y|A}(0|1) p_A(1) = p(1 - q) \\
\text{If } y = 1: & \quad p_{Y|A}(1|0) p_A(0) = pq, \quad p_{Y|A}(1|1) p_A(1) = (1 - p)(1 - q)
\end{align*}
$$

Thus, the operation of the MAP detector can be written as follows, depending on the prior probabilities ($q$) and the bit flip probability $p$:

- Observation $y = 0$:

  $$\hat{a}_{MAP} = \begin{cases} 
  0 & \text{if } q > p \\
  1 & \text{if } q < p
  \end{cases}$$

- Observation $y = 1$:

  $$\hat{a}_{MAP} = \begin{cases} 
  1 & \text{if } q < 1 - p \\
  0 & \text{if } q > 1 - p
  \end{cases}$$

Thus, if $p < q < 1 - p$, the MAP detector is identical to the ML detector (i.e., $\hat{a}_{MAP} = y$), while in other cases, they are not identical.
ML AND MAP DETECTORS FOR VECTOR VARIABLES

Next, we generalize from single scalar variables to vector variables. This has many applications, e.g., detection of bit or symbol sequences (modeled as vectors) instead of single bit or symbol.

Or even more elementary, single complex symbol can be considered as a two-dimensional vector (Re and Im dimensions, complex plane).

Suppose we stack now \( L \) symbols into a corresponding symbol or signal vector as

\[
S = [A_1, A_2, \ldots, A_L]^T
\]

Similarly, we treat the consecutive observations in the receiver as an observation vector \( Y \).

Like before, we describe the statistical dependence of input and observation using the conditional distribution

\[
f_{Y|S}(y | s)
\]

If the different components/elements can be assumed statistically independent, then it follows that (one of the very basic laws of probabilities; joint probability of independent events becomes product of the element probabilities)

\[
f_{Y|S}(y | s) = \prod_{l=1}^{L} f_{Y_l|S_l}(y_l | s_l)
\]

In general, given that the elements \( A_l \) of the vector \( S = [A_1, A_2, \ldots, A_L]^T \) belong to a finite alphabet of size \( M \) (denoted with, say, \( \Omega_A \)), then there are \( K = M^L \) different vectors among which to decide

- These form a so-called vector alphabet \( \Omega_S \).

Then, conceptually, the detectors are defined similar to earlier developments, namely:

\[
\hat{s}_{ML} = \arg\max_{s \in \Omega_S} f_{Y|S}(y | s)
\]

\[
\hat{s}_{MAP} = \arg\max_{s \in \Omega_S} p_{S|Y}(s | y)
= \arg\max_{s \in \Omega_S} f_{Y|S}(y | s)p_S(s)
\]

And like earlier, the difference between ML and MAP is the role of the prior statistics (here \( p_S(s) \)).

Furthermore, if all vectors \( s \in \Omega_S \) are equally likely (uniform priors), the detectors yield identical results (like in scalar case).

Next, we proceed again with some concrete example cases where, e.g., additive white Gaussian noise links the symbols and observations

- To be able to do that, we first shortly refresh our memory with the basics of random vectors and their distributions
- Why? Because the detectors build on these, see above expressions
Side-step: Gaussian random vectors

Random vector $\mathbf{X} = [X_1, X_2, ..., X_L]^T$ is a vector whose components $X_1, X_2, ..., X_L$ are random variables.

Like any random quantity, random vectors are characterized through the probability distribution, say $f_X(x)$
- Contains the statistical characterization of individual components
- Describes also the statistical dependence of different elements

Gaussian random vector has, in general, the following probability distribution (cf. scalar case)

$$f_X(x) = \frac{1}{\sqrt{(2\pi)^L |K|}} \exp\left(-\frac{1}{2}(x - \mathbf{m})^T K^{-1}(x - \mathbf{m})\right)$$

where $\mathbf{m} = E(\mathbf{X})$ and $K = E(\mathbf{X} - \mathbf{m})(\mathbf{X} - \mathbf{m})^T$ denote the mean vector and covariance matrix.

- In the special case of uncorrelated and equal-variance ($\sigma^2$) components, the covariance matrix reads $K = \text{diag}(\sigma^2, \sigma^2, ..., \sigma^2) = \sigma^2 \mathbf{I}$.
- Then the determinant $|K| = (\sigma^2)^L$ and we can write (here $\|\cdot\|$ denotes L2 vector norm)

$$f_X(x) = \frac{1}{\sqrt{(2\pi)^L |K|}} \exp\left(-\frac{1}{2}(x - \mathbf{m})^T K^{-1}(x - \mathbf{m})\right)$$

$$= \frac{1}{\sqrt{(2\pi\sigma^2)^L}} \exp\left(-\frac{1}{2\sigma^2}(x - \mathbf{m})^T(x - \mathbf{m})\right)$$

$$= \frac{1}{\sqrt{(2\pi\sigma^2)^L}} \exp\left(-\frac{1}{2\sigma^2} \|x - \mathbf{m}\|^2\right)$$

Example 1: two-dimensional Gaussian random vector, uncorrelated components. Below distribution and example realizations:
**Example 2**: two-dimensional Gaussian random vector, strongly correlated components. Below distribution and example realizations:

---

**ML AND MAP VECTOR DETECTION IN AWGN CHANNEL**

Now, finally, we are ready to address the vector detection in the following elementary scenario

\[ Y = S + N \]

where vector \( S \in \Omega_S \) and noise vector \( N \) has Gaussian distributed components with variance \( \sigma^2 \) and zero mean.

Furthermore, we assume the noise components to be uncorrelated.

Then (check previous slides), the distribution of the noise vector is

\[
f_N(n) = \frac{1}{\sqrt{(2\pi\sigma^2)^L}} \exp(-\frac{1}{2\sigma^2} ||n||^2)
\]

and thus the likelihood function reads (why?)

\[
f_{Y|S}(y \mid s) = \frac{1}{\sqrt{(2\pi\sigma^2)^L}} \exp(-\frac{1}{2\sigma^2} ||y - s||^2)
\]

Now, the ML detector chooses such \( s \in \Omega_S \), which maximizes the above likelihood function.

It’s easy to see that this is identical to minimizing the Euclidian distance \( ||y - s|| \), or squared Euclidian distance \( ||y - s||^2 \).

Hence, in AWGN case, the ML vector detector corresponds to minimizing the squared Euclidian distance of the observation and the decision candidates, i.e.:

\[
\hat{s}_{ML} = \arg\max_{s \in \Omega_S} f_{Y|S}(y \mid s)
\]

\[
= \arg\min_{s \in \Omega_S} ||y - s||^2
\]
ML and MAP vector detection in AWGN (cont'd)

In AWGN case, the MAP vector detector corresponds to:

\[
\hat{Y} = \arg\max_{\hat{s} \in \mathbb{C}^L} \log p(Y|\hat{s})
\]

Stemming from above, the corresponding MAP detector reads

With complex-valued variables, \(K_n = E(N_n^H)\) and

As a special case, if the noise components are uncorrelated, then \(K_n = E(N_n^H)\) and we obviously end up in the earlier results.

Excursion: ML and MAP vector detection in case of correlated Gaussian noise components (complementary knowledge)

But instead of uncorrelated noise components (like on previous slides), we assume correlated noise here for generality.

Then, the likelihood function reads (see few slides back)

Then, the ML detector reads

Then, the MAP detector reads

Assuming again additive Gaussian noise model

\[
\hat{Y} = \arg\max_{\hat{s} \in \mathbb{C}^L} \log p(Y|\hat{s})
\]

NB #1: If the elements of the vector \(s\) (here the consecutive symbols) are also independent, the minimization of the squared Euclidian distance can be done element-wise.

NB #2: Considering the detection of a single complex-valued symbol interpreted as a 2-dimensional real vector, the detection can be done separately for real and imaginary parts of the symbol alphabet and independent.

NB #3: Considering a complex-valued vector, the minimization of the squared Euclidian distance still holds but simply deploy complex

\[
||\hat{s} - s||^2 = |s_1 - \hat{s}_1|^2 + |s_2 - \hat{s}_2|^2 + \ldots + |s_L - \hat{s}_L|^2
\]

dealing with complex arithmetic always

\[
f_s(y) = \frac{1}{(2\pi)^L \sqrt{\det(K_s)}} \exp\left(-\frac{1}{2} (y - s)^H K_s^{-1} (y - s)\right)
\]

NB #4: "Symbol-by-symbol" ML detection of sequences can be done element-wise.

e.g. square QAM alphabets, does not hold for PSK in general.

above expression with complex arithmetic works always.
ML VECTOR DETECTION IN BINARY SYMMETRIC CHANNEL

In case of BSC, the individual elements of the signal vector, $S_i$, and observation vector, $Y_i$, are binary:

Due to the BSC, the conditional probabilities (likelihoods) are now:

$$p_{Y|S}(y_i|s_i) = \begin{cases} p & ; y_i \neq s_i \\ 1-p & ; y_i = s_i \end{cases}$$

Definition: Hamming distance $d_H(s, y)$ measures the amount of bits at which the two binary vectors $s$ and $y$ differ.

Assuming now that the above BSC model applies independently to the different components, the conditional probabilities for the bit vectors $s$ and $y$ can now be expressed as (why?)

$$p_{Y|S}(y | s) = p^{d_H(s, y)}(1-p)^{L-d_H(s, y)}$$

Assuming further that $p < 0.5$, the ML detector reads

$$\hat{s}_{ML} = \arg\max_{s \in \Omega_S} p_{Y|S}(y|s)$$

$$= \arg\min_{s \in \Omega_S} d_H(s, y)$$

Thus, in case of BSC, ML detector chooses that particular signal vector that is closest to the observation in the Hamming distance sense. This result will be deployed, e.g., in error correction later in the course.

ML vector detection in AWGN vs. BSC channels

In both of these two classical examples (AWGN and BSC), the ML detector boils down to distance minimization.

The only difference is the measure of distance:

- **AWGN**: utilizes Euclidian distance
- **BSC**: utilizes Hamming distance
ML VECTOR DETECTOR AND ERROR PROBABILITY

Next we address shortly the detection error probability of the ML vector detector, specifically in the AWGN and BSC channel cases.

In case of AWGN channel, this is obviously very closely related to our earlier error probability analysis of different symbol constellations under AWGN channel

- now we understand that the earlier assumed minimum distance detector is actually the ML detector
- thus, earlier we actually analyzed the ML detector error probability when it comes to the detection of a single real-valued (1-dimensional space) or complex-valued (2-dimensional space) symbol

Here, we generalize those results to cover an arbitrary size vector space, which we can then use later to understand and evaluate, e.g., the error probability characteristics of different sequence detectors and error control decoders.

Error probability of ML detector with only two signal vectors: AWGN case

Suppose we consider a very simple example case where we have only two different signal vectors, \( s_i \) and \( s_j \), both of size \( L \).

Then, it can easily be shown that for a noisy observation of the form

\[
Y = S + N
\]

where \( S \in \Omega_S = \{s_i, s_j\} \) and the components of the noise vector \( N \) are zero-mean uncorrelated Gaussians, each with variance \( \sigma^2 \), the earlier derived detection error probability result of the form

\[
P[\text{detection error}] = Q\left(\frac{d}{2\sigma}\right)
\]

still holds where \( d = \|s_i - s_j\| \).

This is a generalization of the result on page 143, and is a useful result for us in the continuation.
Error probability of ML detector with only two signal vectors: BSC case

Next, we consider the ML detection in a BSC channel with two \( L \) dimensional bit vectors \( s_i \) and \( s_j \), that differ in \( d_H \) bits (i.e., the Hamming distance is \( d_H \)).

In the BSC case, for a given observed bit vector \( y \), the conditional probabilities read

\[
p_{Y\mid s}(y \mid s) = p^{d_H(s,y)}(1 - p)^{L - d_H(s,y)}
\]

Thus, when the true bit word is \( s_i \), the ML detector chooses \( s_j \) (i.e., makes an error) if

\[
d_H(s_j, y) \leq d_H(s_i, y)
\]

It is also clear that bit flips matter only in those bit positions where the two words \( s_i \) and \( s_j \) have different bits (why?).

Thus, a detection error takes place if there are more than \( t \) bit errors or bit flips (due to the BSC) in those bit positions where \( s_i \) and \( s_j \) are different where

\[
t = \begin{cases} 
\frac{(d_H - 1)}{2} & \text{if } d_H \text{ is odd} \\
\frac{d_H}{2} - 1 & \text{if } d_H \text{ is even}
\end{cases}
\]

The corresponding detection error probability can be expressed using a binomial distribution as

\[
Q_{\text{BSC}}(d_H, p) = \sum_{i=t+1}^{d_H} \binom{d_H}{i} p^i (1 - p)^{d_H - i}
\]

where \( Q_{\text{BSC}}(\ldots) \) represents the tail probability of a binomial distribution (see example next page)

BSC vector detection, error probability example

Suppose \( s_i = [000000] \) and \( s_j = [110111] \) for which the Hamming distance is clearly \( d_H = 5 \) bits.

Now, the ML detector building on the Hamming distance comparison makes a detection error if 3 or more bit flips take place within those 5 differing bit positions.

The detection error probability is thus

\[
Q_{\text{BSC}}(5, p) = \binom{5}{3} p^3 (1 - p)^2 + \binom{5}{4} p^4 (1 - p)^1 + \binom{5}{5} p^5
\]

\[
= 10 p^3 (1 - p)^2 + 5 p^4 (1 - p)^1 + p^5
\]

We will later use these results to, e.g., quantify the error correction capabilities of certain error control codes and associated decoding mechanisms.
Error probability of ML detector, more general case

Let assume now that there are more than two different signal vectors, say $\{s_1, \ldots, s_K\}$.

Assume further that $s_i$ is the true signal vector while the observation vector is $y$.

Then, let $E_j$ represent the error scenario where the observation $y$ is closer to $s_j$ than $s_i$.

In an example case with $K=3$ signal vectors, the corresponding error probability is

$$P[\text{error} | s_i \text{ transmitted}] = P[E_2] + P[E_3] - P[E_2 \cap E_3]$$

By adopting the Union Bound approach (see pages 154-155), i.e., neglecting the joint probability $P[E_2 \cap E_3]$, we obtain

$$P[\text{error} | s_i \text{ transmitted}] \leq P[E_2] + P[E_3]$$

In a more general case, the corresponding error probability upper bound reads:

$$P[\text{error} | s_i \text{ transmitted}] \leq \sum_{j=1, j \neq i}^K P[E_j]$$

On the other hand, the most probable individual pairwise error scenario, $E_{\text{max}}$, gives always a lower bound:

$$P[\text{error} | s_i \text{ transmitted}] \geq P[E_{\text{max}}]$$

In most cases, the most probable pairwise error probability dominates (see the 8PSK example on page 155), and thus the lower bound is a good approximation.

Error probability of ML detector, more general case (cont’d)

Based on the previous, the following simple approximations are commonly adopted:

$$P[\text{error} | s_i \text{ transmitted}] \approx P[E_{\text{max}}]$$

**AWGN:**

$$P[E_{\text{max}}] = Q(d_{E,\text{min}} / 2\sigma)$$

**BSC:**

$$P[E_{\text{max}}] = Q_{\text{bsc}}(d_{H,\text{min}}, p)$$

Thus, the error probabilities depend essentially on the minimum distance $d_{\text{min}}$:

- minimum Euclidian distance $d_{E,\text{min}}$ in AWGN case
- minimum Hamming distance $d_{H,\text{min}}$ in BSC case

While the above expressions apply for the detection error probability of an individual signal vector, in most cases of practical interest the overall space of the signal vectors is symmetric such that the above results can be used to quantify the average error probability as well.

In summary:

- We obtained lower and upper bounds, and good approximations, for the ML vector detector error probabilities in both AWGN and BSC type of channels, building on the Gaussian tail probability function $Q(\cdot)$ or the binomial tail probability function $Q_{\text{bsc}}(\cdot, \cdot)$.
- The error probability behavior is dictated by the minimum distance within the set of signal vectors, either minimum Euclidian distance (AWGN case) or minimum Hamming distance (BSC case).
- Naturally, also the noise variance (AWGN case) and the bit flip probability (BSC case) impact the detection error probabilities.
10. WAVEFORM DETECTION AND OPTIMAL RECEIVERS

Next we address the task of building and adopting optimal receivers as a whole, to demodulate and detect the transmitted bits from the received waveform.

Received waveform is
- always subject to noise, commonly assumed additive white Gaussian noise (AWGN)
- also commonly distorted by multipath propagation (linear distortion)

We first address the concept of optimal receivers at general level, independently of the chosen modulation method (I/Q modulated PAM/PSK/QAM, FSK, etc.)
- This builds on the concepts of signal space, signal sub-space, and sufficient statistics

The general structure of the optimal receivers, at large, that we will establish is always composed of
- pre-processing of the analog incoming continuous waveform into a set of discrete decision variables
- actual detection processing, using the ML or MAP principles

Waveform detection and optimal receivers, cont’d

Then, we apply the developed optimal receiver framework to concrete examples, e.g.
- single symbol detection in baseband PAM, I/Q modulated PAM/PSK/QAM, and FSK systems
- symbol sequence detection in baseband PAM and I/Q modulated PAM/PSK/QAM systems

Interestingly, we will find out that e.g. in the case of I/Q modulated PAM/PSK/QAM systems, the optimal receiver will consist of
- I/Q downconversion from RF to baseband
- receiver filtering with a specific filter, called matched filter
- symbol rate sampling
- final detection processing, which in the general case is composed of ML vector/sequence detection implemented in practice in the form of Viterbi algorithm

Thus, the earlier described intuitive receiver processing composed of I/Q downconversion, receiver filtering (filter $f(t)$ below) and symbol rate sampling is actually the optimal way to pre-process the incoming waveform assuming, strictly-speaking, that the receiver filter is a special filter
- the so-called matched filter
Concepts of signal space and signal sub-space

Definition: the overall signal space is the set of all possible finite energy waveforms (there are infinite amount of such waveforms).

Then, assume that in our communication system, the received noiseless waveform can be only one of the following $M$ waveforms:

$$s_m(t), \quad 1 \leq m \leq M, \quad E_m = \int_{-\infty}^{\infty} |s_m(t)|^2 dt < \infty$$

These $M$ communication waveforms span a subspace, named $M_s$, in the overall signal space. We denote the dimensionality of this subspace by $K \leq M$, and assume that the subspace can be represented using the following orthonormal basis functions

$$\phi_1(t), \phi_2(t), \ldots, \phi_K(t)$$

The ability to span the sub-space means that all the communication waveforms can be represented as weighted linear combinations of the basis functions:

$$s_m(t) = \sum_{k=1}^{K} s_{m,k}\phi_k(t)$$

The orthonormality of the basis functions, in turn, means that they are mutually orthogonal/uncorrelated and they have been normalized to unit energy:

$$\int_{-\infty}^{\infty} \phi_i(t)\phi_j^*(t)dt = \begin{cases} 0 & \text{when } i \neq j \\ 1 & \text{when } i = j \end{cases}$$

Signal space and signal sub-space, cont’d

The most important feature of the basis functions is, indeed, that they span the given sub-space.

The ability to adopt orthonormal basis functions, help in finding the exact weights for a given waveform:

$$\int_{-\infty}^{\infty} s_m(t)\phi_i^*(t)dt = \int_{-\infty}^{\infty} \sum_{k=1}^{K} s_{m,k}\phi_k(t)\phi_i^*(t)dt$$

$$= \sum_{k=1}^{K} s_{m,k}\int_{-\infty}^{\infty} \phi_k(t)\phi_i^*(t)dt$$

$$= \ldots \text{ orthonormal basis functions}$$

$$= s_{m,i}$$

That is, the coefficient $s_{m,i}$ is obtained by correlating the waveform $s_m(t)$ against the $i$-th basis function $\phi_i(t)$.

On the other hand, how to find the set of basis functions for the given set of signals?

- here the property to span the sub-space is important, and you can usually easily deduce the result with simple reasoning
- simple examples will follow next

In general, for notational convenience, we represent the different communication signals of the given consider system by using the set of the basis function coefficients $s_{m,1}, s_{m,2}, \ldots, s_{m,K}$ in vector form as

$$S_m = [s_{m,1}, s_{m,2}, \ldots, s_{m,K}]^T, \quad m = 1, 2, \ldots, M$$

- these are called signal vectors
- examples will follow
Signal sub-space and basis functions, examples

**Binary FSK, single symbol**

With single binary symbol, there are only two different signals, illustrated above.

Assuming that the pulse frequencies are chosen such that the pulses themselves are orthogonal (see earlier slides about FSK basics), the pulses directly serve as orthogonal basis functions.

The energy of the above pulses is \( T / 2 \) (verify it), thus by normalizing them by \( \sqrt{2/T} \) we can get unit-energy basis functions as:

\[
\phi_1(t) = \sqrt{\frac{2}{T}} s_1(t) \quad \text{and} \quad \phi_2(t) = \sqrt{\frac{2}{T}} s_2(t)
\]

Thus, the dimensionality of the signal sub-space is \( K = 2 \), and the corresponding two-dimensional signal vectors read

\[
S_1 = [\sqrt{T}/2, 0]^T \quad \text{and} \quad S_2 = [0, \sqrt{T}/2]^T
\]

This is because

\[
s_1(t) = \sqrt{\frac{T}{2}} \cdot \phi_1(t) + 0 \cdot \phi_2(t) \quad \text{and} \quad s_2(t) = 0 \cdot \phi_1(t) + \sqrt{\frac{T}{2}} \cdot \phi_2(t)
\]

Signal sub-space and basis functions, examples, cont’d

**Baseband 4-PAM, single symbol**

Suppose the alphabet is \{-3, -1, +1, +3\} and the pulse-shape is a rectangular pulse (only for illustration purposes).

Different signals corresponding to the different alphabet symbols are

\[
s_1(t) = -3g(t), \quad s_2(t) = -1g(t), \quad s_3(t) = +1g(t), \quad s_4(t) = +3g(t)
\]

Obviously, the pulse \( g(t) \) itself spans the signal sub-space, thus there is only one basis function which is the pulse itself normalized to unit energy:

\[
\phi_1(t) = \sqrt{\frac{1}{T}} g(t)
\]

The dimensionality of the sub-space is \( K = 1 \), and the one-dimensional signal vectors (scalars) read

\[
S_1 = -3\sqrt{T}, \quad S_2 = -1\sqrt{T}, \quad S_3 = +1\sqrt{T}, \quad S_4 = +3\sqrt{T}
\]
OPTIMAL RECEIVER PRE-PROCESSING, GENERAL FORM

Next we assume that the received noisy waveform is of the form

\[ Y(t) = s_m(t) + N(t) \]

where \( N(t) \) denotes additive white Gaussian noise (AWGN) with spectral density \( N_0 \).

Because of noise, this signal is not part of the signal subspace.

The optimal receiver projects this received noisy signal back to the signal sub-space \( M_s \), by correlating it against all \( K \) basis functions:

\[ Y_k = \int_{-\infty}^{\infty} Y(t) \phi_k^*(t) \, dt = s_{m,k} + N_k \]

where \( N_k = \int_{-\infty}^{\infty} N(t) \phi_k^*(t) \, dt \) denotes the noise projection. By collecting the different correlator outputs into a vector \( \bar{Y} \), we obtain

\[ \bar{Y} = s_m + N \]

Concept of sufficient statistics

The set of the correlator outputs \( Y_k, 1 \leq k \leq K \), or the vector, \( \bar{Y} \) forms the so called sufficient statistics for the detection.

This means that the actual detection process can utilize only these \( K \) correlations \( Y_k, 1 \leq k \leq K \), instead of the full continuous time waveform, without losing any information in the pre-processing

- everything that is not contained in \( Y_k, 1 \leq k \leq K \), is statistically independent of \( S_m \) and \( N \) (proof available e.g. in the course book)
- that is, we efficiently discretize the received continuous waveform into a set of \( K \) decision variables

Thus, the actual detection can rely on \( \bar{Y} \), instead of \( Y(t) \), to make reliable decisions regarding which of the possible \( M \) signals is entering the receiver, without any information being lost from the detection perspective.
Noise at correlator outputs

Suppose we first consider the following noise projections or correlations of the form

$$
N_f = \int_{-\infty}^{\infty} N(t)f^*(t)dt
$$

$$
N_g = \int_{-\infty}^{\infty} N(t)g^*(t)dt
$$

where $N(t)$ is white Gaussian noise with spectral density $N_0$.

Then, it is easy to show (try it, you need the AWGN assumption) that

$$
E \left[ N_f N_g^* \right] = N_0 \int_{-\infty}^{\infty} g(t)f^*(t)dt
$$

$$
E \left[ |N_f|^2 \right] = N_0 \int_{-\infty}^{\infty} |f(t)|^2 dt
$$

Based on this, the noise components

$$
N_k = \int_{-\infty}^{\infty} N(t)\phi_k^*(t)dt \quad , 1 \leq k \leq K
$$

contained in the sufficient statistics are all Gaussian distributed, mutually uncorrelated, and have all variance $N_0$.

This is an important result, e.g., in

- being able to define the actual ML detector (will be done next)
- to analyze the error probability of the receiver

OPTIMAL RECEIVER AND ML DETECTION

Based on our previous developments, the correlations $Y_k, 1 \leq k \leq K$ compose the following noisy vector model:

$$
\begin{bmatrix}
Y_1 \\
\vdots \\
Y_K
\end{bmatrix}
= \begin{bmatrix}
s_{m,1} \\
\vdots \\
s_{m,K}
\end{bmatrix} + \begin{bmatrix}
N_1 \\
\vdots \\
N_K
\end{bmatrix} \Leftrightarrow \mathbf{Y} = \mathbf{S}_m + \mathbf{N}
$$

Based on the previous page, the elements of the noise vector $\mathbf{N}$ are uncorrelated equal-variance Gaussians, and thus we can define the ML vector detector directly as:

$$
\hat{s}_{ML} = \arg\min_{\mathbf{s} \in \Omega_k} \| \mathbf{Y} - \mathbf{s} \|^2 , \quad \Omega_S = \{ \mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_M \}
$$

Thus the overall receiver from the incoming noisy waveform to the ML decision appears as sketched below:

$$
d_m = \| \mathbf{Y} - \mathbf{S}_m \|^2 = \sum_{k=1}^{K} |Y_k - s_{m,k}|^2 , \quad m = 1, 2, \ldots, M
$$
Optimal receiver and ML detection, cont’d

Thus the overall optimal ML receiver works as follows:

1) Calculate correlations

\[ Y_k = \int_{-\infty}^{\infty} Y(t) \phi_k^*(t) dt, \quad 1 \leq k \leq K \]

2) Calculate distances

\[ d_m = \| \mathbf{y} - \mathbf{s}_m \|^2 = \sum_{k=1}^{K} |Y_k - s_{m,k}|^2 \]

3) Choose \( \mathbf{s}_m \) which minimizes the distance

Alternatively, the distance \( d_m \) can also be expressed as

\[ d_m = \sum_{k=1}^{K} |Y_k|^2 - 2 \text{Re} \left[ \sum_{k=1}^{K} Y_k s_{m,k}^* \right] + \sum_{k=1}^{K} |s_{m,k}|^2 \]

\[ = \sum_{k=1}^{K} |Y_k|^2 - 2K_m \]

where

\[ K_m = \text{Re} \left[ \sum_{k=1}^{K} Y_k s_{m,k}^* \right] - \frac{1}{2} \sum_{k=1}^{K} |s_{m,k}|^2 = \text{Re} \left[ \sum_{k=1}^{K} Y_k s_{m,k}^* \right] - E_m / 2 \]

\[ = \text{Re} \{ \langle \mathbf{y}, \mathbf{s}_m \rangle \} - E_m / 2 \]

\[ E_m = \sum_{k=1}^{K} |s_{m,k}|^2 = \| \mathbf{s}_m \|^2 \]

Since the term \( \sum_{k=1}^{K} |Y_k|^2 \) does not impact the decision, we can alternatively use the metric \( K_m \) to make the decision

- choose \( \mathbf{s}_m \) which maximizes \( K_m \)

---

Based on the previous, we can sketch the following alternative implementation of the optimal receiver as:

\[ K_m = \text{Re} \left[ \sum_{k=1}^{K} Y_k s_{m,k}^* \right] - E_m / 2 \]

Here, you essentially calculate two different correlation measures

- First correlating the continuous-time received signal against the basis functions to get \( Y_k, 1 \leq k \leq K \)
- Then correlating \( Y_k, 1 \leq k \leq K \) against different signal vector coefficients \( s_{m,k} \), for each \( 1 \leq m \leq M \), to get \( \text{Re} \left[ \sum_{k=1}^{K} Y_k s_{m,k}^* \right] \)

Finally, notice that as yet another alternative implementation of the same receiver, the decision variables \( K_m \) can be calculated as (show it):

\[ K_m = \text{Re} \left[ \int_{-\infty}^{\infty} Y(t) s_{m,k}^* dt \right] - \frac{1}{2} \int_{-\infty}^{\infty} |s_{m,k}|^2 dt \]

- correlating directly against the possible waveforms \( s_{m}(t) \)
Baseband PAM: Optimum receiver for a single symbol

For a simplistic example case of communicating a single symbol \( A \in \Omega_A \), the incoming received signal reads

\[
Y(t) = Ah(t) + N(t)
\]

where \( h(t) = g(t) * b(t) \) refers to the pulse-shape at receiver input (recap: \( g(t) \) is the TX pulse-shape, \( b(t) \) is the channel response).

Denoting the energy of the received pulse-shape by \( E_h \), we can write

\[
Y(t) = Ah(t) + N(t) = A\sqrt{E_h}\phi_1(t) + N(t), \quad S = [\sqrt{E_h}A]
\]

where the one and only basis function \( \phi_1(t) = \sqrt{E_h^{-1}}h(t) \). The optimum receiver is thus (in baseband PAM, all variables are assumed real-valued)

\[
\sqrt{E_h}h(t)
\]

\[
Y(t) \rightarrow \int Y_i \rightarrow \text{SLICER} \rightarrow \hat{A}
\]

In an alternative implementation, the needed correlation can be calculated using a matched filter \( h(-t) \) followed by a sampler as:

\[
\begin{align*}
Y(t) & \rightarrow \sqrt{E_h}h(-t) \\
\int & \int \rightarrow \text{SAMPLER} \\
\rightarrow & Y_i \\
& \text{SLICER} \\
& \hat{A}
\end{align*}
\]

This is exactly the same receiver structure as earlier (RX filter + sampler + slicer), the only difference is that the RX is filter is a special filter – the matched filter \( h(-t) \) !

I/Q modulated PAM/PSK/QAM: Optimum receiver for a single symbol

In the I/Q modulated case, when again communicating only a single symbol \( A \in \Omega_A \), the received signal reads

\[
Y(t) = \sqrt{2} \text{Re}[Ah(t)e^{j\omega_0 t}] + N(t)
\]

where \( h(t) = g(t) * b_{LP}(t) \) is the received pulse-shape with \( b_{LP}(t) \) denoting the lowpass equivalent channel response.

Denoting the energy of the received pulse \( h(t) \) by \( E_h \), we can write

\[
Y(t) = \sqrt{2} \text{Re}[A]\text{Re}[h(t)e^{j\omega_0 t}] - \sqrt{2} \text{Im}[A]\text{Im}[h(t)e^{j\omega_0 t}] + N(t)
\]

\[
= \sqrt{E_h} \text{Re}[A]\phi_1(t) + \sqrt{E_h} \text{Im}[A]\phi_2(t) + N(t)
\]

where the two needed real-valued basis functions read

\[
\phi_1(t) = \sqrt{2E_h^{-1}}\text{Re}[h(t)e^{j\omega_0 t}], \quad \phi_2(t) = -\sqrt{2E_h^{-1}}\text{Im}[h(t)e^{j\omega_0 t}]
\]

- show yourself that the above two functions really are orthonormal
- thus the two-dimensional signal vectors are of the form
  \[
  S = [\sqrt{E_h}\text{Re}[A], \sqrt{E_h}\text{Im}[A]]^T
  \]
- the optimum receiver building on this appears below

\[
\begin{align*}
Y(t) & \rightarrow \int \int \rightarrow \text{CORRELATOR} \\
\rightarrow & Y_i \\
\rightarrow & \text{SLICER} \\
& \hat{A}
\end{align*}
\]

\[
\begin{align*}
Y(t) & \rightarrow \int \int \rightarrow \text{CORRELATOR} \\
\rightarrow & Y_i \\
& \text{SLICER} \\
& \hat{A}
\end{align*}
\]

\[
\begin{align*}
\sqrt{2E_h} \text{Re}[h(t)e^{j\omega_0 t}] \\
\rightarrow \int \int \rightarrow \text{SLICER} \\
& \hat{A}
\end{align*}
\]

\[
\begin{align*}
\sqrt{2E_h} \text{Im}[h(t)e^{j\omega_0 t}] \\
\rightarrow \int \int \rightarrow \text{SLICER} \\
& \hat{A}
\end{align*}
\]
I/Q modulated PAM/PSK/QAM: Optimum receiver for a single symbol

The previous two real-valued basis functions can be combined into one complex-valued basis function of the form

$$\phi(t) = \sqrt{2E_h^{-1}} h(t) e^{j\omega_c t}$$

This is intuitively also clear since obviously the complex-modulated pulse $h(t)e^{j\omega_c t}$ spans all possible received noiseless signals.

Calculating the correlation against this complex-valued basis function can be expressed as:

$$\int_{-\infty}^{\infty} Y(t) \phi^*(t) dt = \sqrt{2E_h^{-1}} \int_{-\infty}^{\infty} Y(t) h^*(t) e^{-j\omega_c t} dt$$

$$= \left\{ \sqrt{2E_h^{-1}} (Y(t) e^{-j\omega_c t}) * h^*(-t) \right\} |_{t=0}$$

This leads to the following optimum receiver structure:

```
 Y(t)    e^{-j\omega_c t}    COMPLEX MATCHED FILTER    SAMPLER    COMPLEX SLICER
    \downarrow                        \downarrow                        \leftarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
    \rightarrow                        \rightarrow                        \rightarrow
```

This is, again, exactly the same receiver as already deduced earlier (I/Q downconversion + RX filtering + sampling)

- the only difference is that the RX filter is the matched filter $h^*(-t)$ !

About the matched filter

Thus, like established on the previous page, when the received pulse-shape is $h(t) = g(t) * b_{LP}(t)$, the true matched filter impulse response is $h^*(-t)$.

Directly based on the properties of the Fourier transform, the frequency response of the matched filter is $H^*(f)$ when $H(f)$ denotes the Fourier transform of the received pulse $h(t)$ (... show it).

Therefore, since

$$|H^*(f)| = |H(f)|$$

the matched filter in the receiver will emphasize those frequencies that are already strong in the input pulse while attenuates those that are already weak

- automatically zero response outside the useful signal band (like any good RX filter should do)
- but also filters noise inside the signal band (something that normal RX filters wont do)
- see illustration next page

From this, it follows that the matched filter maximizes the RX filter output SNR from an individual incoming pulse point of view !

Notice also that the matched filter acts automatically as a phase equalizer, since the total transfer function is $H(f)H^*(f) = |H(f)|^2$, i.e., is real-valued (zero phase)

- however, the amplitude response is clearly NOT equalized (just the opposite)
- in other words, matched filter is NOT an equalizer (one common misunderstanding)
About the matched filter, cont’d

Since $h(t) = g(t) \ast b_{LP}(t)$ and therefore
\[ h^*(-t) = g^*(-t) \ast b_{LP}^*(-t) \]

one can view that the true matched filter contains two parts

- a filter matched to the TX pulse $g(t)$, i.e., $g^*(-t)$
- followed by another filter matched to the lowpass equivalent channel $b_{LP}(t)$, i.e., $b_{LP}^*(-t)$

In practical systems, especially if analog pulse-shaping filters are used, one cannot truly realize or adopt the true matched filter in the receiver as by default, the channel response $b_{LP}(t)$ is not known.

- however, if the receiver filtering is implemented using digital filtering, combined with higher than symbol rate sampling and one form of channel estimation, this can be done

- we will talk explicitly about the channel estimation later in the course

Also notice that if a real-valued symmetrical square-root Nyquist filter $g_{\text{R}}(t)$ (say e.g. square-root raised-cosine pulse, discussed on page 98) is used as the TX pulse shaping filter, then

- due to the symmetry and real-valued nature of such filters, it follows that $g_{\text{R}}(t) = g_{\text{R}}^*(-t)$ and thus using $g_{\text{R}}(t)$ also as the RX filter means matched filtering from the TX-RX pulses point of view

- it is, however, strictly speaking sub-optimum solution since the part matched to the channel response is missing

- this anyway formally establishes that it is better to split the intended pulse-shape filtering “half & half” between TX and RX, through the square-root filters, instead of having a full raised-cosine filter at TX and a regular LPF at RX
Next we address constructing an optimum receiver for symbol sequence detection, assuming I/Q modulated PAM/PSK/QAM as a concrete example case.

Basic assumptions and notations in the following are

- We consider transmitting and receiving a sequence of \( L \) symbols using I/Q modulated PAM/PSK/QAM
- Symbol alphabet size is \( M \)
  
  \[ N = M^L \]

The general form for the received signal reads now

\[
Y(t) = \sqrt{2} \Re \left[ e^{j\omega_c t} \sum_{k=0}^{L-1} A_k h(t - kT) \right] + N(t)
\]

where \( h(t) = g(t) \ast b_{LP}(t) \) denotes the received pulse-shape while \( N(t) \) refers to AWGN.

We will first find the basis function representation for the set of all possible noiseless received signals (the signal sub-space), then establish the optimum receiver pre-processing, yielding a finite set of discrete decision variables, and then device the corresponding ML vector detector.

We will find out that at the output of the optimum receiver preprocessing engine, there is generally inter-symbol interference (ISI)

- this must be taken into account in the ML vector detector, in the form of Viterbi algorithm
- or alternatively, removed first by a channel equalizer, followed by symbol-by-symbol detector

Basis functions and optimum receiver preprocessing

Stemming from the structure of the received signal, namely

\[
Y(t) = \sqrt{2} \Re \left[ e^{j\omega_c t} \sum_{k=0}^{L-1} A_k h(t - kT) \right] + N(t)
\]

it is clear that the basis functions of the form

\[
\phi_0(t) \simeq h(t) e^{j\omega_c t}, \quad \phi_1(t) \simeq h(t - T) e^{j\omega_c t}, \quad \ldots, \quad \phi_{L-1}(t) \simeq h(t - (L-1)T) e^{j\omega_c t}
\]

span the signal sub-space (why?).

The dimensionality of the sub-space, \( K \), is thus identical to the sequence length \( L \), i.e., \( K = L \).

Then, the correlation or projection against the basis function \( \phi_k(t) \) can be expressed as (verify it)

\[
R_k = \int_{-\infty}^{\infty} Y(t) \phi_k^*(t) dt = \int_{-\infty}^{\infty} Y(t) h^*(t - kT) e^{-j\omega_c t} dt
\]

\[
= \left\{ Y(t) e^{-j\omega_c t} \ast h^*(-t) \right\}_{t = kT}
\]

- These are nothing but symbol rate samples at the output of the I/Q downconversion and matched filtering based processing stages
- Notice that I have deliberately changed the notation such that the correlations against the different basis functions are now called \( R_k \) instead of \( Y_k \) (used earlier)
  
  - this is only to reflect better the time-sequence interpretation, i.e., the samples \( R_k \) are really symbol-rate samples of the I/Q downconverted and filtered signal \( R(t) = Y(t) e^{-j\omega_c t} \ast h^*(-t) \)

This optimum receiver preprocessing is illustrated on next page.
Basis functions and optimum RX preprocessing, cont’d

Illustration of the sampled matched filter based receiver preprocessing:

\[ Y(t) \xrightarrow{e^{-j\omega_0 t}} \text{MATCHED FILTER} \rightarrow \sqrt{2} h^*(-t) \rightarrow \sqrt{2} H(f) \rightarrow R_i \]

Notice the large similarity against the earlier deduced receiver structures – the only difference is the exact nature of the RX filter, which is here the matched filter!

**Corollary 1:** The orthogonality of the basis functions?

The basis functions \( \phi_0(t) \simeq h(t)e^{j\omega_0 t}, \phi_1(t) \simeq h(t-T)e^{j\omega_0 t}, \ldots \) clearly span the signal subspace but are actually not (necessarily) orthogonal:

\[
\int_{-\infty}^{\infty} \phi_k(t)\phi_k^*(t)dt = \int_{-\infty}^{\infty} h(t-lT)e^{j\omega_0(t-lT)}h^*(t-kT)e^{-j\omega_0(t-kT)}dt
\]

\[
= \int_{-\infty}^{\infty} h(t)h^*(t-(k-l)T)dt
\]

\[
= h(t) \ast h^*(-t)
\]

\[
\neq 0 \quad \text{for} \quad k \neq l
\]

- i.e., the orthogonality depends on the more detailed nature of the received pulse-shape \( h(t) = g(t) \ast b_{LP}(t) \)
- In general, for an arbitrary received pulse-shape \( h(t) = g(t) \ast b_{LP}(t) \), the basis functions are not orthogonal.

**Corollary 2:** Correlation of noise samples?

From Corollary 1, it automatically follows that (see also slide 287)

\[
E\left[ N_i N_k^* \right] \simeq N_0 \int_{-\infty}^{\infty} \phi_k(t)\phi_k^*(t)dt
\]

\[
= h(t) \ast h^*(-t)
\]

\[
\neq 0 \quad \text{for} \quad k \neq l
\]

Thus, for an arbitrary received pulse-shape \( h(t) = g(t) \ast b_{LP}(t) \), the basis functions are not orthogonal and therefore different noise samples at the output of the sampled matched filter based preprocessing are also correlated.

**Corollary 3:** Is there Inter-symbol Interference (ISI)?

The total effective pulse-shape at the output of the matched filter reads

\[
p(t) = h(t) \ast h^*(-t) = \int_{-\infty}^{\infty} h(\tau)h^*(\tau-t)d\tau
\]

Thus, there is no ISI at the sampled matched filter output only iff

\[
\forall k \neq 0, k \in \mathbb{Z} : \quad p_k = p(kT) = \int_{-\infty}^{\infty} h(t)h^*(t-kT)dt = 0
\]

Again, for an arbitrary received pulse-shape \( h(t) = g(t) \ast b_{LP}(t) \), there is nothing that would enforce the above to hold, and thus in general there is ISI at the output of the sampled matched filter.
Basis functions and optimum RX preprocessing, cont’d

Thus in general, the matched filter DOES NOT eliminate ISI (this is one common misunderstanding)

- only in the special case where the received pulse-shape \( h(t) \) happens to be symmetrical, real-valued square-root Nyquist pulse (e.g. a square-root raised cosine pulse), the overall pulse shape \( h(t) \ast h^*(\cdot-t) = h(t) \ast h(t) \) is a Nyquist pulse and there is no ISI.
- but this is only a special case, in general there is ISI.

Notice also that since

\[
p_k = p(kT) = \int_{-\infty}^{\infty} h(t)h^*(t-kT)dt \approx \int_{-\infty}^{\infty} \phi_k(t)\phi_k^*(t)dt \approx E\left[ N_l N_k^* \right]
\]

the existence of ISI, lack of basis function orthogonality and correlation of noise samples are all directly linked to each other when the true matched filter is used as the RX filter.

- if there is ISI, then also the noise samples are correlated (this is the general case).
- if there is no ISI, the noise samples are uncorrelated.

In summary: For an arbitrary received pulse-shape \( h(t) = g(t) \ast b_{\mu}(t) \) the total effective pulse-shape \( p(t) = h(t) \ast h^*(\cdot-t) \) does not obey the Nyquist criteria for zero ISI. In this case, also the noise samples at the output of the sampled matched filter are correlated.

Optimum ML receiver as a whole, conceptual view

Based on our earlier studies (see slide 242), since the noise components are not uncorrelated but are anyway still Gaussians, the overall ML detection based optimum receiver as a whole looks like this:

- the vector of \( L \) received samples, \( \vec{Y} \), is compared against all possible \( N = M^L \) signal vectors \( S_1, S_2, \ldots, S_N \)
  - these different signal vectors are the different noiseless outputs that the sampled matched filter based receiver would have, corresponding to different possible transmitted symbol sequences incorporating the ISI.
  - because noise components are not uncorrelated the simple Euclidian distance metric of the form \( \| \vec{Y} - S_m \| \) cannot be used
  - instead, the more complicated metric from slide 242 is adopted.

From the ML detection point of view, this means effectively that

- for increasing \( L \), \( N = M^L \) increases terribly fast.
- thus, this receiver cannot really be implemented, explicitly, in reality in this form, but serves as the reference.

This is why we will seek, in the continuation, more computing friendly approaches for the actual detection processing.
Optimum ML receiver as a whole, conceptual view, cont’d

The more computing friendly approaches, addressed in the continuation, will build on

- adding a noise whitening filter to first remove the noise correlation, followed then by Euclidian distance minimization based ML vector/sequence detector implemented in the form of Viterbi algorithm
  - substantially reduced complexity compared to the reference optimum receiver on the previous slide, while still facilitating optimum ML receiver as a whole
  - notice that adding the noise whitening filter will also impact the structure of the signal vectors

- or as an alternative, adopting a channel equalizer structure that first explicitly removes ISI after which symbol-by-symbol detection (normal minimum distance slicing, per symbol) is adopted
  - further substantial reduction in computing complexity, but also somewhat degraded performance compared to ML vector/sequence detector

We will address these aspects in the continuation.
11. DEALING WITH INTERSYMBOL INTERFERENCE (ISI) IN RECEIVERS

Next, we address more computing feasible solutions to do reliable decisions about the transmitted bits or symbols, under ISI, in the receivers, with specific focus on the I/Q modulated PAM/PSK/QAM systems.

There are two alternative scenarios, regarding the receiver preprocessing and particularly the role of the exact receiver filter:

- practical receiver with RX filter \( f(t) \); most commonly a square-root raised cosine filter matched to the corresponding similar TX filter

- optimum receiver with RX filter of the form
  \[
  f(t) = h^*(t) = g^*(t) + b_{LP}^*(t)
  \]

In both cases, in general, there is ISI at the output of the RX filter, and hence finding computing feasible ways to handle ISI is very important.

We first establish some fundamental baseband equivalent discrete-time signal and system models, and then stemming from those describe the following solutions to handle ISI:

- Maximum likelihood sequence detection (MLSD), implemented through the Viterbi algorithm; this is the optimum solution in ML sense
- Different channel equalization methods, particularly linear equalizers and decision feedback equalizers, combined with symbol-by-symbol detection; strictly-speaking sub-optimum but computationally much more simple

Baseband equivalent discrete-time models

Recap (see slides 126-131): For an I/Q modulated PAM/PSK/QAM system with TX filter \( g(t) \), channel response \( b(t) \), and RX filter \( f(t) \), the received symbol rate samples \( R_k \) read

\[
R_k = A_k * p_k + Z_k
\]

where \( Z_k \) denotes I/Q downconverted, filtered and sampled noise while

\[
p_k = p(kT)
\]

denotes the corresponding samples of the total pulse-shape

\[
p(t) = g(t) * b_{LP}(t) * f(t)
\]

where \( b_{LP}(t) \) refers to the baseband equivalent channel impulse response.

Furthermore, the spectral density of the noise sequence \( Z_k \) reads

\[
S_Z(e^{j2\pi ft}) = \frac{2N_0}{T} \sum_{m=-\infty}^{\infty} \left| F\left(f - m \frac{1}{T}\right) \right|^2
\]

The above are general results that apply for arbitrary receiver filter \( f(t) \).

Next we consider the two interesting cases in more details, namely:

- practical receiver with RX filter \( f(t) \) being a square-root raised cosine filter
- optimum receiver with RX filter of the form
  \[
  f(t) = h^*(t) = g^*(t) + b_{LP}^*(t)
  \]
Baseband equivalent discrete-time models, cont’d

Case 1: RX filter $f(t)$ is a square-root raised cosine filter

Since the spectral density of the noise sequence $Z_k$ is in general a sampled version of $2N_0 |F(f)|^2 = 2N_0 F(f) F^*(f)$, it follows that in the case of square-root raised cosine RX filter

$$S_Z(e^{j2\pi ft}) = \frac{2N_0}{T} \sum_{m=-\infty}^{\infty} |F\left(f - m \frac{1}{T}\right)|^2 = 2N_0$$

In other words, the sampled noise spectral density is constant meaning that in this special case, the noise samples are uncorrelated.

This is because $F\left(F^*\right)$ maps in time-domain to $f(t) * f^*(-t)$ (trivial to show, just use the definition of Fourier transform), and on the other hand if $f(t)$ is a square-root raised-cosine, then $f(t) = f^*(-t)$ (due to pulse being symmetric and real-valued).

Therefore, $f(t) * f^*(-t) = f(t) * f(t) = \text{Nyquist pulse}$ and thus when sampled at symbol-rate to reflect the autocorrelation function of the symbol-rate sampled noise sequence, you’ll get

$$\{f(t) * f(t)\}_{t=kT} = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{for all } k \neq 0 \end{cases}$$

which means that different noise samples are, indeed, mutually uncorrelated. q.e.d.

Thus, using a square-root raised cosine filter as the RX filter will yield uncorrelated noise samples when sampling takes place at symbol rate.

In general, there is anyway ISI whose exact characteristics and nature depend on the samples $p_k = p(kT)$ of the total effective pulse-shape $p(t) = g(t) * b_{LP}(t) * f(t)$.

Baseband equivalent discrete-time models, cont’d

Case 2: RX filter $f(t)$ is the true matched filter $h^*(-t)$

Since now $f(t) = h^*(-t)$, we can first write the total effective pulse-shape $p(t)$ as

$$p(t) = h(t) * h^*(-t) = \int_{-\infty}^{\infty} h(\tau) h^*(\tau - t) d\tau$$

and therefore

$$p_k = p(kT) = \int_{-\infty}^{\infty} h(t) h^*(t - kT) dt$$

In general, like we already concluded in the earlier lecture, there is nothing that would force this automatically a Nyquist pulse, and hence there is ISI

- notice that due to conjugate symmetry of $h(t) * h^*(-t)$ if also follows that the ISI profile is conjugate symmetric, i.e., $p_k = p^*_k$

On the other, since in this case $F(f) = H^*(f)$, the sampled noise spectral density reads now

$$S_Z(e^{j2\pi ft}) = \frac{2N_0}{T} \sum_{m=-\infty}^{\infty} |H\left(f - m \frac{1}{T}\right)|^2$$

With similar reasoning as on the previous page, this is symbol rate sampled version of $2N_0 |H(f)|^2 = 2N_0 H(f) H^*(f)$ which in time-domain means effectively $2N_0 \{h(t) * h^*(-t)\}_{t=kT} = 2N_0 p_k$.

But since, in general, $p(t) = h(t) * h^*(-t)$ is not a Nyquist pulse, we can conclude that the noise samples are correlated with an autocorrelation function of $2N_0 p_k$. 
Baseband equivalent discrete-time models, cont'd

Using the z-transform notation of the form

\[ P(z) = \sum_{k=-\infty}^{\infty} p_k z^{-k}, \quad P(e^{j2\pi f T}) = \sum_{k=-\infty}^{\infty} p_k e^{-j2\pi f k T} \]

we can summarize our baseband equivalent system model(s) as follows, where in the case of RX filter being the true matched filter (MF), it also follows that (see previous slide)

\[ S_Z(e^{j2\pi f T}) = 2N_0 P(e^{j2\pi f T}) \]

**Concept of Noise Whitening**

In the optimum RX case, the symbol rate sampled noise spectral density is \( S_Z(e^{j2\pi f T}) = 2N_0 P(e^{j2\pi f T}) \), which means an autocorrelation function \( 2N_0 p_k \).

Since \( p_k \) is conjugate symmetric, i.e., \( p_k = p_k^* \), it automatically implies that the z-domain transfer function \( P(z) = \sum_k p_k z^{-k} \) has conjugate symmetric roots

- i.e., if \( z_0 \) is one of the roots of \( P(z) = \sum_k p_k z^{-k} \), then also \( 1 / z_0^* \) is a root

**Example:** Suppose that the baseband equivalent channel is a two-path channel of the form \( b_{LP}(t) = \delta(t) + b \delta(t - T) \), and the TX pulse-shape \( g(t) \) is a square-root raised cosine pulse. Then it directly follows that

- \( h(t) = g(t) + kg(t - T) \) (trivial, show it)
- \( p(t) = h(t) * h^*(-t) = \ldots \)
- \( p_k = \{h(t) * h^*(-t)\}_{t=kT}^{} = \begin{cases} b^* & \text{for } k = -1 \\ 1 + |b|^2 & \text{for } k = 0 \\ b & \text{for } k = 1 \\ 0 & \text{for all } |k| > 1 \end{cases} \)

Thus \( P(z) = \sum_k p_k z^{-k} = b^* z + 1 + |b|^2 + b z^{-1} \) which we can directly factorize as \( P(z) = (1 + b z^{-1})(1 + b^* z) \) and hence the roots are

- \( -b \) and \( -1 / b^* \)
- illustrated in the figure, assuming \( |b| < 1 \) (principal illustration only)
Concept of Noise Whitening, cont’d

Now, as a generalization of the previous example, it is easy to see that due to the conjugate symmetry $p_k = p_{-k}^*$, the z-domain polynomial $P(z) = \sum_k p_k z^{-k}$ can in general be factorized as

$$P(z) = \sum_k p_k z^{-k} = W(z)W^*(1/z)$$

where the conjugate symmetric roots of $P(z)$ are split equally between $W(z)$ and $W^*(1/z)$.

Then, because of the conjugate symmetric roots, it is easy to show that the frequency responses of both filters $W(z)$ and $W^*(1/z)$ are identical and therefore relative to the square-root of $P(e^{j2\pi fT})$, i.e.,

$$|W(e^{j2\pi fT})| = |W^*(e^{-j2\pi fT})| \simeq \sqrt{P(e^{j2\pi fT})}$$

Example: Continuing the previous example we get directly

$$P(z) = b^*z + 1 + |b|^2 + bz^{-1} = (1 + bz^{-1})(1 + b^*z) = W(z)W^*(1/z)$$

It follows that $W(z) = 1 + bz^{-1}$ and $W^*(1/z) = 1 + b^*z$, and thus

$$|W(e^{j2\pi fT})| = \left|1 + be^{-j2\pi fT}\right| = \sqrt{1 + |b|^2 + 2 \text{Re}[be^{-j2\pi fT}]}$$

$$|W^*(e^{-j2\pi fT})| = \left|1 + b^*e^{j2\pi fT}\right| = \sqrt{1 + |b|^2 + 2 \text{Re}[be^{-j2\pi fT}]}$$

$$P(e^{j2\pi fT}) = b^*e^{j2\pi fT} + 1 + |b|^2 + be^{-j2\pi fT} = 1 + |b|^2 + 2 \text{Re}[be^{-j2\pi fT}] = |W(e^{j2\pi fT})||W^*(e^{-j2\pi fT})|$$

Example: Continuing the previous example, it is easy to see that due to the conjugate symmetry $p_k = p_{-k}^*$, the z-domain polynomial $P(z) = \sum_k p_k z^{-k}$ can in general be factorized as

$$P(z) = \sum_k p_k z^{-k} = W(z)W^*(1/z)$$

where the conjugate symmetric roots of $P(z)$ are split equally between $W(z)$ and $W^*(1/z)$.

Thus, by using an inverse filter of the form

$$\frac{1}{W^*(1/z)}$$

or

$$\frac{1}{W(z)}$$

to process the sequence $R_k$ obtained from the sampled matched filter will whiten the noise component in it.

This is because input spectral density is always shaped by the square of the amplitude response of the filter processing it, and thus

$$2N_0 P(e^{j2\pi fT}) \frac{1}{W(e^{j2\pi fT})^2} = 2N_0 P(e^{j2\pi fT}) \frac{1}{W^*(e^{-j2\pi fT})^2} = 2N_0$$

While whitening the noise, the extra filtering stage will of course also impact the overall effective baseband equivalent system model, or the ISI profile, as

$$P(z) \frac{1}{W^*(1/z)} = W(z)$$

or

$$P(z) \frac{1}{W(z)} = W^*(1/z)$$

Example: Again we continue our previous example and choose the noise whitening filter as

$$\frac{1}{W^*(1/z)} = \frac{1}{1 + b^*z}$$

This will then modify the effective system model or ISI profile as

$$P(z) \frac{1}{W^*(1/z)} = W(z) = 1 + bz^{-1}$$
Concept of Noise Whitening, cont’d

The noise whitening filter \( 1 / W^*(1/z) = 1/(1 + b^*z) \) assumed in this example is a purely anticausal recursive filter, and hence in an actual implementation its impulse response has to be truncated and delayed to be a causal FIR filter.

Notice also that this anticausal recursive filter has a pole at \(-1/b^*\) and is stable only if \(|1/b| > 1\) or \(|b| < 1\).

- recap: stability of recursive filters (check your basic DSP course notes)
  - causal filters: stable iff all poles inside unit circle
  - anticausal filters: stable iff all poles outside unit circle

Hence, if \(|b| > 1\), one can do the original factorization of \( P(z) \) in an alternative form as

\[
P(z) = b^*z + 1 + b^* + bz^{-1} = (b^* + z^{-1})(b + z) = W(z)W^*(1/z)
\]

and choose \( 1 / W^*(1/z) = 1/(b + z) \). This, again interpreted as a purely anticausal recursive filter, has a pole at \(-b\) and is thus stable when \(|b| > 1\). In this case, the remaining system model or ISI profile is

\[
P(z) = \frac{1}{W^*(1/z)} = W(z) = b^* + z^{-1}
\]

To finally summarize (thank god):

- a stable noise whitening filter \( 1 / W^*(1/z) \) can always be found by properly factorizing \( P(z) = W(z)W^*(1/z) \)
- by using an anticausal noise whitening filter, the remaining ISI profile \( P(z) / W^*(1/z) = W(z) \) is always purely causal containing only so called post-cursor ISI such that the direct path is strongest
  - in a physical implementation, anticausal \( 1 / W^*(1/z) \) has to be truncated and delayed causal

The overall system model including noise whitening is shown below where

- \( \tilde{P}(z) = P(z) / W^*(1/z) = W(z) \) denotes the overall effective system model or ISI profile including the noise whitening filter
- the whitened noise is denoted by \( N_k \) while the observed samples are called \( Y_k \)

Based on this, we can assume a white noise model in the further equalizer and MLSD developments, independently of which RX filter (MF or square-root RC) is actually being used

- the special case where the RX filter is a square-root raised-cosine is obtained as a substitution
  - \( \tilde{P}(z) = P(z) \) (or \( 1 / W^*(1/z) = 1 \))
  - \( Y_k = R_k \) and \( N_k = Z_k \)

\[
S_N(e^{j2\pi T}) = 2N_0
\]
Concept of Noise Whitening, cont’d

Additional side-notes:

- in our formulation, we have chosen to use stable anticausal filters as the noise whitening filter
  - has to be truncated and delayed causal in the physical implementation
- this has the benefit that the remaining ISI profile is a minimum phase FIR response
  - see our simple example – depending on whether $|b| < 1$ or $|b| > 1$, the remaining system models are
    $$ P(z) \frac{1}{W^*(1/z)} = 1 + bz^{-1} \quad \text{or} \quad P(z) \frac{1}{W(1/z)} = b^* + z^{-1} $$
- it is also possible to choose the noise whitening filters to be stable causal recursive filters
  - In our simple example, assuming the case of $|b| < 1$, such filter would be
    $$ \frac{1}{W(z)} = \frac{1}{1 + bz^{-1}} $$
    which is a causal stable recursive filter (assuming $|b| < 1$).
  - This means that the remaining system model reads
    $$ P(z) \frac{1}{W(z)} = 1 + b^*z $$
    which in a real causal overall system would be $z^{-1} + b^* = b^* + z^{-1}$ which is a maximum phase FIR since $|b| < 1$, i.e., the post-cursor ISI term is stronger than the direct term
- In the continuation, we will assume the anticausal noise whitening filter approach (unless explicitly said otherwise).

ML SEQUENCE DETECTION AND VITERBI ALGORITHM

Sequence detection refers to applying the ML (or MAP) vector detection principle to the observed sequence as a whole, to make a decision about the transmitted symbol sequence (see slides 235-242 and 274 for basics).

In general, the reason why individual symbols cannot be detected separately, is obviously the ISI that is in general present in the observed sequence

$$ Y_k = A_k * \bar{p}_k + N_k $$

- ISI causes continuous memory (continuous interaction between neighboring symbols) into the received signal, and thus the symbol sequence must be detected as a whole
- this is, indeed, the sequence detection concept

In general, as the transmitted symbol sequences are commonly fairly long, or very long, there are substantial complexity issues involved

- thus instead of brute force search based approaches, more clever solutions are needed
- this is what we will study in this part, primarily in the form of Viterbi algorithm based ML sequence detection
- however, before we do that, we'll first revisit the conceptual maximum likelihood sequence or vector detector
- the computationally efficient Viterbi algorithm will follow after that
Basic received signal model, recap

Like we established earlier, the received signal samples \( Y_k \) relate to the transmit symbols \( A_k \) as

\[
Y_k = A_k \ast p_k + N_k = S_k + N_k
\]

where \( N_k \) denotes the white(red) noise sequence, \( A_k \) the original symbol sequence and the intersymbol interference coefficients are \( p_k \)

- the signal \( S_k = A_k \ast p_k \) is the ISI distorted symbol sequence without noise

In the following, we develop ML vector detection framework based sequence detector, building on this model of the received samples, taking the ISI and computational constraints into account

- conceptually, this is actually easy since the noise sequence \( N_k \) is now white and Gaussian

ML sequence detector (MLSD), conceptual form

Suppose the transmit symbol sequence \( A_k \) is of length \( L \) symbols, \( A_1, A_2, \ldots, A_L \).

Then we can write a following linear model relating the observed samples, stacked into vector, and the transmit symbols, also stacked into vector, as

\[
Y = PA + N = S + N
\]

where the associated vectors are of the form \( Y = [Y_1, Y_2, \ldots, Y_{L+D-1}]^T \), \( A = [A_1, A_2, \ldots, A_L]^T \), \( N = [N_1, N_2, \ldots, N_{L+D-1}]^T \) and \( P \) is a convolution matrix (example soon) composed of the ISI coefficients

- here \( D \) refers to the length of the ISI profile \( p_k \) and it is understood that also the start and end transients are observed (which is a detail)

With a symbol alphabet size \( M \) for individual symbols, the total size for the vector alphabet is \( N = M^L \).

Now, based on previous developments, and assuming that the ISI coefficients (matrix \( P \)) are known, the ML vector detector reads

\[
\hat{a}_{ML} = \arg\max_{a \in \Omega_A} f_{Y|A}(y|a)
= \arg\min_{a \in \Omega_A} \| y - Pa \|^2
\]

where we have utilized the assumption of uncorrelated Gaussian noise components in the noise vector \( N \).

This conceptual form of MLSD based receiver is illustrated, as a whole next page.

What this receiver essentially does is that it tries all possible transmit symbol sequences to the ISI profile \( p_k \) and compares the resulting sequences \( S_k = A_k \ast p_k \) to the observed sequence \( Y_k \) and chooses such sequence \( \hat{A}_k \) that minimizes the squared Euclidian distance.
ML sequence detector (MLSD), conceptual form, cont’d

Notice that the computing complexity is/can easily be substantial or overwhelming:

- \( N = M^L \) decision candidates, a TERRIBLY LARGE VALUE for any larger value of the sequence length \( L \)
  - Complexity increases exponentially for increasing sequence length
- This largely motivates for reduced complexity processing (Viterbi algorithm, which will follow soon)

Example: ISI over \( D = 3 \) consecutive symbols, i.e.

\[
Y_k = \bar{p}_0 A_k + \bar{p}_1 A_{k-1} + \bar{p}_2 A_{k-2} + N_k
\]

\[
= A_k \cdot \bar{p}_k + N_k
\]

\[
= S_k + N_k
\]

Then, the convolution matrix \( P \) reads (why?)

\[
\begin{bmatrix}
Y_1 \\
Y_2 \\
Y_3 \\
: \\
Y_{L+2}
\end{bmatrix}
= \begin{bmatrix}
\bar{p}_0 & 0 & \cdots & 0 & 0 \\
\bar{p}_1 & \bar{p}_0 & 0 & \cdots & 0 \\
\bar{p}_2 & \bar{p}_1 & \bar{p}_0 & 0 & \cdots & 0 \\
0 & \bar{p}_2 & \bar{p}_1 & \bar{p}_0 & 0 & \cdots & 0 \\
& & & & & \vdots & \vdots
\end{bmatrix}
\begin{bmatrix}
A_1 \\
A_2 \\
A_3 \\
: \\
A_L
\end{bmatrix}
+ \begin{bmatrix}
N_1 \\
N_2 \\
N_3 \\
: \\
N_{L+2}
\end{bmatrix}
\]

(… showing also the start and end transients.)

Notice that in real receivers, the ISI profile \( \bar{p}_k \) is not trivially known, but can be estimated using known pilot or reference symbols:

- we will address the channel estimation aspects later in the course
Reducing the computational complexity through state machine based modeling

Previous solution builds on explicit comparison of all possible sequence candidates

- $M^L$ different sequences

For example $M = 4$ and $L = 1000 => M^L = 4^{1000} \approx 1000000^{100}$

- Terrible 😊
- Not feasible at all

So we must think how to reduce the complexity, while hopefully not compromising the performance (yielding same end results).

One approach is to model the ISI process as a state machine, which then finally leads to the celebrated Viterbi algorithm

- Used also in other contexts, e.g., in decoding convolutional error correction codes in receiver
- We will address the error correction codes and their decoding later in the course

Intersymbol interference as a finite state machine

Here we seek to model the system containing ISI, reproduced below

![Intersymbol interference as a finite state machine diagram](image)

The only assumption is that the ISI profile $\bar{p}_k$ is to be of finite length, say $D$ symbols (like already mentioned earlier)

- not any real limitation, as practically relevant ISI comes over finite time window anyways

Then, we can model the system as the following shift-register process:

![Shift-register process diagram](image)

Here the state of the system is denoted by $\Psi_k = \{A_{k-1}, \ldots, A_{k-D+1}\}$ and as before

$$S_k = \sum_{i=0}^{D-1} \bar{p}_i A_{k-i}$$
Examples (... thank god)

Example 1: binary symbols ($\Omega_A = \{0,1\}$) and ISI over two consecutive symbols with an example profile of

$$\bar{p}_k = \delta_k + 0.5\delta_{k-1} \Rightarrow S_k = A_k + 0.5A_{k-1}$$

Now we can do the modeling as follows:

- Shift-register model

```
+----+        +----+
| A_k |        | A_{k-1} |
| z^{-1}|        |          |
|      |        |          |
+----+        +----+
          |        |
          |        |
| A_k + 0.5A_{k-1} |          |
          |        |
          |        |
      +----+        +----+
      | S_k |        |          |
      +----+        +----+
```

- System states, and their possible evolution over one bit/symbol interval:

```
STATE 0          STATE 1
(0, 0)           (1, 1)
(0, 0.5)         (1, 1.5)
(0, 1.1)         (1, 1.2)
```

The latter graph is called state-transition diagram, and shows also which input triggers which state transition and what’s the corresponding noiseless output of the state machine model.

Examples (cont’d)

Example 2: binary symbols ($\Omega_A = \{0,1\}$) and ISI over three consecutive symbols with an example profile of

$$\bar{p}_k = \delta_k + 0.5\delta_{k-1} - 0.3\delta_{k-2} \Rightarrow S_k = A_k + 0.5A_{k-1} - 0.3A_{k-2}$$

- Shift-register model:

```
+----+        +----+
| A_k |        | A_{k-1} |
| z^{-1}|        |          |
|      |        |          |
+----+        +----+
          |        |
          |        |
| A_k + 0.5A_{k-1} - 0.3A_{k-2} |          |
          |        |
          |        |
      +----+        +----+
      | S_k |        |          |
      +----+        +----+
```

- State-transition diagram:

```
STATE 0,0         STATE 0,1
(0, 0)            (0, 0.5)
(0, 0.2)          (0, 0.7)
```

Notice that in general, only selected state-transitions (like in above graph) are possible

- why?
Trellis diagram

Trellis diagram is another way to represent the evolution of the states.

For the previous example #1, it looks like:

Then, when considering an actual sequence of $L$ symbols, all the possible occurrences can be represented as paths through the following combined trellis diagram:

One stage in the above complete trellis diagram corresponds to single symbol period.

As a whole, the above graph shows all the possible scenarios and state transitions over time, from initial state $\Psi_0$ to final state $\Psi_L$.

Each path through the trellis diagram corresponds to a different input symbol sequence.

ML sequence detector based on trellis diagram

For given observed sequence $Y_k = A_k * \bar{p}_k + N_k = S_k + N_k$, as we already learned, the sequence detector seeks to make a decision of the complete transmitted sequence as a whole.

In trellis diagram, this corresponds to finding the correct path through the trellis.

According to ML principle, this corresponds to the path with minimum path metric

- assuming uncorrelated Gaussian noise samples, this corresponds to the path with minimum Euclidian distance compared to observed sequence

Each branch has thus a branch metric or weight of the form

$$\text{Branch metric} = \|y_k - s_k\|^2 ; \text{uncorrelated Gaussian noise}$$

The overall path metric is always the sum of branch metrics (why?).

(... because: $\|y - s\|^2 = \|y_1 - s_1\|^2 + \|y_2 - s_2\|^2 + \ldots + \|y_L - s_L\|^2$)

Thus you can implement the ML sequence detector as:

- calculate all branch metrics
- calculate all corresponding path metrics
- choose the minimum and read the corresponding input symbol sequence as the decision
VITERBI ALGORITHM

Previous calculations are still terribly complex, especially for long sequences.

Viterbi algorithm is then an efficient solution to solve the problem, with reduced complexity, by deploying the principle of dynamic programming (from applied math).

The principle is as follows:

1) Trellis diagram is processed from the initial state to the closing state, as before.

2) But in each stage, only the shortest paths, meaning the paths with lowest cumulative metrics so far, from initial state to the possible input states of the next stage, are stored.

   Everything else is USELESS! (can be thrown away)

3) Then, you calculate all the branch metrics of the next stage, and repeat.

Thus, in any given stage, you store only \( N \) paths and their cumulative path metrics at that point, where \( N \) denotes the number of parallel states in the trellis:

\[
N = M^{D-1}
\]

Eventually, there is only one path remaining, from initial state to final state, i.e., the one with lowest overall cumulative metric:

- This is the final survivor, and you read the decision sequence from the corresponding transitions.

We illustrate with an example.

Example on Viterbi algorithm

Below is an example for the previous ISI profile \( p_k = \delta_k + 0.5\delta_{k-1} \) (used also in the earlier examples). Furthermore, the received noisy and ISI containing sequence is assumed to be \( y_k = \{0.2, 0.6, 0.9, 0.1\} \).

Diagram shows the branch metrics and survivors and their cumulative metrics at each stage:

Thus the final ML sequence decision for the input symbols is: \( \{0, 1, 0, 0\} \).
Further notes on Viterbi algorithm

Viterbi algorithm is particularly nice in the sense that there is a fixed complexity for each stage. Thus the overall complexity is linear in sequence length $L$ (instead of being exponential in it, like in the brute force comparisons).

One problem, on the other hand, is that in general we need to wait for the decision of the whole sequence until the very end.

- **Long detection delay/latency**, common problem in all sequence detectors processing the sequence as a whole.

In practice, to limit the latency, truncated version of the Viterbi algorithm is commonly deployed.

- You start to force decisions after say $d$ first stages
  - Using cumulative metrics so far, forcing the decision on the first input symbol
- Then continue as earlier, i.e., calculate new branch metrics, but again force next decision for the next input symbol, and continue

This way the detection delay is constant ($d$ symbols), and as long as the truncation depth $d$ is big enough, this does not substantially impact the performance (though is strictly speaking already sub-optimal).

Further notes on Viterbi algorithm, cont’d

Viterbi algorithm has large amount of applications in engineering and applied math:

- we’ll use it later for decoding certain error control codes, in the later parts of the course
- but can be applied also e.g. for speech recognizion, pattern recognizition and all engineering problems where a sequence of states of a Markov state machine are to be tracked based on observed events.

Next, we’ll look at alternative methods to handle ISI in receivers, namely different channel equalization schemes.
**LINEAR EQUALIZER (LE)**

Idea: remove or reduce the amount of ISI in the received discrete-time signal using a discrete-time linear filter $C(z)$. After this, make decisions about the transmitted symbols, symbol-by-symbol. This allows for substantially reduced complexity compared to MLSD.

Regarding how to choose or optimize the equalizer coefficients, there are two main principles:

- **LE-ZF** zero-forcing, ZF, approach; seek to force ISI to zero (remove all ISI) at equalizer output signal $Q_k$
- **LE-MSE** mean-squared error, MSE, approach; minimize the average squared error, compared to transmitted symbol values, at the equalizer output

$$
\text{MSE} = E[(A_k - Q_k)^2], \quad \text{where } E[.] \text{ refers to statistical expectation (averaging)}
$$

Some immediate notes:

- Since the equalizer is always processing a noisy signal of the form $Y_k = A_k \ast \overline{P}_k + N_k$, also the noise is processed by the equalizer  
  - hence, depending on the equalizer response, the noise may be substantially boosted at certain frequencies
  - for this reason, MSE is a better criterion since it takes both the residual ISI and noise at equalizer output into account
- Also, practical equalizer filters have finite complexity (e.g. a finite length FIR filter), so it may not be always feasible to completely remove all ISI in this respect anyway

---

**LE-ZF, transfer function based reference solution**

Relying on our previous discrete-time received signal model of the form

$$
Y_k = A_k \ast \overline{P}_k + N_k
$$

it is easy to see that the ideal ZF linear equalizer has the following z-domain transfer function:

$$
C_{ZF}(z) = \frac{1}{\overline{P}(z)}
$$

ZF equalizer is conceptually very self-explaining and intuitive – it fully removes all ISI through pure inverse filtering.

Thus, the total equalized system transfer function becomes

$$
C_{ZF}(z)\overline{P}(z) = 1
$$

- practical filters always impose a delay, normalized to zero above

**Example:** We continue our example with $\overline{P}(z) = b^*z + 1 + |b|^2 + bz^{-1}$, assume $|b| < 1$, and thus the LE-ZF reads

$$
C_{ZF}(z) = \frac{1}{\overline{P}(z)} = \frac{1}{1 + bz^{-1}}
$$

- stable causal recursive filter, pole at $z = -b$
LE-ZF, transfer function based reference solution, cont’d

In general, if there is a noise whitening filter $1 / W^*(1/z)$ in the receiver, it appears in cascade with the linear equalizer filter, and thus in reality only one combined filter can be adopted of the form

$$C(z) = \frac{1}{W^*(1/z)} C(z)$$

In the LE-ZF case, this reads

$$C_{ZF}(z) = \frac{1}{W^*(1/z)} C_{ZF}(z) = \frac{1}{W^*(1/z)} \frac{1}{P(z)} = \frac{1}{P(z)}$$

In other words, the combined noise whitening filter and LE-ZF is nothing but the inverse of the original baseband equivalent channel $P(z)$

- Intuitive
- Notice that strictly-speaking, like we discussed already earlier, this combined filter can be stable only as a cascade of anticausal and causal subfilters (because of the nature of the roots of $P(z)$)
- In practice, a truncated and delayed causal approximation can be used
- Notice also that the final form applies directly to the more practical receiver with square-root raised cosine receiver filter

**Example:** We again continue our simple example where $P(z) = b^*z + 1 + |b|^2 + bz^{-1}$. Then, the combined noise whitening filter and LE-ZF equalizer $z$ transfer function reads

$$C_{ZF}(z) = \frac{1}{W^*(1/z)} C_{ZF}(z) = \frac{1}{W^*(1/z)} \frac{1}{P(z)} = \frac{1}{P(z)}$$

Concrete illustrations of the responses are given, in terms of the frequency responses and (truncated) impulse responses, in an example case of $b = 0.8$, next pages.
LE-ZF, transfer function based reference solution, cont’d

Notice the substantial noise enhancement that will be imposed by the ZF equalizer at those frequencies where the unequalized system has low response values!!

LE-MSE, transfer function based reference solution

For the same discrete-time received signal model of the form

\[ y_k = A_k * \bar{p}_k + N_k \]

the corresponding MSE-based linear equalizer has the following transfer function (derivation available e.g. in the course book, not difficult to derive):

\[ C_{MSE}(z) = \frac{\overline{P^*(1/z)}}{P(z)\overline{P^*(1/z)} + \sigma_N^2 / \sigma_A^2} \]

MSE-based solution seeks for a balance between residual ISI and filtered noise at equalizer output.

Notation-wise:

- \( \overline{P^*(1/z)} \) stands for the z-transfer function of \( \overline{p_k} \), i.e., conjugated and mirrored impulse response of the white(ned) noise system.
- \( \sigma_A^2 = E]\left[|A_k|^2\right] \) and \( \sigma_N^2 = E]\left[|N_k|^2\right] \) stand for the symbol power and white noise power, respectively, hence \( \sigma_N^2 / \sigma_A^2 \) is relative to the inverse of the SNR.
- notice that in the limiting case of \( \sigma_N^2 / \sigma_A^2 \to 0 \), the MSE equalizer tends towards the ZF equalizer (do you see it?)

The total equalized system transfer function becomes now

\[ C_{MSE}(z)P(z) = \frac{\overline{P(z)P^*(1/z)}}{\overline{P(z)P^*(1/z)} + \sigma_N^2 / \sigma_A^2} = \frac{P(z)}{P(z) + \sigma_N^2 / \sigma_A^2} \]

- this is not exactly a unity transfer function, thus there is some residual ISI at MSE equalizer output (assuming \( \sigma_N^2 / \sigma_A^2 \approx 0 \))
- practical filters always impose a delay, which is again normalized to zero above
LE-MSE, transfer function based reference solution, cont’d

Similar to the LE-ZF case, if there is a noise whitening filter 1 / \(W^*(1 / z)\) in the receiver, we can again implement the combination of the noise whitening filter and the actual equalizer as one combined filter of the form

\[
\overline{C}_{MSE}(z) = \frac{1}{W^*(1 / z)} C_{MSE}(z)
\]

\[
= \frac{1}{W^*(1 / z) \overline{P}(z) \overline{P}^*(1 / z) + \sigma_N^2 / \sigma_A^2}
\]

\[
= \frac{1}{P(z) + \sigma_N^2 / \sigma_A^2}
\]

where the last form follows from \(P(z) = P(z) / W^*(1 / z) = W(z)\) and \(P(z) = W(z)W^*(1 / z) = \overline{P}(z) \overline{P}^*(1 / z)\).

**Example:** We again continue our simple example where
\(P(z) = b^* z + 1 + |b|^2 + b^{-1}\). Then, the combined noise whitening filter and LE-MSE equalizer z transfer function reads

\[
\overline{C}_{MSE}(z) = \frac{1}{P(z) + \sigma_N^2 / \sigma_A^2}
\]

\[
= \frac{1}{b^* z + 1 + |b|^2 + b^{-1} + \sigma_N^2 / \sigma_A^2}
\]

Concrete illustrations of the frequency responses are given, in an example case of \(b = 0.8\), next page, with two different values for the inverse of the SNR \(\sigma_N^2 / \sigma_A^2 = \{0.1, 0.01\}\).
FRACTIONALLY SPACED EQUALIZER (FSE)

Assuming that the excess bandwidth/rolloff is < 100% (this is practically always the case), the baseband signal bandwidth always fulfills

$$W = (1 + \alpha) \frac{1}{2T} < 1/T$$

Thus, if sampling the received signal at rate 2 x symbol rate
- there is no aliasing, and the analog receiver filter can be a normal lowpass filter (LPF) which removes noise and interference outside the bandwidth $-1/T...1/T$

Then, the actual receiver pulse-shape filtering (either the true matched filter or square-root raised coside) can be implemented using discrete-time filtering, followed by decimation by 2 to symbol rate samples.
- this is a more practical receiver structure which then allows also for, e.g., digital symbol timing recovery and adjustment, along the resampling process

In an actual implementation, the first sample rate can also be higher than 2x symbol rate, but in communication theoretic sense, 2x symbol is sufficient since that anyway allows avoiding aliasing.

The actual so-called fractional-spaced equalizer, FSE, is obtained when the symbol rate linear equalizer and the discrete-time receiver filter are combined to one filter, running at 2x symbol rate:

Decision feedback equalizer is an extension of the basic linear equalizer (LE) where two equalization filters, forward filter and feedback filter, are adopted.

It builds on the fact that, in general, both the earlier and coming symbols contribute to the ISI at any symbol instant
- if symbol decisions are made symbol-by-symbol, there are already decisions about the earlier symbols available when calculating the current equalized sample
- ... and assuming that these decisions are correct, at least most of the time, they can be used for reconstructing and cancelling ISI due to earlier symbols through feedback

Thus, the feedback filter is used to remove the ISI due to earlier symbols while the feedforward filter is removing the ISI due to coming symbols
- as the forward filter does not have to remove all ISI, it has smaller noise enhancement compared to ordinary LE
- assuming correct symbol decisions, also feedback part is noiseless

On the other hand, symbol decisions are never 100% correct (error probability is never zero) in reality, and thus there is a fundamental error propagation problem
- incorrect symbol decisions corrupt ISI suppression and may result in erroneous detection of the current symbol as well
Zero-Forcing DFE

The concept of ZF-DFE is basically easy: use the forward filter $C(z)$, or the combined noise whitening filter + the forward filter, to force the ISI due to coming symbols to zero

- this will then leave only ISI due to earlier symbols, which can easily be handled by the feedback filter (assuming the symbol decisions are correct)

In terms of the system models, we can think of the zero-forcing DFE as follows:

That is,

- what ever is the ISI profile $P(z)$, or the corresponding (whitened) system transfer function $P(z)$, compose the forward filter as

\[ C(z) = \frac{1}{P(z)}(1 + D(z)) \]

- this is the general design principle - the only limitation is that the transfer function $D(z)$ must be purely causal, because the corresponding filter is sitting in the feedback loop

Notice that there are multiple solutions to above design principle for any given $P(z)$ that yield zero pre-cursor ISI

- thus choose the one, i.e., that particular $D(z)$, that minimizes the noise variance at the detector input

Zero-Forcing DFE, cont’d

In the case of a true matched filter based receiver, followed by noise whitening filter $1 / W^*(1 / z)$, finding the zero-forcing DFE filters becomes almost trivial.

This is because the noise whitening filter $1 / W^*(1 / z)$ itself removes the pre-cursor ISI (see our earlier discussions) and thus leaves only a causal post-cursor ISI profile

- by whitening the noise, it will also, by definition, yield a minimum noise variance at its output (compared to any other filter)

**Example:** We again continue our previous example with

\[ P(z) = b^*z + 1 + |b|^2 + bz^{-1}, \]

and assume $|b| < 1$. In this case, a stable anti-causal noise whitening filter was found to be

\[ \frac{1}{W^*(1 / z)} = \frac{1}{1 + b^*z} \]

This will then yield an effective system model or ISI profile as

\[ \tilde{P}(z) = P(z) \frac{1}{W^*(1 / z)} = W(z) = 1 + bz^{-1} \]

Thus, this means that

\[ C(z) = 1 \]

and

\[ D(z) = bz^{-1} \]

Easy, isn’t it?
Zero-Forcing DFE, cont'd

Thus, in general, the combined noise whitening and forward filter in the ZF-DFE reads

\[
\tilde{C}(z) = C(z) \frac{1}{W^*(1/z)} = \frac{1}{W^*(1/z)}
\]

up to a possible constant scaling factor.

That is, if the complex gain of the direct useful signal term at detector input is to be normalized to one, and if the noise whitening filter does not automatically yield that, an additional complex scaling is adopted.

**Example:** We again continue our previous example with
\[
P(z) = b^*z + 1 + |b|^2 + b^{-1},
\]
but now assume \(|b| > 1\). In this case, a stable anti-causal noise whitening filter was found to be

\[
\frac{1}{W^*(1/z)} = \frac{1}{b + z}
\]

This will now yield an effective system model or ISI profile of the form

\[
\tilde{P}(z) = P(z) \frac{1}{W^*(1/z)} = W(z) = b^* + z^{-1}
\]

Thus, if the gain of the direct term is to be normalized to one, this means that

\[
C(z) = 1 / b^*
\]

and thus

\[
D(z) = \frac{1}{b^*} z^{-1}
\]

Some intermediate notes

The previous transfer function based solutions do not directly concern the complexity of the obtained equalization filter or filters

- easily leads to infinitely long filters, in terms of impulse responses

Hence, we will next shortly revisit the same equalization problem but assume, from the beginning, that the equalizer filter(s) is (are) finite tap FIR filter(s)

- this then means optimization of the finite amount of filter coefficients to, e.g., minimize MSE or minimize ISI, within the constrained complexity

- we'll cover both LE and DFE cases, assuming finite length FIR filters

Furthermore, all previous expressions assume that the discrete-time baseband equivalent system response, \(\tilde{p}_k\) or \(\tilde{P}(z)\), is known

- needs to be estimated in practice; this is the channel estimation problem

- we will also address this soon, in the channel estimation part

- channel estimation is also needed in the earlier MLSD based receiver to be able to calculate the branch metrics
FIR FILTER BASED LINEAR EQUALIZERS

Suppose we now reformulate the linear equalization problem such that the equalizer is restricted to be a finite length FIR filter, say of the form

$C(z) = \sum_{n=-N_1}^{N_2} c_n z^{-n} = c_{-N_1} z^{N_1} + \ldots + c_0 + \ldots + c_{N_2} z^{-N_2}$

- impulse response length is $N_1 + N_2 + 1$ taps
- to simplify the notations and the block-diagram, the noise whitening filter is not shown or is not assumed to be used
- notice that the above formulation is anti-causal, that is, the filter is delayless
- this is only to simplify the notation, a practical FIR filter would have a delay of $N_1$ samples

The purpose is then to optimize the finite set of coefficients of the FIR equalizer filter such that
- ISI is minimized, or
- MSE is minimized

For mathematical convenience, we adopt vector-matrix representation of linear discrete-time filtering in the continuation
- only a mathematical convention which allows for easy and feasible presentation and solving the optimization problems
- … take a short recap of vectors and matrices in your basic engineering math course book (if need to refresh your memory)

FINITE LENGTH FIR ZF LINEAR EQUALIZER

We assume that the discrete-time baseband equivalent system model $P(z)$ is also of finite length, say of length $L_1 + L_2 + 1$ (or that only a finite number of ISI terms are considered), written as

$P(z) = \sum_{n=-L_1}^{L_2} p_n z^{-n} = p_{-L_1} z^{L_1} + \ldots + p_0 + \ldots + p_{L_2} z^{-L_2}$

- $L_1$ precursor and $L_2$ postcursor ISI terms
- $p_0$ represents the coefficient of the current useful symbol, i.e., the one relative to which ISI is defined
  - $p_0$ is the strongest tap in $P(z)$ in practice

Then, for a finite length FIR equalizer, the total equalized system transfer function is of the form

$C(z)P(z) = \left( \sum_{n=-N_1}^{N_1} c_n z^{-n} \right) \left( \sum_{n=-L_1}^{L_2} p_n z^{-n} \right)$

This total equalized system has an impulse response whose length is $N_1 + L_1 + N_2 + L_2 + 1$ (why?)

The taps of the total equalized system impulse response can be conveniently expressed through a vector-matrix product as shown on the following page, through a concrete example
Finite length FIR ZF linear equalizer, cont’d

**Example:** suppose we consider a simple example where the discrete-time baseband equivalent system comprises only 3 taps, that is

\[ P(z) = p_{-1}z + p_0 + p_1z^{-1} \]

Then, the impulse response values of the total equalized system

\[ C(z)P(z) = \left( \sum_{n=-N_1}^{N_2} c_n z^{-n} \right) \left( p_{-1}z + p_0 + p_1z^{-1} \right) \]

can be easily expressed as a column vector through a vector-matrix product as

\[
\begin{pmatrix}
p_{-1} & 0 & \ldots & 0 & 0 \\
p_0 & p_{-1} & 0 & \ldots & 0 \\
p_1 & p_0 & p_{-1} & 0 & \ldots & 0 \\
0 & p_1 & p_0 & p_{-1} & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \ldots & 0 & p_1 & p_0 & p_{-1} \\
0 & \ldots & 0 & p_1 & p_0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & p_1 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 0 & \ldots & p_1 \\
\end{pmatrix}
\begin{pmatrix}
c_{-N_1} \\
c_{-N_1+1} \\
\vdots \\
c_{N_2-1} \\
c_{N_2} \\
\end{pmatrix}
\]

where the column vector \( c \) collects the impulse response coefficients of the equalizer while \( P \) is basically a convolution matrix obtained from the impulse response coefficients of the discrete-time baseband equivalent system.

While the above example is for a simple 3-tap discrete-time system model, you can easily see how the structure of the matrix \( P \) generalizes to longer impulse responses \( p_k = p(kT) \), right?

Finite length FIR ZF linear equalizer, cont’d

Now, to try to force ISI to zero, we should choose the equalizer coefficients \( c \) such that the total equalized system impulse response, written previously in vector form as \( Pc \), is all zeros except for the tap corresponding to the current symbol.

We’ll write this target impulse response in vector form as

\[
\begin{pmatrix}
0 \\
\vdots \\
0 \\
1 \\
0 \\
\vdots \\
0 \\
\end{pmatrix}
\]

\[ N_1 + L_1 \]

\[ N_2 + L_2 \]

Notice that the location of the non-zero tap (location of the entry '1') reflects the equalized weight for the current symbol.

Hence, in general, the equalized system impulse response has \( N_1 + L_1 \) pre-cursor taps and \( N_2 + L_2 \) post-cursor taps.

However, since the equalizer has only \( N_1 + N_2 + 1 \) coefficients, such optimization problem of the form \( Pc = u_{zf} \), is an over-determined system of linear equations (more equations than unknowns) and cannot be solved exactly, in general.

Instead, we can then adopt a linear least-squares approach and obtain such equalizer coefficients \( c_{LS} \) which minimize the squared error norm \( \| Pc - u_{zf} \| \), i.e., the sum of the squared errors between the target equalized impulse response and realized equalized impulse response.

This is a well-known problem in applied math, and the solution is

\[
c_{LS} = (P^H P)^{-1} P^H u_{zf}
\]

where superscript \((.)^H\) denotes conjugate transpose and \((.)^{-1}\) denotes matrix inverse.

We’ll illustrate with an example, soon.
An alternative approach is to explicitly consider only selected \( N_1 + N_2 + 1 \) taps of the impulse response of the total equalized system.

\[
C(z)P(z) = \left( \sum_{n=-N_1}^{N_2} c_n z^{-n} \right) \left( \sum_{n=-L_1}^{L_1} p_n z^{-n} \right)
\]

and force ISI explicitly to zero in this window:

- this can be done, exactly, since the equalizer filter has \( N_1 + N_2 + 1 \) coefficients
- however, the impulse response values outside this window are not explicitly controlled in this alternative approach.

Example: We continue our previous example with \( P(z) = p_{-1} z + p_0 + p_1 z^{-1} \) and the corresponding convolution matrix shown on page 318 (also below). The full impulse response samples of the total equalized system are described by \( \bar{P}c \). Then, the window of considered \( N_1 + N_2 + 1 \) impulse response taps read formally

\[
\bar{P}c
\]

where \( \bar{P} \) refers to a sub-matrix obtained from \( P \) as shown below:

\[
\begin{bmatrix}
0 & 0 & \ldots & 0 & 0 \\
p_{-1} & 0 & \ldots & 0 & 0 \\
p_0 & p_{-1} & 0 & \ldots & 0 \\
p_1 & p_0 & p_{-1} & 0 & \ldots & 0 \\
0 & p_1 & p_0 & p_{-1} & 0 & \ldots & 0 \\
0 & \ldots & 0 & p_1 & p_0 & p_{-1} \\
0 & \ldots & 0 & p_1 & p_0 \\
0 & 0 & \ldots & 0 & p_1 \\
\end{bmatrix}
\]

To construct \( \bar{P} \), take \( N_1 + N_2 + 1 \) rows from \( P \) that correspond to the considered window of impulse response samples (\( N_1 \) rows above and \( N_2 \) rows below the row \( N_1 + L_1 + 1 \)).

Then, to force the ISI exactly to zero within this window of \( N_1 + N_2 + 1 \) equalized taps, we write the target impulse response, in vector form, as

\[
\tilde{u}_{xf} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}
\]

Again, the entity '1' corresponds to the equalized weight of the current symbol while then \( N_1 \) precursor ISI taps and \( N_2 \) postcursor ISI taps are forced explicitly to zero.

Then, we choose the equalizer coefficients such that

\[
\bar{P}c = \tilde{u}_{xf}
\]

holds, which yields

\[
c_{INV} = \bar{P}^{-1} \tilde{u}_{xf}
\]

Above expression assumes that \( \bar{P} \) is an invertible (full-rank) matrix.

Both approaches (\( c_{LS} \) and \( c_{INV} \)) assume that the ISI coefficients \( p_k = p(kT) \) are known:

- obtained through channel estimation in practice
- we'll address that soon
Finite length FIR ZF linear equalizer, cont'd

We'll consider a practical example where 16-QAM data is transmitted over a frequency-selective multipath channel at symbol rate $R_{sym} = 10$ MHz with square-root raise cosine pulse-shape filtering in both TX and RX (similar example was considered already much earlier in the lecture notes, on pages 132-134)

- without a frequency-selective channel, the pulse-shape at the RX filter output would be a beautiful raised-cosine Nyquist pulse
- however, the example frequency-selective channel will distort the pulse-shape badly and create intersymbol interference, ISI
- notice that the fading response is non-symmetric around zero frequency, when interpreted for I/Q downconverted RX signal

The overall distorted pulse-shape $p(t)$ is illustrated in the below figure

- can you now read or interpret the ISI coefficients $p_k = p(kT)$ from the figure?
Finite length FIR ZF linear equalizer, cont’d

We’ll then use the previous FIR zero forcing equalization approach, with 3 different equalizer lengths of
- 21 taps \( N_1 = N_2 = 10 \),
- 41 taps \( N_1 = N_2 = 20 \),
- 101 taps \( N_1 = N_2 = 50 \)
to equalize the ISI.

As a concrete example, we use the previous Least-Squares based fitting approach of the form

\[
c_{LS} = (P^H P)^{-1} P^H u_{zf}
\]

to find the equalizer coefficients.

In the following figures, we’ll then show the equalizer response, total equalized system response, and equalized received symbols (without noise, such that the residual ISI can be visually interpreted)
Finite length FIR ZF linear equalizer, cont’d

FIR equalizer length 41 taps:
- already close to perfect ISI suppression

FIR equalizer length 101 taps:
- ISI suppression is practically perfect, essentially a pure inverse filter (despite the limited degrees of freedom)
FINITE LENGTH FIR MSE LINEAR EQUALIZER

After some simple manipulations, the MSE can be written as

\[
MSE = E\left[ \left| A_k - Q_k \right|^2 \right] = E\left[ \left| A_k - c^T r_k \right|^2 \right] = J_{MSE}
\]

where

- \( \sigma_A^2 = E\left[ \left| A_k \right|^2 \right] \) denotes the symbol power
- \( \alpha = E\left[ A_k r_k^* \right] \) denotes the cross-correlation vector between the transmit symbol and the received samples
- \( \Phi = E\left[ r_k r_k^H \right] \) denotes the autocorrelation matrix of the received samples

Luckily, the above MSE expression is a well-behaving quadratic function of the equalizer coefficients, and can thus be minimized through ordinary gradient-based minimization

\[
\frac{\partial J_{MSE}}{\partial c} = \nabla_c J_{MSE} = ...
\]

\[
= -2\alpha + 2\Phi^* c = 0
\]

This derivative and the corresponding minimum MSE solution can be shown to read

\[
\mathbf{c}_MSE = (\Phi^*)^{-1} \alpha
\]
Finite length FIR MSE linear equalizer, cont’d

Based on the previous slide, the coefficients of the MSE minimizing linear equalizer are essentially obtained as a solution to a set of linear equations.

For our later purposes, it is also convenient to observe that the gradient or derivative of the MSE can be alternatively expressed as

$$\frac{\partial J_{\text{MSE}}}{\partial c} = \nabla_c J_{\text{MSE}} = -2\alpha + 2\Phi^* c = -2E[E_k r_k^*]$$

where $E_k = A_k - Q_k$.

Based on this, when $c = c_{\text{MSE}}$, it holds that

$$E[E_k r_k^*] = 0 \quad \text{(when } c = c_{\text{MSE}})$$

This is commonly called the orthogonality criterion or feature of the MSE optimum filter

- i.e., the equalization error $E_k = A_k - Q_k$ is uncorrelated/orthogonal to the equalizer input samples when the equalizer coefficients are the MSE optimum coefficients

In practice, $\alpha = E[A_k r_k^*]$ and $\Phi = E[r_k r_k^H]$ are unknown (depend on the channel response, which is unknown, and SNR) but can be estimated using, e.g., known reference symbols

- we’ll touch that shortly, in the channel estimation part
- we’ll first illustrate the FIR MSE equalizer principle with an example

Finite length FIR MSE linear equalizer, cont’d

We’ll again continue our previous example, with 21 taps or 101 taps in the FIR equalizer, and assume received signal SNR = 15 dB

- clearly, the MSE equalizer is not exactly a pure inverse at those frequencies where the channel is notching the signal very badly

Discrete-Time Effective Channel Responses

21 taps equal.

101 taps equal.
FINITE LENGTH FIR DECISION FEEDBACK EQUALIZER

We'll next visit the DFE equalizer principle, and assume now that both the forward filter $C(z)$ and the feedback filter $D(z)$ are FIR filters.

For notational convenience and because of the operating ideology of the DFE principle, we assume the following FIR structures

$$C(z) = \sum_{m=-N+1}^{0} c_m z^{-m}, \quad D(z) = \sum_{m=1}^{M} d_m z^{-m}$$

- forward filter $C(z)$ is purely anticausal, $N$ taps
- feedback filter $D(z)$ is strictly causal, $M$ taps

In a practical implementation, the forward filter is delayed to be causal which thus imposes a similar delay on the detected symbols.

Next, we shortly address the optimization of the FIR filters, from both MSE and ZF criterion perspective (more explicit focus on MSE based optimization, though).

Finite length FIR DFE, cont'd

Assuming now the previous structures for the forward filter and the feedback filter, the current output sample of the decision feedback equalizer structure reads (assuming, for notational simplicity, that the decisions are correct, $\hat{A}_k = A_k$)

$$Q_k = c^T r_k - d^T a_k$$

$$= w^T u_k$$

where the associated vectors are of the form $r_k = [R_{k+N-1}, \ldots, R_0]^T$, $c = [c_{-N+1}, \ldots, c_0]^T$, $a_k = [A_{k-1}, \ldots, A_{k-M}]^T$ and $d = [d_1, \ldots, d_M]^T$.

Furthermore, $w$ stacks the forward filter and feedback filter coefficients while $u_k$ stacks the corresponding input samples and detected symbols as

$$w = [c_{-(N-1)}, \ldots, c_0, -d_1, \ldots, -d_M]^T$$

$$u_k = [R_{k+N-1}, \ldots, R_k, A_{k-1}, \ldots, A_{k-M}]^T$$

Then, the equalization error reads $e_k = A_k - Q_k = A_k - w^T u_k$, and the minimum MSE coefficients minimizing $\text{MSE} = E\|A_k - w^T u_k\|^2$ are directly given by

$$w_{\text{MSE}} = (\Phi^*)^{-1} \alpha$$

where

$$\alpha = E[A_k u_k^T]$$

$$\Phi = E[u_k u_k^H]$$

The proof follows directly from the earlier linear MSE equalizer case, as formally the DFE based processing can be written in a similar seemingly linear form $Q_k = w^T u_k$ when correct symbol decisions are assumed.
Finite length FIR DFE, cont’d

Again, with known training data (known reference sequence of $A_k$’s), sample estimates of $\alpha$ and $\Phi$ can be directly used to estimate the above statistics, and thereon to calculate the equalizer coefficients.

Alternatively, ZF design criterion can also be adopted in optimizing the DFE filter coefficients. In this case, I’ll skip the exact math but will give you the following hints how to do it

- for the feedforward filter

$$C(z) = \sum_{m=-(N-1)}^{0} c_m z^{-m}$$

you can follow the earlier linear equalizer ZF design approaches but simply take into account the following when composing the vector matrix representation (for “LS” or “INV” based solution) in solving for the coefficients

- the feedforward filter (above) is purely anticausal, and you only need to try to force the pre-cursor ISI to zero, hence take this into account when composing the convolution matrix ($P$ or $P^*$) and the target vector $u_x$ or $u_x^*$

- otherwise, everything works exactly like in case of LE

- for the feedback filter (after solving for the feedforward filter coeff)

$$D(z) = \sum_{m=1}^{M} d_m z^{-m}$$

you can directly choose the coefficients $d_m$ such that they cancel the remaining post-cursor ISI that is there after our feedforward filter $C(z)$ (designed above) has processed the signal

---

CHANNEL ESTIMATION ASPECTS, shortly

All the previous equalization solutions as well as the MLSD/Viterbi processing depend, explicitly or implicitly, on the impulse response values of the discrete-time baseband equivalent channel $p_k = p(kT)$

So the question is: how to estimate this response in practice?

We again build on the basic received signal model, for simplicity without noise whitening filter, reproduced below as

$$R_k = A_k * p_k + Z_k$$

Now, if we assume that $p_k = p(kT)$ is of finite length, and that the transmitter sends a known reference sequence (i.e., we know the transmit symbols $A_k$ over certain time window), we can easily estimate $p_k = p(kT)$ through, e.g., least-squares based model fitting.

Example: Assume, as a simple example, that we want to estimate a discrete-time baseband equivalent system which comprises of 3 taps, i.e., $P(z) = p_{-1}z + p_0 + p_1z^{-1}$

Then, for such system, the noiseless received signal would read $p_{-1}A_{k+1} + p_0A_k + p_1A_{k-1}$, and we can write a vector matrix model for the noiseless received samples over a block of $M$ reference symbols as

$$\begin{bmatrix} A_1 & 0 & 0 \\ A_2 & A_1 & 0 \\ A_3 & A_2 & A_1 \\ \vdots & \vdots & \vdots \\ A_M & A_{M-1} & A_{M-2} \\ 0 & A_M & A_{M-1} \\ 0 & 0 & A_M \end{bmatrix} \begin{bmatrix} p_{-1} \\ p_0 \\ p_1 \end{bmatrix} = Ap$$
Channel estimation aspects, shortly, cont’d

By stacking the corresponding true observed signal samples at receiver into a vector $r$ over the block of $M$ reference symbols, the least squares model fitting based estimate for the channel impulse response then reads

$$\hat{p}_{LS} = (A^H A)^{-1} A^H r$$

This is, by definition of the LS model fitting or parameter estimation, such vector $\hat{p}$ that minimizes

$$\|r - A\hat{p}\|^2$$

Some notes:

- the previous page example regarding the structure of the matrix $A$ builds on the assumption of estimating a three-tap channel, but it will directly generalize to longer channel/ISI lengths
  - you see how to do it, right?
- in practice we don’t know the true channel/ISI length, but you just decide how long estimate you wish to obtain and stack the reference symbol values into the convolution matrix $A$ accordingly
- composing the matrix $A$ assumes that we know the transmit symbols over the processing block
  - this is why they are called reference symbols or a reference sequence (also called training sequence or pilot sequence; just alternative terms for the same thing)
  - i.e., a sequence that is known in advance by the receiver and transmitted by the transmitter
- in practice, also e.g. recursive least-squares (RLS) can be used to compute the above solution
  - recursive computations where computing effort per sample or symbol duration is relatively low
  - avoids the need for matrix inversion also
  - search the literature for ‘RLS’, for more details

Channel estimation aspects, shortly, cont’d

For the very same linear system model of the form

$$R_k = A_k * p_k + Z_k$$

the corresponding minimum MSE estimator for the channel impulse response (vector $p$) for given reference symbol samples, reads

$$\hat{p}_{MSE} = (A^H A + \sigma_p^2 R_p^{-1})^{-1} A^H r$$

where the vector $p$ is treated as a random vector, and

- $\sigma_p^2 = E[Z_k^2]$ denotes the noise power which is in the above equation strictly-speaking assumed to be white noise
- $A$ denotes the convolution matrix obtained by proper stacking of the known reference symbol samples, as in the previous pages
- $R_p = E[pp^H]$ denotes the correlation matrix of the discrete-time baseband equivalent channel taps
  - this means that we assume the channel power delay profile to be known, yielding the diagonal entries of the above matrix
  - off-diagonal entries of $R_p$ commonly assumed zero (uncorrelated scattering assumption)
- $r$ denotes again the true observed signal samples at receiver, over the known reference sequence, stacked into a column vector (as in the previous pages)

By definition of the MSE minimization, the above estimator minimizes

$$E[\|p - \hat{p}\|^2]$$

Notice the large similarity to the previous Least-Squares model fitting based channel estimator, reproduced below as (these two become identical for zero noise power):

$$\hat{p}_{LS} = (A^H A)^{-1} A^H r$$
Channel estimation aspects, shortly, cont’d

For the finite length MSE equalizer, formulated earlier as

\[ c_{\text{MSE}} = (\Phi^*)^{-1} \alpha \]

where \( \alpha = E[A_k r_k^*] \) and \( \Phi = E[r_k r_k^H] \), the known reference symbols can also be used to directly estimate \( \alpha \), without explicit channel estimation, as

\[ \hat{\alpha} = \sum_{k=1}^{M} A_k r_k^* \]

where \( M \) refers again to the length of the known reference sequence.

Furthermore, the matrix \( \Phi \) can be estimated over the same window of received samples directly as

\[ \hat{\Phi} = \sum_{k=1}^{M} r_k r_k^H \]

These estimates can then be directly used to compute \( c_{\text{MSE}} = (\Phi^*)^{-1} \alpha \) without explicit channel estimation.

As an alternative approach, since the relevant correlation quantities (\( \alpha = E[A_k r_k^*] \) and \( \Phi = E[r_k r_k^H] \)) depend on the channel impulse response \( p = [p_{-1}, p_0, p_1]^T \) (see classroom exercises for examples), the correlation quantities can also be calculated through an explicit channel estimation stage, if so wanted

- however, as shown in the classroom exercises, \( \alpha = E[A_k r_k^*] \) and \( \Phi = E[r_k r_k^H] \) depend also on the symbol power and noise power, hence using direct sample estimates of \( \alpha \) and \( \Phi \) is more straightforward in this sense
- ... that is, with explicit channel estimation based approach, you need to separately estimate also then e.g. the noise power as well
12. ADAPTIVE FILTERING METHODS FOR CHANNEL EQUALIZATION AND CHANNEL ESTIMATION

Next we address how the so-called adaptive or self-tuning filtering methods can be utilized in channel equalization (shown above, assuming DFE structure for generality) and/or in channel estimation.

The equalization structures themselves are those that we have already been learning – the linear equalizer or the decision feedback equalizer.

The term “adaptive filtering” refers to an idea that the equalizer or the MLSD coefficients are updated iteratively, sample by sample, building on

• known reference symbols (training data), or
• symbol decisions (this naturally works only if the coefficients are already fairly well known; facilitates e.g. tracking slow changes over time)

Most of the common adaptive learning rules build on the instantaneous error signal, which in the equalization context means the difference between the current equalizer output (calculated with the current equalizer coefficients) and the corresponding transmitted symbol

• intuition: good coefficients => error signal values are small
Adaptive filtering based equalization and estimation, cont'd

Depending on how the instantaneous error signal is calculated, the adaptive filtering based solutions have two different operation modes:

- **Known reference symbols based training mode**: In this approach, the transmitted symbols are all known, for a specific training window, and thus the error signal values are calculated against these known symbol values.

  This kind of training sequences exist in all practical communication systems, especially in mobile communication systems, for example once per radio frame or radio sub-frame.

  This way, the receiver can reliably recalculate or adapt its equalizer coefficients or channel estimate in regular intervals.

- **Decision directed mode**: In this approach, the made data symbol decisions are used as the reference, i.e., the error signal values are calculated against these symbol decisions.

  This principle works only if the decisions are fairly reliable, meaning that the equalizer coefficients are already fairly good.

  The adaptation rules incorporate always, explicitly or implicitly, some averaging, so the symbol error probability can in practice be up to 1-10% and still the update works reliably.

  In practice, the decision directed mode can be used to follow or track slow changes in the channel/ISI characteristics, e.g., in mobile communication systems during the data transmission.

In general, system specifications and standards always specify what kind of known training symbol structures are available.

General equalizer structure, revisited

The above shows the general structure (DFE), out of which linear equalizer (LE) is obtained as a special case.

The figure also shows that the error signal is calculated using the symbol decision but like explained on the previous slide, also the known training symbols can be used.

The error signal, assuming that the symbol decisions are correct ($\hat{A}_k = A_k$) or that a known training signal is available, reads then

$$E_k = A_k - Q_k$$

As earlier, we assume FIR filter based equalizers and the following partially anti-causal filter structures for notational convenience:

LE:  
$$C(z) = \sum_{m=-N_1}^{N_2} c_m z^{-m}$$

DFE:  
$$C(z) = \sum_{m=-(N-1)}^{0} c_m z^{-m}, \quad D(z) = \sum_{m=1}^{M} d_m z^{-m}$$
Mean-Squared Error, MSE, and basic notations - recap

We focus on the MSE minimization based adaptive filters in the continuation and thus explicitly define MSE as

\[
J_{MSE} = E[|E_k|^2] = E[|A_k - Q_k|^2]
\]

- target: find coefficients of \( C(z) \), or \( C(z) \) and \( D(z) \), such that \( J_{MSE} \) is minimized

In case of linear equalizer, the equalizer output reads

\[
Q_k = R_k * c_k = \sum_{m=-N_1}^{N_2} c_m R_{k-m} = c_{-N_1} R_{k+N_1} + \ldots + c_{-1} R_{k+1} + c_0 R_k + c_1 R_{k-1} + \ldots + c_{N_2} R_{k-N_2}
\]

where \( c_{-N_1}, \ldots, c_{N_2} \) denote the equalizer coefficients.

As earlier, we write this as

\[
Q_k = c^T r_k
\]

where \( r_k = [R_{k+N_1}, \ldots, R_k, \ldots, R_{k-N_2}]^T \) and \( c = [c_{-N_1}, \ldots, c_0, \ldots, c_{N_2}]^T \).

In case of decision feedback equalizer, the output sample reads

\[
Q_k = R_k * c_k - \hat{A}_k * d_k = \sum_{m=-(N-1)}^{0} c_m R_{k-m} - \sum_{m=1}^{M} d_m \hat{A}_{k-m}
\]

\[
= c^T r_k - d^T \hat{a}_k
\]

where \( r_k = [R_{k+N-1}, \ldots, R_0]^T \), \( c = [c_{-N+1}, \ldots, c_0]^T \), \( \hat{a}_k = [\hat{A}_{k-1}, \ldots, \hat{A}_{k-M}]^T \) and \( d = [d_1, \ldots, d_M]^T \).

Next, we address the adaptive linear equalizer in more details.

As described on the previous slide, the equalizer output sample at time \( k \) reads

\[
Q_k = c^T r_k
\]

where \( r_k = [R_{k+N_1}, \ldots, R_k, \ldots, R_{k-N_2}]^T \) (equalizer input samples) and \( c = [c_{-N_1}, \ldots, c_0, \ldots, c_{N_2}]^T \) (equalizer weights).

As shown already earlier, the optimum reference solution minimizing \( J_{MSE} = E[|E_k|^2] = E[|A_k - Q_k|^2] = E[|A_k - c^T r_k|^2] \) reads

\[
c_{MSE} = (\Phi^*)^{-1}\alpha
\]

where

\[
\sigma_A^2 = E[|A_k|^2] \text{ denotes the symbol power}
\]

\[
\alpha = E[ A_k r_k^*] \text{ denotes the cross-correlation vector between the transmit symbol and the received samples}
\]

\[
\Phi = E[ r_k r_k^H] \text{ denotes the autocorrelation matrix of the received samples}
\]
Iterative MSE minimization through gradient descent

The previous page reference solution establishes that the optimum MSE equalizer weights are obtained by solving a set of linear equations of the form $\Phi^T \mathbf{c} = \alpha$.

Especially for large equalizer lengths, this may become computational demanding
- amount of equations = length of the equalizer

Thus, in the following, we first establish an iterative computing-friendly way to solve the set of linear equations.

Idea: Initialize the equalizer weights $\mathbf{c}$ with some initial trial solution $\mathbf{c}_0$. Then calculate the MSE surface gradient at this point. If the gradient is not zero, take a small step towards the negative gradient (why?).

This is then iterated. In each iteration, the amount of the weight update is controlled by the chosen step-size $\beta > 0$ and we can write the coefficient update from iteration $j$ to $j+1$ as

$$
\mathbf{c}_{j+1} = \mathbf{c}_j - \beta \frac{\partial J_{MSE}}{\partial \mathbf{c}} = \mathbf{c}_j - \beta (-2\alpha + 2\Phi^T \mathbf{c}_j)
$$

$$
= (I - \beta \Phi^T) \mathbf{c}_j + \beta \alpha
$$

This iterative MSE minimization is also commonly called a method of steepest descent (since we always take step towards the negative gradient).

The updated coefficient vector $\mathbf{c}_{j+1}$ is getting closer to the optimum coefficients $\mathbf{c}_{MSE}$, assuming that the step-size is not too large.

Below is an example with two equalizer coefficients, and the ellipses show different contours of the MSE surface (previous page figure viewed from the top down).

The iterative gradient descent algorithm thus solves the set of linear equations $\Phi^T \mathbf{c} = \alpha$ without a matrix inversion. This is very computing friendly, especially when the equalizer is long.

Notice, however, that this solution is not yet an actual data dependent adaptive filtering algorithm – it’s only an alternative way to solve for the MSE optimum coefficients
- but it serves as a basis when we, soon, derive the actual least-mean square (LMS) adaptive filtering algorithm

Before that, we address the stability and convergence of the iterative gradient search in a bit more detailed manner, in particular from the step size selection point of view.
Stability analysis of the gradient descent algorithm

Thus we established on the previous pages that the MSE optimum coefficients $c_{MSE} = (\Phi^*)^{-1} \alpha$ can in practice be obtained iteratively using the gradient descent algorithm of the form

$$c_{j+1} = (I - \beta \Phi^*) c_j + \beta \alpha$$

Now, an interesting question is: When the above algorithm is stable? And under what conditions it really converges towards the optimum solution $c_{MSE} = (\Phi^*)^{-1} \alpha$?

To address this, we first define coefficient error vector relative to the optimum solution $c_{MSE} = (\Phi^*)^{-1} \alpha$ as

$$\varepsilon_j = c_j - c_{MSE} = c_j - (\Phi^*)^{-1} \alpha$$

Combining this with the previous iterative coefficient update, it is easy to show that the coefficient error vector obeys

$$\varepsilon_{j+1} = (I - \beta \Phi^*) \varepsilon_j$$

$\bullet$ thus the question is: under what conditions $\varepsilon_j \to 0$?

Since in general $(I - \beta \Phi^*)$ is not a diagonal matrix, the error value of an individual coefficient at iteration $j + 1$ depends on all the coefficient errors at the previous iteration $j$

$\bullet$ thus, addressing the stability and convergence directly is difficult

To solve this problem, we write the conjugate of the autocorrelation matrix $\Phi^*$ using its eigenvalue decomposition as

$$\Phi^* = U \Lambda U^H$$

where $\Lambda$ is a diagonal matrix containing the eigenvalues of $\Phi^*$ while the matrix $U$ contains the corresponding eigenvectors

$\bullet$ in principle, this is again just applied math – no stress

Mathematicians have also shown that the eigenvalue decomposition can always be calculated such that the eigenvectors (columns of $U$) are orthogonal, which implies that $U^H U = I$ (such matrix is called unitary)

Now, using the decomposition $\Phi^* = U \Lambda U^H$ we can write

$$\varepsilon_{j+1} = (I - \beta U \Lambda U^H) \varepsilon_j$$

which by multiplying both sides from left with $U^H$ reduces to

$$U^H \varepsilon_{j+1} = (U^H - \beta U^H U \Lambda U^H) \varepsilon_j$$

$$= (I - \beta \Lambda) U^H \varepsilon_j$$

If we now also denote $U^H \varepsilon_j = \tilde{\varepsilon}_j$, we obtain

$$\tilde{\varepsilon}_{j+1} = (I - \beta \Lambda) \tilde{\varepsilon}_j$$

Now, since $(I - \beta \Lambda)$ is diagonal, the stability of this recursion is easy to determine

$\bullet$ individual entry of $\tilde{\varepsilon}$ at iteration $j + 1$ depends only on the corresponding component at previous iteration

Furthermore, since $U^H \varepsilon_j = \tilde{\varepsilon}_j$ is an invertible linear transformation, $\tilde{\varepsilon}_j \to 0$ implies stability for the true error vector $\varepsilon_j$.

Based on above, the recursion for the $n$-th entry of $\tilde{\varepsilon}$, denoted by $\tilde{\varepsilon}^{(n)}$, reads

$$\tilde{\varepsilon}^{(n)}_{j+1} = (1 - \beta \lambda_n) \tilde{\varepsilon}^{(n)}_j$$

Such first-order scalar recursion is stable, i.e., $\tilde{\varepsilon}^{(n)} \to 0$ if and only if

$$-1 < (1 - \beta \lambda_n) < 1$$
Stability analysis of the gradient descent algorithm, cont’d

Mathematicians have also shown that the eigenvalues of correlation matrices are real-valued and non-negative.

Furthermore, the step-size $\beta$ is, by definition, a positive-valued quantity, thus we can write the stability criterion as

$$-1 < (1 - \beta \lambda_n) \quad \text{which implies} \quad \beta < \frac{2}{\lambda_n}$$

Thus finally, in order for all the coefficients to converge, we can write the final stability criterion as

$$\beta < \frac{2}{\lambda_{\text{MAX}}}$$

where $\lambda_{\text{MAX}}$ denotes the largest eigenvalue of $\Phi^*$. Knowing the largest eigenvalue of $\Phi^*$ may be difficult in practice, hence a first approximation that people commonly do is $\lambda_{\text{MAX}} \leq \sum_k \lambda_k$ (since all eigenvalues are non-negative).

On the other hand, a known result from math is that the sum of the eigenvalues is always the same as the sum of diagonal values of the original matrix $\Phi$ (called matrix trace)

- but for correlation matrices, these diagonal values describe the power of the equalizer input samples, by definition

Thus, a practical design guideline can finally be expressed as

$$\beta < \frac{2}{P_{\text{IN}}}$$

where $P_{\text{IN}}$ is the sum-power of the equalizer input samples. This is an easily applicable stability criterion in practice!

Convergence rate and further notes about eigenvalues

The fact that the eigenvalues of $\Phi^*$ have an impact on the stability of the iterative gradient descent algorithm is not eventually very surprising.

This is because we are anyway solving the set of linear equations $\Phi^* \mathbf{c} = \mathbf{\alpha}$, and on the other hand already the basic math courses have established that the numerical challenges in solving any set of linear equations $A \mathbf{x} = \mathbf{y}$ depend largely on the condition number of the matrix $A$ (or $\Phi^*$ in our application).

This condition number describes how close to being singular the matrix $A$ is, which is strictly-speaking quantified through the singular value spread which is the ratio of the largest and smallest singular value of $A$.

However, since $\Phi$ is a correlation matrix and thus hermitian symmetric ($\Phi^H = \Phi$), its singular values are the same as the eigenvalues (these are again known results from linear algebra/matrix calculus).

This then directly implies that if the eigenvaluespread of $\Phi$ is large, solving the set of linear equations $\Phi^* \mathbf{c} = \mathbf{\alpha}$ is challenging – no matter what specific solving technique you pursue

- in the case of the iterative gradient descent algorithm, this means that large(r) amount of iterations is needed when eigenvalue spread is large
- when studied further, it can be shown that in such cases with large eigenvalue spread (left figure below), the contours of the MSE surface are strongly elliptic instead of being circles
Example: Gradient descent and converge rate

Below are examples with a 15-tap equalizer \((N_1 = N_2 = 7)\) showing the norm of the coefficient error vector over the iteration index with equalizer input signal having an eigenvalue spread of (i) \(\sim 3.0\) and (ii) \(\sim 23.0\), and with two different step-sizes.

In all cases, the initialization is with \(c_0 = 0\) and the step-size \(\beta \ll 2 / \lambda_{\text{MAX}}\).

The previous gradient descent algorithm is not an adaptive data dependent filtering algorithm but just an iterative solver for MSE optimum coefficients, still relying on the availability of the correlation quantities \(\Phi = E [r_k r_k^H]\) and \(\alpha = E [A_k r_k^*]\).

The following least-mean square, LMS, algorithm uses the received data samples and the known reference symbols to relax these assumptions, and thus provides a true adaptive filtering solution.

The idea is to use instantaneous sample estimates for \(\hat{\Phi}\) and \(\hat{\alpha}\) as:

\[
\hat{\Phi} = r_k r_k^H \quad \text{and} \quad \hat{\alpha} = A_k r_k^*
\]

The corresponding instantaneous estimate of the MSE surface gradient

\[\nabla_c J_{\text{MSE}} = -2\alpha + 2\Phi^* c = -2E [E_k r_k^*]\]

is then

\[\hat{\nabla}_c J_{\text{MSE}} = -2\hat{\alpha} + 2\hat{\Phi}^* c_k = -2E_k r_k^*\]

Following the same ideology as in the previous gradient descent algorithm, but using the instantaneous gradient estimate yields then

\[
c_{k+1} = c_k - \frac{\beta}{2} \hat{\nabla}_c J_{\text{MSE}}
= c_k + \beta E_k r_k^* \quad \text{where} \quad E_k = A_k - c_k^T r_k
\]

- this is the LMS algorithm
- processing is very simple and straight-forward!
- notice that \(E_k = A_k - c_k^T r_k\) corresponds to the instantaneous equalization error with the current equalizer coefficients \(c_k\).
**LMS algorithm, implementation**

\[
Q_k = c_k^T r_k \quad \text{equalizer output at symbol instant } k
\]
\[
E_k = A_k - Q_k \quad \text{instantaneous error at symbol instant } k
\]
\[
c_{k+1} = c_k + \beta E_k r_k^* \quad \text{update of the equalizer coefficients}
\]

Based on above, the \(l\)-th coefficient \(c_{kl}\) is updated as:

\[
[c_{k+1}]_l = [c_k]_l + \beta E_k r_{k-1}
\]

Drawn as a block-diagram it corresponds to the following (only the update and processing of tap \([c_k]_l\) shown explicitly):

**LMS algorithm, stability and convergence**

Compared to the pure iterative gradient descent, LMS algorithm uses a very noisy data-dependent gradient estimate.

This means that the coefficient vector \(c_k\) is actually a random vector since received data is random

- sounds scary?

Then the question is: how to even try to analyze the stability of the random vector recursion of the form

\[
c_{k+1} = c_k + \beta E_k r_k^*
\]

and especially its convergence towards a deterministic optimum solution \(c_{\text{MSE}} = (\Phi^*)^{-1} \alpha\)?

One option is to consider how the recursion behaves in terms of the statistical average/mean, that is, whether the following recursion is well-behaving:

\[
E[c_{k+1}] = E[c_k] + \beta E[E_k r_k^*]
\]

By definition, the above behaves identical to the pure iterative gradient descent (why?), and thus the convergence of the mean of the LMS algorithm behaves accordingly

- previous stability bounds, \(\beta < 2 / \lambda_{\text{MAX}}\) and \(\beta < 2 / P_{IN}\) apply directly!

Notice, on the other hand, that this is very weak form of convergence – it considers only the mean behavior.
LMS algorithm, stability and convergence, cont’d

An alternative more ambitious approach is to consider the behavior of the MSE, $E[|E_k|^2]$, over the LMS-based data dependent learning.

If pursuing this path, with a little effort it can be shown that the following holds (details in the course book)

$$E[|E_k|^2] = J_{\text{min}} + E[e_k^H \Phi^* e_k]$$

where $J_{\text{min}} = E[|A_k|^2] - \alpha^H c_{\text{MSE}}$ is the MSE value corresponding to the fixed optimum solution $c_{\text{MSE}} = (\Phi^*)^{-1} \alpha$ and $e_k = c_k - c_{\text{MSE}}$.

- the second term is the so-called excess MSE, which depend on the LMS algorithms capability to control and learn the coefficients

Exact analysis of the excess MSE tends to be tedious

- it is, however, important to acknowledge that the MSE level that LMS can reach at the equalizer output is higher compared to the reference optimum MSE solution
- this is natural and intuitive since LMS relies heavily on data-dependent coarse gradient estimates
- we’ll illustrate with a simple example (next page), more complete treatment available in the course book and in general in adaptive filtering literature

Suppose we consider a very simple example where the equalizer input samples are uncorrelated and zero mean (this cannot really hold, in reality, due to ISI). In such special case, the autocorrelation matrix of the equalizer input samples is diagonal, of the form

$$\Phi^* = \phi_0 I, \quad \text{where} \quad \phi_0 = E[|R_k|^2]$$

All the corresponding eigenvalues are $\lambda_1 = \lambda_2 = \ldots = \lambda_N = \phi_0$. Thus, for convergence of the mean ($E[c_k]$), the stability criterion is $\beta < 2/\phi_0$.

Mean-squared error reads now

$$E[|E_k|^2] = J_{\text{min}} + \phi_0 E[\|e_k\|^2]$$

It can then be shown that $E[\|e_k\|^2]$ behaves as:

$$E[\|e_{k+1}\|^2] = \gamma E[\|e_k\|^2] + \beta^2 N \phi_0 J_{\text{min}}$$

where $\gamma = 1 - 2\beta \phi_0 + N \beta^2 \phi_0^2$ and for notational simplicity $N$ denotes the equalizer length.

From the MSE perspective, the stability criterion is now $|\gamma| < 1$ (why?) which can be solved for

$$\beta < \frac{2}{N \phi_0} = \frac{2}{N \lambda_{\text{MAX}}}$$

Notice that this is more strict step-size bound compared to $\beta < 2/\phi_0$.

Furthermore, for the excess MSE it holds that

$$\phi_0 E[\|e_k\|^2] \to \frac{N \phi_0}{2 / \beta - N \phi_0} J_{\text{min}} \quad \text{when} \quad k \to \infty$$
LMS algorithm, stability and convergence, summary

Because of the noisy data-dependent gradient estimate, even when converging, the LMS algorithm contains always residual fluctuations around the optimum MSE coefficients

- Stability and convergence of the LMS algorithm requires commonly a step-size that is clearly below the maximum step-size limit of the pure gradient descent
  - common working rule is to use 10-100 smaller step size

- In the steady-state, the amount of fluctuations around the ideal MSE coefficients is always relative to the step-size $\beta$ (see the previous page example expressions)

- If the purpose is to minimize the excess MSE in the steady-state, the step-size should be chosen basically as small as possible

- On the other hand, the convergence speed and ability to track changes also depend on the step-size. If the step-size is very small, the convergence is very slow.

Thus, if the purpose is to also track time-varying changes in the channel characteristics, the step-size selection is always a compromise between the following:

- convergence rate and tracking capability
- steady-state excess MSE

Furthermore, the sensitivity to the eigenvalue spread of the equalizer input samples’ covariance matrix is unavoidably inherited from the pure gradient descent algorithm

- this is because LMS is nothing but data dependent approximation of the iterative gradient descent

Example: LMS algorithm convergence and steady-state fluctuations

Below are some examples of the LMS algorithm based coefficient error vector norms over time, against the MSE optimum filter, with a simple 5-tap equalizer and with eigenvalue spreads of the equalizer input signal being (i) $\sim 4$ and (ii) $\sim 11$. Also the step-size is varied.

In all cases, the equalizer is initialized as $c_0 = 0$ and $\beta << 2/\lambda_{\text{MAX}}$. 
Some modifications of the basic LMS algorithm

**Normalized LMS, NLMS**

In the basic LMS, the strength of the equalizer input signal impacts directly the convergence and stability (see the previous results).

Thus, in practice, it is sensible to normalize the step-size $\beta$ relative to the input signal variance ($\sigma_R^2$) as

$$
\beta = \frac{a}{\sigma_R^2 + b}
$$

where $a$ and $b$ are constants. Also the input signal variance can be calculated recursively as (here $\mu$ is an averaging parameter)

$$
\sigma^2_{R,k} = \mu\sigma^2_{R,k-1} + (1 - \mu)|R_k|^2
$$

to allow for tracking possible changes of the input signal power in the same learning loop.

**Gear-Shift approach**

Here the basic idea is to use a larger step-size in the beginning such that convergence is fast while then reduce the step-size such that steady-state MSE is small.

It takes engineering insight to adapt this principle to practical problems, and it largely depends on how fast changes are assumed to be tracked over time.

In general, adaptive filtering literature comprises various other adaptive learning algorithms as well.

**ADAPTIVE DFE**

The overall DFE block-diagram, with FIR filters as the forward filter and feedback filter, reads (recap):

Now, as discussed already earlier, the equalizer output sample $Q_k$ and the instantaneous equalization error $E_k$ read:

$$
Q_k = \sum_{i=-(N-1)}^{0} c_i R_{k-i} - \sum_{i=1}^{M} d_i \hat{A}_{k-i} = c^T r_k - d^T \hat{a}_k
$$

$$
E_k = \hat{A}_k - Q_k \quad \text{(decision directed mode)}
$$

$$
E_k = \hat{A}_k - Q_k \quad \text{(reference signal based mode)}
$$

where $r_k = [R_{k+N-1}, \ldots, R_0]^T$, $c = [c_{-N+1}, \ldots, c_0]^T$,

$\hat{a}_k = [\hat{A}_{k-1}, \ldots, \hat{A}_{k-M}]^T$ and $d = [d_1, \ldots, d_M]^T$. 
Fixed MSE optimum reference solution (recap)

Assuming that $\hat{A}_k = A_k$ (decisions are correct, or there is training data available) we can first write:

\[
Q_k = c^T r_k - d^T a_k = w^T u_k
\]

where the stacked vectors read:

\[
w = [c_{(N-1)}, \ldots, c_0, -d_1, \ldots, -d_M]^T
\]

\[
u_k = [R_{k+(N-1)}, \ldots, R_k, A_{k-1}, \ldots, A_{k-M}]^T
\]

Then the error signal \( E_k = A_k - Q_k = A_k - w^T u_k \) and thus the MSE reads:

\[
J_{MSE} = E \left[ |E_k|^2 \right] = E \left[ |A_k - Q_k|^2 \right] = E \left[ |A_k - w^T u_k|^2 \right]
\]

Now, as already discussed earlier, we can obtain the MSE optimum reference solution by calculating the gradient and setting it to zero:

\[
\frac{\partial J_{MSE}}{\partial w} = 0 \Rightarrow w_{MSE} = (\Phi^*)^{-1} \tilde{\alpha}
\]

where now:

\[
\tilde{\alpha} = E \left[ A_k u_k^* \right]
\]

\[
\Phi = E \left[ u_k u_k^H \right]
\]

The corresponding forward filter and feedback filter coefficients are obtained from \( w_{MSE} \) through the stacked structure:

\[
w = [c_{(N-1)}, \ldots, c_0, -d_1, \ldots, -d_M]^T
\]

LMS algorithm for DFE

Using the previous notations of the form:

\[
Q_k = w^T u_k
\]

\[
E_k = A_k - Q_k = A_k - w^T u_k
\]

\[
w = [c_{(N-1)}, \ldots, c_0, -d_1, \ldots, -d_M]^T
\]

\[
u_k = [R_{k+(N-1)}, \ldots, R_k, A_{k-1}, \ldots, A_{k-M}]^T
\]

and following identical steps as in the linear equalizer case, the LMS algorithm can be expressed for the DFE as:

\[
Q_k = w_k^T u_k
\]

\[
E_k = A_k - Q_k = A_k - w^T u_k
\]

\[
w_{k+1} = w_k + \beta E_k u_k^E
\]

If no training data is available, the symbol variables are substituted with the corresponding decisions $\hat{A}_k, \hat{A}_{k-1}, \ldots$

- this applies to both error signal calculation as well to the samples processed by the feedback filter.
ADAPTIVE FILTER BASED CHANNEL ESTIMATION

Next we shortly address how LMS-based adaptive filtering principles can also be used for channel/ISI profile estimation:

- channel estimate used then, e.g., in MLSD/Viterbi processing
- or in computing, e.g., a zero forcing (ZF) linear equalizer or ZF decision feedback equalizer (MSE based ones can be adapted directly, with LMS, like we have just learned)

The principle of adaptive filtering based channel estimation is conceptually illustrated below:

The idea is now to feed in the training data sequence into an FIR filter $C(z)$ whose coefficients are adapted so that the difference between the filter output and the true observed signal is minimized:

- this is intuitively plausible since when $C(z)$ is close to $P(z)$ (the true channel/ISI profile), the output of $C(z)$ and the true observed signal $R_k$ are very similar (error values are small)
- the filter $C(z)$ will then form the channel estimate

Instead of the training signal based mode, also the decision directed mode can be adopted in order to adapt to (slow) changes in the channel response.

LMS-based channel estimation

With LMS based processing, for an $N_1+N_2+1$-tap channel estimator $C(z)$, the processing can be described as:

- estimator output at symbol instant $k$: $S_k = c_k^T a_k$
- instantaneous error at symbol instant $k$: $E_k = R_k - S_k$
- update of the estimator coefficients: $c_{k+1} = c_k + \beta E_k a_k^*$

where $a_k = [A_{k+N_1}, \ldots, A_k, \ldots, A_{k-N_2}]^T$, $c_k$ denotes the channel estimator coefficients at symbol instant $k$, and $\beta$ is again the adaptation step-size.

Notice that:

- here the estimator filter processes directly the known reference symbols and compares the output $S_k = c_k^T a_k$ to the actual observed received signal sample $R_k$ to form an error signal $E_k = R_k - S_k$, which is used in adapting the estimator coefficients
- in the previous adaptive channel equalization case, in turn, the equalizer filter processes the received signal samples and compares the output $c_k^T r_k$ to the known reference signal sample $A_k$, forms an error signal $E_k = A_k - Q_k$ and updates the equalizer coefficients
- both the adaptive channel estimator, described above, and the adaptive update of the equalizer coefficients, described earlier, can also adopt the decision directed learning mode:
  - adaptive equalizer; error is calculated against the symbol decision as $E_k = \hat{A}_k - Q_k$ (instead of known symbol $A_k$)
  - adaptive channel estimator; estimator output $S_k = c_k^T a_k$ is calculated using the symbol decisions $a_k = [\hat{A}_{k+N_1}, \ldots, \hat{A}_k, \ldots, \hat{A}_{k-N_2}]^T$ (instead of known symbols)
  - in causal implementation, there is naturally an appropriated delay imposed in the processing – as always
LMS-based channel estimation, cont’d

We’ll again continue our earlier example from page 322 (and onwards), and use now LMS-based channel estimation, with received SNR of 15 dB, 500 reference symbols and two different FIR estimator lengths

- The results are illustrated below in terms of frequency responses

![Discrete-Time Effective Channel Responses](image1)

-5 -4 -3 -2 -1 0 1 2 3 4 5
Frequency in MHz

-20 -15 -10 -5 0 5 10 15 20
Power Response in dB

Discrete-Time Effective Channel Responses

-5 -4 -3 -2 -1 0 1 2 3 4 5
Frequency in MHz

-20 -15 -10 -5 0 5 10 15 20
Power Response in dB

The true channel impulse response length (see the pulses on page 323) is approximately 7 or 8 taps (others are very close to zero), so clearly with 5 tap estimator you cannot exactly reproduce an accurate response estimate yet.

With 15 tap estimator, there is then basically already some “overfitting” or over-parametrization in the estimator, but the LMS algorithm will automatically tune the unnecessary taps to zero.

You can easily reproduce the experiment, and explore also the impulse responses (true channel vs. the channel estimate).
13. ERROR CONTROL CODING

Error control coding methods are very central ingredients in every digital communication system. In general, different levels or aspects of error control can be categorized as:
- Error detection
- Error correction
- Error prevention

These will be addressed in details in this part of the course.

By using error control methods, we can utilize and communicate bits reliably over challenging transmission media, like mobile radio channel.

Without error control methods, this would typically pose unfeasible requirements in terms of e.g. spectral efficiency and/or energy efficiency. Thus with error control coding, combined with proper modulation, the channel capacity can be deployed better.

Overall, coding theory is a very wide area of applied mathematics. Here we will take again an “engineering approach” and look things mostly from communications/radio system engineer perspective.

Contents in this part:
- Basics of error control coding, redundancy and different levels of error control
- Hard and soft decoding principles
- Measuring coding gain
- Block codes, convolutional codes and trellis codes
- Retransmission principles, ARQ & HARQ, shortly

Then also more recent and advanced coding methods like Turbo codes, LDPC codes and polar codes will be shortly introduced.

---

Error Control Methods, at large

On the transmitter side, error control coding, or channel coding, is typically purely bit-level signal processing.

In all error control methods, the idea is to create certain known “structure” or redundancy to the transmitted bit stream and thereon to the transmitted symbol stream:
- Adding new (redundant) bits to the information bit stream. Then, if the modulation does not depend on channel coding, this means that also the amount of transmitted symbols increases (redundant symbols). This is the classical approach.
- As an alternative, channel coding and modulation can be designed jointly, and the redundancy can be built in to the transmitted signal without increasing the amount of symbols. This is called trellis coding.

On the receiver side, this known structure (redundancy) can then be deployed for error control in data decoding stage, in the following ways:
- **Error Detection**: The decoder can tell if the coded received block contains certain amount of errors, but cannot tell which bits are erroneous.
  
  If errors are detected, then typically a retransmission of the corresponding block is requested (ARQ, HARQ).
- **Error Correction**: Here the detected errors are also corrected on the receiver side, i.e., the decoder can tell which bits in the block are in error.
- **Error Prevention**: This refers to receiver methods where we try to avoid making any bit errors in the first place (instead of first making errors and then correcting). Directly tied to soft decoding.
Hard and Soft Decoding on Receiver Side

Hard Decoding based Receiver

Here symbols are first detected without thinking the redundancy at all, and mapped to detected coded bits. Then the channel decoder utilizes the redundancy at bit-level, to detect or correct errors. The link between the encoder and decoder can be modeled as binary symmetric channel. This approach is simple in the sense that hard decoding based error control can be added on top of existing communication system without changing other functionalities (like raw symbol detection) at all.

Soft Decoding Receiver

Here (raw) symbol detection and channel decoding are combined, i.e., soft observations are mapped directly to information bits.

It is easy to see that soft decoding is superior (performance-wise) to hard decoding since hard decoder looses information in raw detection

- One way to think of this is that the essential minimum Euclidian distance in detection/decoding is bigger, and thus error probability smaller, in soft decoding

On the other hand, implementation is more complicated.

Simple example of hard and soft decoding

Let’s consider the simplest possible case that we try to communicate one single information bit \( b \in \{0,1\} \). Suppose the channel coding uses a (3,1) repetition code, which means that we actually transmit the information bit three times in a row. Thus the coded bits, or so called code words, are then

\[
\begin{align*}
b = 0 &\rightarrow c_1 = [0, 0, 0] \\
b = 1 &\rightarrow c_2 = [1, 1, 1]
\end{align*}
\]

Assuming further that binary modulation (symbols \( \{1, -1\} \)) is used, the corresponding symbol blocks or signal vectors are

\[
\begin{align*}
b = 0 &\rightarrow c_1 = [0, 0, 0] \rightarrow s_1 = [-1, -1, -1] \\
b = 1 &\rightarrow c_2 = [1, 1, 1] \rightarrow s_2 = [+1, +1, +1]
\end{align*}
\]

Suppose now that the received noisy/distorted observations are

\[
q = [-0.3, -0.1, +0.6]
\]

**Hard decoder**

- Raw symbol detection \( \Rightarrow -1, -1, +1 \)
- Mapping these to bits \( \Rightarrow 0, 0, 1 \)
- Compare to code words \( \Rightarrow \) closer to \( c_1 = [0, 0, 0] \)
- Thus the decoded information bit is \( \hat{b}_{\text{hard}} = 0 \)

**Soft decoder**

- Compare directly the whole observation to the two possible symbol blocks in terms of Euclidian distance:

\[
\begin{align*}
d_E^2(c_1, q) &= 0.3^2 + 0.1^2 + 0.6^2 = 0.96, \\
d_E^2(c_2, q) &= 0.2^2 + 0.1^2 + 0.3^2 = 0.46
\end{align*}
\]

- Since \( d_E^2(c_1, q) < d_E^2(c_2, q) \), the decoded information bit is \( \hat{b}_{\text{soft}} = 0 \)

**Question:** Which decision is correct?
Common Coding Families

Block Codes
- Coding and redundancy are implemented at block level. $k$ data bits are mapped to $n > k$ coded bits. No memory between consecutive blocks.
- Code rate $r_c = k/n < 1$.
- Both hard and soft decoding are possible on receiver side.

Convolutional Codes
- Also here the encoding is done in a block-wise manner but key difference to block codes is that here we also have memory between the blocks. This enables typically shorter block sizes ($k$ and $n$) compared to classical block codes.
- Code rate $r_c = k/n < 1$.
- Both hard and soft decoding are possible on receiver side.

Trellis codes (coded modulation, signal space codes)
- Here the idea is that the amount of redundancy built in at bit level is taken into account also in the symbol alphabet (modulation) design. This means that the modulation depends on the code rate $r_c = k/n$ (e.g. using 64QAM to transmit 4 information bits per symbol, with rate 4/6 code).
- Baseline code can in principle be a block code or convolutional code.
- Soft decoding is typically deployed.

Common Coding Families, cont’d

Turbo codes
- Two or more interleaved convolutional codes, commonly in parallel
- Interleaver allows for efficient iterative decoding at RX

LDPC (low density parity check) codes
- Special case of block-codes with certain sparse structure for parity check matrix
- Allows for efficient iterative decoding with reasonable complexity

Polar codes
- special case of block codes with capacity achieving properties

Some examples from practical systems
- 3G WCDMA/UMTS and 3.5G HSPA networks
  - convolutional coding is the baseline, many different code rates defined and used
- 4G LTE/LTE-Advanced networks
  - Turbo coding is the baseline, many different code rates defined and used
- emerging 5G/New Radio (NR) networks (currently in early phases of 3GPP standardization)
  - data channels: LDPC coding, many different code rates
  - control channels: Polar code
Coding Gain

The link level gain obtained through coding, can be addressed from two slightly different angles. When compared to uncoded system
- the error rate of information bits is smaller, for given received SNR/SINR
- the target error rate of information bits can be achieved with smaller received SNR/SINR

In terms of quantitative analysis, the latter option is more straightforward. From this perspective, Coding Gain means the difference in the needed received SNR/SINR, between the coded and uncoded systems, such that given target error rate is achieved for the information bits.

Coding Gain (cont’d)

Another important aspect in Coding Gain analysis, and in comparing different codes, is the possible impact of coding into the actual information bit rate and/or bandwidth requirements.

Both block and convolutional codes increase the amount of transmit bits
- \( n/k \) times compared to uncoded system

Thus if we assume that the symbol alphabet does not depend on the code rate, this implies that:

(a) If information bit rate needs to be independent of coding, the symbol rate and thus bandwidth of the coded system are \( n/k \) times bigger than those of the uncoded system

(b) If, in turn, the link bandwidth is fixed independently of what coding and modulation are used, then the information bit rate of the coded link is only \( k/n \) relative to uncoded system (i.e. smaller)

This means that the obtained Coding Gain in case (a) is smaller compared to case (b), simply because receiver noise powers are different (larger bandwidth implies larger noise power):

(a) Coded link symbol rate, and thus bandwidth, and thereon receiver noise power are \( n/k \) times, notated \( \sigma_c^2 = (n/k)\sigma_u^2 \)

This eats part of the error control power of coding!

(b) Coded and uncoded links have identical bandwidths and thereon noise powers, i.e. \( \sigma_c^2 = \sigma_u^2 \)

In summary: Formal Coding Gain is bigger in case (b) but the price to pay is the reduced information bit rate

- This is commonly the case in practice; bandwidth is given (fixed), more redundancy implies better reliability but lower throughput
- Perhaps one can say that viewpoint (a) is more accurate from communication theory point of view but viewpoint (b) is in many sense more practical
Maximum Likelihood decoding and distance metrics

Hard decoding is purely bit-level signal processing

- Based on earlier discussions, and assuming consecutive bit errors statistically independent, the ML decoder can be built around calculation and minimization of the Hamming-distance
  - Hamming distance measures how many bits are different in two binary words (refer to our basic material of ML detector for binary symmetric channel, BSC)

Soft decoder is processing soft received samples

- Under Gaussian noise/interference assumption, the ML decoder builds then on Euclidian distance minimization

Obviously these two metrics are related, depending on the modulation and symbol alphabet.

Example: Two bit words \( c_1 = [1, 0, 0, 0] \) and \( c_2 = [0, 1, 0, 1] \).

- Hamming-distance, i.e., the number of different bits is \( d_H(c_1, c_2) = 3 \).
- Assuming then binary modulation with alphabet \( \{+a, -a\} \), as a simple example, the corresponding symbol words or signal vectors are \( s_1 = [+a, -a, -a, -a] \) and \( s_2 = [-a, +a, -a, +a] \) whose Euclidian distance is \( d_E(s_1, s_2) = 2a\sqrt{3} = 2a\sqrt{d_H(c_1, c_2)} \)

This can easily be seen to hold for any two arbitrary words. Thus under the binary modulation assumptions (with alphabet \( \{+a, -a\} \)), the Hamming distance and Euclidian distance are always related through

\[
d_E = 2a\sqrt{d_H}
\]

We will deploy this result many times in, e.g., coding gain analysis.

BLOCK CODES

Notation: \((n, k)\) block code \(C\)

- Source (data) word length \(k\) bits, code word length \(n\) bits
- Maps source bit stream in blocks of \(k\) bits to coded bit stream with \(n\)-bit code words such that every \(k\)-bit source word has unique \(n\)-bit code word
- E.g. \((7,4)\) code below:

Mapping is memoryless, i.e., current \(n\) coded bits depend only on current \(k\) source bits.

The number of different code words equals the number of different source words, and for \(n > k, \; 2^k < 2^n\)

- Thus the code creates redundancy, i.e., only some \(n\)-bit words are valid code words

More formally, code \(C\) (set of code words) forms a subspace in the overall space of all \(n\)-bit words

- One way to interpret the redundancy and the structure it creates into the signal!

Example: \((3,1)\) repetition code, i.e., repeating every source bit 3 times

\[\begin{align*}
0 & \rightarrow 000 \\
1 & \rightarrow 111
\end{align*}\]
**Basics of Block Codes**

In linear block codes, all code words can be represented in vector-matrix form as

\[ c = bG \]

where \( b = [b_1, b_2, \ldots, b_k] \) denotes a row vector of source bits
\( c = [c_1, c_2, \ldots, c_n] \) denotes a row vector of coded bits and
\( G \) is the so-called generator matrix

All arithmetics are in modulo-2.

This implies directly that
- All-zero word is always one valid code word (why?)
- Modulo-2 sum of any two code words is also always a valid code word (why?)

Also using the previous subspace interpretation, and since

\[ c = bG = b_1g_1 + b_2g_2 + \ldots + b_kg_k = \sum_{i=1}^{k} b_ig_i \]

where \( g_i \) denotes \( i \)-th row of the generator matrix \( G \), the rows of the generator matrix form the base for the code subspace.

**Systematic codes** are a subset of all linear codes in which the source bits appear directly as such at the coder output. For block codes, this implies that the first \( k \) code bits are directly the source bits. The generator matrix has then the following partition

\[ G = [I_k | P] \]

where \( I_k \) denotes a \( k \times k \) identity matrix. It can be shown that all linear block codes can be transformed systematic with certain elementary manipulations to the generator matrix (see classroom exercises).

**Basics of Block Codes** (cont’d)

Like we will see soon in more details, the error control capabilities of hard decoding receivers (and implicitly also soft decoding receivers) are directly dependent on the minimum Hamming distance of the code (set of code words)

\[ d_{H,\text{min}} = \min d_H(c_i, c_j); \ c_i \in C, c_j \neq c_i \in C \]

Another related quantity called **Hamming weight** \( w_H(c_k) \) is defined as the number of bits 1 included in the word \( c_k \).

This, and the linearity of the code, implies then directly that (why?)

\[ d_{H,\text{min}} = \min w_H(c); \ c \in C, c \neq 0 \]

- In English: minimum Hamming distance of the code is the same as the minimum Hamming weight of the code words (excluding the all-zero word which is always one valid code word)

Also, based on earlier derivations, and assuming binary modulation (alphabet \( \{+, -\} \)) the corresponding **minimum Euclidian distance** is given by

\[ d_{E,\text{min}} = 2a \sqrt{d_{H,\text{min}}} \]

This, in turn, will essentially define the performance of soft decoding, as we will show explicitly a little later.
Example 1: elementary parity check codes

Suppose we take a \((3,2)\) parity check code as an example where the idea is to append the source word (of length 2 in this example) with one parity bit:

\[ c_1 = b_1, \quad c_2 = b_2, \quad c_3 = b_1 + b_2 \]

Clearly the code is systematic and the generator matrix is of the form

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

Mapping of all source words to corresponding code words is thus

\[
\begin{align*}
00 & \rightarrow 000 \\
01 & \rightarrow 011 \\
10 & \rightarrow 101 \\
11 & \rightarrow 110
\end{align*}
\]

Clearly the minimum Hamming weight and thus the minimum Hamming distance is 2.

The generator matrix of a more general \((k + 1, k)\) parity check code is

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

Clearly such code is always systematic, and it is easy to show that independent of \(k\), the minimum Hamming weight and thus the minimum Hamming distance is always 2.

Example 2: (7,4) Hamming-code

The generator matrix for (7,4) Hamming-code is (picked directly from coding literature)

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

Clearly also this code is systematic.

The minimum Hamming-distance for this code is 3 (verify yourself).
Maximum Likelihood Decoders and Performance

Like discussed already at preliminary level, the operating principle of ML decoders is the following:

Choose the code word that is closest to the received sequence. Distance metrics are
- Hamming distance in hard decoding
- Euclidian distance in soft decoding (where symbol mapping is also taken into account).

That the above distances and their minimization really maximize the likelihood function is based on the following assumptions (for details, refer to our earlier discussions regarding the basics of ML vector detector in BSC and AWGN channels)

- Independent bit errors at hard decoder input, stemming from errors in raw symbol detection (Binary Symmetric Channel, BSC, model)
- Additive white Gaussian noise/interference (AWGN) model for soft decoder input

From the performance point of view, the key thing is the minimum distance, i.e., the minimum Hamming distance in the set of all code words (hard decoder) or the corresponding minimum Euclidian distance in the set of all signal vectors (soft decoder).

In the following, we will address the performance analysis of both hard and soft decoding in more details

- Hard decoder: error detection and correction capabilities
- Soft decoder: error prevention and Coding Gain

Coding Gain analysis in terms of the needed received SNR/SINR for given target error rate is fairly straight-forward for soft decoder. This gives then also an upper bound for the Coding Gain of hard decoder.

Soft Decoder Performance

The input of soft decoder is a sequence of soft (undetected) received samples from receiver front-end, with redundancy built-in on transmitter side.

For simplicity, we assume again binary modulation case with alphabet \{+a, -a\} and further assume (for analysis purposes) that received soft samples are only corrupted by additive Gaussian noise/interference.

In the following we use the following notation for symbol alphabet and receiver noise/interference variance

- Uncoded system \(+a_u, -a_u\), \(\sigma_u^2\)
- Coded system \(+a_c, -a_c\), \(\sigma_c^2\)

The purpose is then to determine the achievable Coding Gain at SNR/SINR axis through bit error rate (BER) analysis for the information bits. The obtained Coding Gain for soft decoding forms then also an upper bound of Coding Gain in hard decoding.

Like discussed before, depending on the assumptions

\[ \sigma_c^2 = \left(\frac{n}{k}\right)\sigma_u^2 \quad \text{or} \quad \sigma_c^2 = \sigma_u^2 \]

which has direct impact on the Coding Gain.

Starting point in the Coding Gain analysis is the well-known BER formula for binary uncoded system (see our earlier materials)

\[ P[\text{bit error, uncoded system}] = Q\left[\frac{a_u}{\sigma_u}\right] = Q\left[\frac{a_u^2}{\sigma_u^2}\right] = Q\left[\sqrt{\text{SNR}_u}\right] \]
Soft Decoder Performance (cont’d)

In case of soft decoder, the received signal vector is conceptually of the form

\[ q = a + n \]

where \( a \) denotes the symbol/signal vector corresponding to transmitted codeword \( c \) and \( n \) denotes Gaussian noise vector.

Based on earlier developments, ML decoder chooses that signal vector that is closest in Euclidian distance sense to the observation \( q \). Then

\[
(2^k - 1)Q \left[ \frac{d_{E,\text{min}}}{2\sigma_c} \right] \geq P[\text{detection error}] \geq Q \left[ \frac{d_{E,\text{min}}}{2\sigma_c} \right]
\]

where \( d_{E,\text{min}} \) denotes the minimum Euclidian distance of the used code.

Above upper bound is obtained through Union Bound method (assuming all codewords are at minimum distance from each other). The lower bound, in turn, is obtained when there is always only one code word at minimum distance from another word.

The detection and information bit error rates, in turn, are related through

\[
P[\text{detection error}] \geq P[\text{bit error}] \geq \frac{1}{k} P[\text{detection error}]
\]

Thus overall we get

\[
(2^k - 1)Q \left[ \frac{d_{E,\text{min}}}{2\sigma_c} \right] \geq P[\text{bit error, coded system}] \geq \frac{1}{k} Q \left[ \frac{d_{E,\text{min}}}{2\sigma_c} \right]
\]

Example: (7,4) Hamming code

\[
15Q \left[ \frac{d_{E,\text{min}}}{2\sigma_c} \right] \geq P[\text{bit error, coded system}] \geq \frac{1}{4} Q \left[ \frac{d_{E,\text{min}}}{2\sigma_c} \right]
\]
Soft Decoder Performance (cont’d)

Thus from SNR point of view, the Coding Gain in dBs is directly

\[
\text{Coding Gain}_{\#1} = 10 \log_{10}(d_{H,\min}) \ [\text{dB}]
\]

Now if \( \sigma_c^2 = \sigma_u^2 \) (i.e., bandwidth fixed and identical noise powers, coded system has lower information bit rate), this is then also the difference in the needed received signal power.

If, in turn, we want to keep the information bit rate independent of the used code, the coded system needs bigger bandwidth and thus has bigger receiver noise power \( \sigma_c^2 = (n/k)\sigma_u^2 \).

This eats part of the formal coding gain observed at SNR axis alone, and we get

\[
\text{Coding Gain}_{\#2} = 10 \log_{10}\left(\frac{k}{n} d_{H,\min}\right) \ [\text{dB}]
\]

Examples

**Example 1:** Simple (3, 2) parity check code, \( d_{H,\min} = 2 \).

\[
\text{Coding Gain}_{\#1} = 10 \log_{10}(2) \approx 3 \text{ dB}
\]

\[
\text{Coding Gain}_{\#2} = 10 \log_{10}\left(\frac{2}{3}\right) \approx 1.25 \text{ dB}
\]

Notice that even though this code is almost trivial, it can still provide reasonable coding gain when soft decoding is used!!

**Example 2:** (7, 4) Hamming code, \( d_{H,\min} = 3 \).

\[
\text{Coding Gain}_{\#1} = 10 \log_{10}(3) \approx 4.77 \text{ dB}
\]

\[
\text{Coding Gain}_{\#2} = 10 \log_{10}\left(\frac{4}{7}\right) \approx 2.34 \text{ dB}
\]

Again, fairly simple code but good coding gain available!!

These form then also the upper bound of coding gain for hard decoder.
Hard Decoding Performance

Hard decoding is done after raw symbol detection and symbol-to-bit mapping. From decoder point of view, observation can be modeled with a BSC, with individual coded bit error rate $p$.

Error control capabilities depend directly on the minimum Hamming distance $d_{H,\text{min}}$:

- Code can detect maximum of $s \leq d_{H,\text{min}} - 1 \iff d_{H,\text{min}} \geq s + 1$ bit errors in a single received binary word.

  This is because $\max s = d_{H,\text{min}} - 1$ bit errors cannot yet "transform" any valid code word into another valid code word, and thus we can detect at RX that errors have occurred.

- Code can correct maximum of $t \leq \lfloor (d_{H,\text{min}} - 1) / 2 \rfloor \iff d_{H,\text{min}} \geq 2t + 1$ bit errors in a single received bit word.

  This is because $\max t = \lfloor (d_{H,\text{min}} - 1) / 2 \rfloor$ bit errors still keep the received word closer to the original code word than any other valid code word, and we can correct the errors at RX through minimum distance decoding.

  **Example 1:** $d_{H,\text{min}} = 2 \Rightarrow t = 0$

  Code cannot correct any bit errors but it can detect a single bit error.

  **Example 2:** $d_{H,\text{min}} = 3 \Rightarrow t = 1$

  Code can correct all possible individual bit errors, or detect two errors.

Previous requirements guarantee the given error control capability, no matter what is the exact error combination and actual transmitted codeword.

In many cases, the code can, on the other hand, decode correctly some error patterns with more than $t$ errors. This depends on the more detailed structure of the code subspace.

So called **perfect code** is a special case, which can never correct more than $t$ bit errors.

With perfect codes, the code subspace has most "economical" structure (in the space of all possible $n$-bit words) for given $t$. Perfect codes have biggest possible $t$ for given $n$ and $k$.

In general, only a very small part of all codes are perfect.

**Example:** (7,4) Hamming code

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

This code is perfect since all 7-bit words are either valid code words or at distance 1 from one other code word.
Hard Decoding Performance (cont’d)

Assuming the hard decoder input follows a BSC model, the probability for having \( m \) errors in a decoding block of \( n \) bits follows binomial distribution (why? .. well, see again our earlier materials regarding ML detector and BSC):

\[
P(m, n) = \binom{n}{m} p^m (1-p)^{n-m}
\]

In above, \( p \) denotes the individual bit error (flip) probability at decoder input.

Then decoder error probability (i.e., the probability that we are closer to another word than the original) is given by

\[
P \left( \text{detection error} \right) = 1 - \sum_{m=t+1}^{n} P(m, n) = 1 - \sum_{m=0}^{t} P(m, n)
\]

Strictly speaking this holds only for perfect codes, and is thus an upper bound for other codes.

Many practical codes are quasi-perfect, which means:

\[
\sum_{m=t+2}^{n} P(m, n) \leq P \left( \text{detection error} \right) \leq \sum_{m=t+1}^{n} P(m, n)
\]

In literature, you can find also many alternative bounds which may very well be more tight than the above.

A general rule of thumb is that with most practical codes, the Coding Gain in hard decoding is commonly 1-2 dB lower than in corresponding soft decoding.

Example: (7,4) Hamming code (\( t = 1 \)) and suppose \( p = 0.01 \) in the coded system

\[
P \left[ \text{detection error} \right] = 1 - (1-p)^7 - 7p(1-p)^6
\]

\[p = 0.01 \Rightarrow P \left[ \text{detection error} \right] \approx 0.002\]

Thus in the coded system, 4 consecutive information bits are communicated correctly with probability

\[1 - 0.002 = 0.998\]

The corresponding probability without coding (and still assuming \( p = 0.01 \)) is

\[(1-p)^4 = 0.99^4 \approx 0.961\]

This comparison does not take into account the impact of coding on information bit rate and/or bandwidth (and thereon to noise power and thus the value of \( p \)).

With a little extra effort, the actual coding gain of hard decoder can also be analysed, by using the above results. We’ll elaborate shortly on that, next.
Coding Gain of Hard Decoding

The actual Coding Gain of hard decoding at SNR axis can also be analyzed. Below we use the (7,4) Hamming code as an example.

First we set the target error probability for an individual information bit, \( p_i \), for example \( p_i = 10^{-5} \).

Then in uncoded link this means (binary modulation assumed here)

\[
Q \left( \frac{a_u}{\sigma_u} \right) = Q \left( \sqrt{\frac{a_u^2}{\sigma_u^2}} \right) = Q \left( \sqrt{SNR_u} \right) \Rightarrow SNR_u \approx 12.6 dB
\]

Then the probability that a block of 4 consecutive information bits are communicated error free (without coding, assuming independent errors) is

\[
(1 - p_i)^4 \approx 1 - 4p_i = 1 - 4 \times 10^{-5}
\]

With coding included (see previous slides), the corresponding probability for error-free transmission of 4 information bits with (7,4) Hamming code is (can correct one raw error)

\[
(1 - p_c)^7 + 7p_c(1 - p_c)^6
\]

where \( p_c \) denotes the raw bit error probability in the coded system. Then we can solve (numerically, if nothing else)

\[
(1 - p_c)^7 + 7p_c(1 - p_c)^6 = 1 - 4 \times 10^{-5}
\]

which yields \( p_c \approx 4.5 \times 10^{-3} \).

This we can directly map to required received SNR in the coded link as (why? .. still assuming binary modulation)

\[
p_c = Q \left( \frac{a_c}{\sigma_c} \right) = Q \left( \sqrt{\frac{a_c^2}{\sigma_c^2}} \right) = Q \left( \sqrt{SNR_c} \right) \Rightarrow SNR_c \approx 10.1 dB
\]

Coding Gain of Hard Decoding (cont’d)

The difference in the required SNR’s is the basic Coding Gain (viewpoint #1), which is hence now

\[
12.6 dB - 10.1 dB \approx 2.5 dB
\]

If, in turn, we assume that information bit rate is fixed (and thus the coded system has bigger noise power, viewpoint #2), the Coding Gain is only

\[
12.6 dB - 10.1 dB - 10 \log_{10}(7/4) \approx 0.1 dB \quad (!?)
\]

Interpretations ?

Note that the corresponding Coding Gain numbers with soft decoding are 4.77dB and 2.34dB ! (check couple of slides back)
Parity Check Matrix and Hard Decoding

Like described earlier, hard ML decoding can in principle be implemented by calculating the Hamming distances of all valid code words to the received bit word and choosing the minimum

- High computational complexity, especially with bigger codes (lots of code words), at least if implemented brute force

Instead of above, many codes have rich algebraic structure which enables much more efficient decoding methods and algorithms (compared to brute force distance comparisons).

Below we sketch the basics of one decoding approach, based on the so-called parity check matrix and syndrome calculations.

Recap: for systematic \((n,k)\) code, the generator matrix is of the form

\[
G = \begin{bmatrix} \mathbf{I}_k & \mathbf{P} \end{bmatrix}
\]

Thus the code word can always be decomposed as

\[
c = \mathbf{b} \mathbf{G} = \begin{bmatrix} \mathbf{b} & \mathbf{a} \end{bmatrix}
\]

where \(\mathbf{b}\) is the \(k\)-bit source word (in row vector form) and \(\mathbf{a} = \mathbf{b} \mathbf{P}\) is the redundant part added by the code.

Thus we can work this forward as

\[
\mathbf{a} = \mathbf{b} \mathbf{P} \quad \Rightarrow \quad \mathbf{b} \mathbf{P} + \mathbf{a} = \mathbf{0} \quad \Rightarrow \quad \begin{bmatrix} \mathbf{b} & \mathbf{a} \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix} = \mathbf{0}
\]

\[
\Rightarrow \mathbf{c}^T \mathbf{H} = \mathbf{0} \quad \text{where} \quad \mathbf{H} = \begin{bmatrix} \mathbf{P}^T & \mathbf{I}_{n-k} \end{bmatrix}
\]

Above matrix \(\mathbf{H}\) is called parity check matrix. It also implies \(\mathbf{G}^T \mathbf{H} = \mathbf{0}\).

Examples

**Example 1**: \((k + 1, k)\) parity check code

\[
G = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_k & \mathbf{1} \end{bmatrix}
\]

Parity check matrix (well, here vector) is thus \(\mathbf{H} = [1 \ldots 1]\).

**Example 2**: \((7,4)\) Hamming code

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

Parity check matrix is in this case

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

For example \(\mathbf{c}_1 = [0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1]\) is a valid code word of this code since \(\mathbf{c}_1 \mathbf{H}^T = [0 \ 0 \ 0]\).

On the other hand, \(\mathbf{c}_2 = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1]\) is not a valid code word since \(\mathbf{c}_2 \mathbf{H}^T = [1 \ 0 \ 0]\).
**Parity Check Matrix and Syndrome**

Like sketched above, the parity check matrix is trivial to find for systematic codes.

Parity check matrix can in general be found for any linear code (systematic and non-systematic)

- In more mathematically-oriented literature, codes are actually typically described using parity check matrix (not generator matrix)

In hard decoding based error correction, the parity check matrix can be utilized as follows. First we define the so called syndrome as

\[ s = rH^T \]

where \( r \) denotes the received bit word (from raw detection), with possible errors in some bit positions. This can always be written also as

\[ r = c + e \]

where \( c \) is the original transmitted code word and \( e \) models bit errors (vector of all zeros except ones in those bit positions which are in error in the received word). By direct substitution we can now write:

\[ s = rH^T = cH^T + eH^T = eH^T \]

By definition, syndrome is zero if and only if the received word is a valid code word. Syndrome is also independent of the original code word and depends only on the error pattern \( e \) (see above equation).

Now, different values for \( eH^T \) can be tabulated in advance for all possible error patterns (within the error correction capabilities of the used code), and the error pattern can then be found in the receiver by comparing the syndrome \( s = rH^T \) with the tabulated values of \( eH^T \).

- easy and straight-forward to implement

Example(s) in the class-room exercises.

**Hamming codes**

Hamming codes are defined for the following block lengths

\[ (n,k) = (2^m - 1, 2^m - 1 - m) \]

where integer \( m > 1 \).

Hamming codes are an interesting class of codes for which the parity check matrix is of the dimensions \( (n-k) \times n = m \times (2^m - 1) \) and which contains all possible \( m \)-bit words as columns (except for all zero word).

Example: (7,4) Hamming code

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

Columns contain all different 3-bit words except all zeros.

It can be shown that all Hamming codes have minimum distance \( d_{H,\text{min}} = 3 \), and thus they can correct a single bit error \( t = 1 \). Hamming codes are also perfect codes.

Example in the classroom exercises.
Cyclic Codes

Definition: Linear \((n, k)\) code is cyclic if any arbitrary cyclic shift of any valid code word yields again a valid code word (code words form a closed set under cyclic shifts).

Such cyclic codes have very rich algebraic structure which enables very efficient decoding methods.

Cyclic codes can be defined using generator polynomial or parity check polynomial (instead of the corresponding matrices)

- this enables very efficient implementations, through polynomial algebra
- an example of the polynomial algebra will be provided in the classroom exercises

For example, many Hamming codes are cyclic.

Example: \((7,4)\) Hamming code

For example, the word 1000101 is a valid code word of this code (verify). Its cyclic shifts

\[
1100010 \quad 0110001 \quad 1011000 \quad 0101100 \quad 0010110 \quad 0001011 \quad 1000101
\]

are also valid code words (double check yourself). Code is cyclic.

Cyclic redundancy check, CRC, codes

CRC codes are, by definition, codes that are meant for simple error detection, not for error correction.

CRC codes are block codes, thus they append extra check bits at the end of the processing/coding block

- these are commonly called check bits in the CRC context
- if the check sum does not match in the receiver, or if the syndrome is non-zero, retransmission is requested
- naturally, the larger is the check bit overhead, the better are the error detection capabilities

CRC codes are also, by definition, cyclic block codes and can thus be processed and manipulated very efficiently

- syndrome calculations can be based on multiplying the received word, interpreted as a polynomial, with the parity check polynomial of the code
- alternative forms based on polynomial divisions also exist

Example: in LTE/LTE-Advanced radio network, CRC codes are used as follows

- at transport block level: adding 24 check bits per transport block
- at Turbo coding block level (much shorter than transport blocks): again adding 24 check bits, to assist e.g. early termination of the iterative Turbo decoding in the RX (we’ll talk about Turbo codes a little later)
- (if interested, double check the details from 3GPP LTE specs 😊)
Examples of Important Cyclic Codes

BCH codes (Bose, Chaudhuri, Hocquenghem)
For arbitrary positive integers $m$ and $t$, there exists a BSC code which can correct $t$ errors and for which

$$n = 2^m - 1, \quad k \geq n - mt$$

BSC codes are in practice one of the most important class of cyclic codes, due to their efficient decoding algorithms and different possibilities for flexible parameter selection and allocation.

Reed-Solomon (RS) codes
RS codes are important subset of BCH codes.

Also RS codes have very efficient decoding algorithms and implementations, and they can also correct error bursts up to certain level.

RS codes are pretty popular in, e.g., many digital TV broadcasting systems (e.g. DVB). A bit more information on these in the following slides.

Reed-Solomon (RS) Codes, in short (very short)

Reed-Solomon codes (RS codes) are defined for code word lengths

$$n = 2^m - 1$$

where $m$ is a positive integer. Furthermore, the code is composed of $m$-bit symbols. These RS codes are actually a subset of previous BCH codes.

$(n, k)$ Reed-Solomon code can correct

$$t = (n - k) / 2$$

t erroneous symbols per code word. Whether one symbol contains one or more bit errors is irrelevant! Thus these codes are typically deployed for correcting error bursts.

Example:

$(255, 235)$ RS code is deployed as the baseline code in some digital TV transmission systems (like DVB)

- Codeword length is $255 \times 8 = 2040$ bits
- Each code word carries 1880 information bits which consist of 235 8-bit symbols
- Code is designed such that it can correct 10 erroneous symbols in each code word; thus the maximum number of actual bit errors is 80 per code word (of 2040 bits)
- As an example, the code can correct a single heavy error burst of 72 consecutive bit errors within one code word (of 2040 bits)
Shortened Reed-Solomon Codes

Different possibilities for Reed-Solomon code word lengths (in bits) are
3×2, 7×3, 31×5, 63×6, 127×7, 255×8, ... (2^m – 1) × m)

Thus there is only very limited set of possible code lengths. This
problem can be partially relieved by a method called shortening (similar
ideas are applicable also with many other codes, in terms of puncturing
that is discussed a little later).

For example, using the basic (255,235) RS code as starting point, it can
be shortened into a (208,188) code by

1) Forming first a 235 symbol source word by combining 188 actual
source symbols with 47 zero symbols
2) Encode this with standard (235,255) RS code
3) Remove then the useless 47 zero symbols (known locations)
4) On the receiver side, these 47 zero symbols are then inserted
back (known locations) before decoding and removed after
decoding

CONVOLUTIONAL CODES

Previous block codes are totally memoryless. In convolutional codes,
the idea is to build memory into the coding processing.

Good convolutional codes can easily reach (or outperform) the
performance of good block codes, though the introduction of LDPC and
Polar codes (we talk about these later) has clearly diminished the
performance difference.

Complexity-wise, soft encoding is typically more feasible to implement
with convolutional codes (compared to block codes).

Key parameters in the continuation:
- Code rate \( r_c = k/n \)
- Constraint length \( M \) (or \( kM \), depends a bit on notational
conventions)

Convolutional encoder contains delay lines and modulo-2 adders such
that at each encoding cycle
- \( k \) information bits flow in, in parallel
- \( n \) coded bits flow out, in parallel
- Each output bit depends on the corresponding input bits and the
previous input bits of maximum \( M – 1 \) earlier encoding cycles
(block-level memory)

Like with block codes, convolutional code is said to be systematic if all
\( k \) input bits appear directly at output.

Also generator matrix and parity check matrix are, in principle, defined
similarly as with block codes.

With convolutional codes, however, the elements of these matrices are
binary-coefficient polynomials of the delay operator \( D \).

Examples will follow.
**Example: rate 2/3 encoder**

Corresponding bit-level expressions:

\[
\begin{align*}
C_k^{(1)} &= B_k^{(1)} \\
C_k^{(2)} &= B_k^{(2)} \\
C_k^{(3)} &= B_k^{(1)} \oplus B_{k-1}^{(1)} \oplus B_{k-1}^{(2)}
\end{align*}
\]

This code is clearly systematic.

Notation, now and in continuation:
- \(D\) refers to a delay unit of one bit interval
- \(\oplus\) refers to modulo-2 sum

**Generator Matrix and Parity Check Matrix**

A generic encoder can be represented as

\[
\begin{bmatrix}
C(0)(D) \\
\vdots \\
C(n)(D)
\end{bmatrix} = \begin{bmatrix}
B(0)(D) \\
\vdots \\
B(m)(D)
\end{bmatrix} G(D)
\]

where \(G(D)\) is the generator matrix and

\[
\begin{align*}
B(D) &= \begin{bmatrix}
B^{(1)}(D), B^{(2)}(D), \ldots, B^{(k)}(D)
\end{bmatrix} \\
C(D) &= \begin{bmatrix}
C^{(1)}(D), C^{(2)}(D), \ldots, C^{(n)}(D)
\end{bmatrix}
\end{align*}
\]

Here \(B^{(i)}(D)\) and \(C^{(i)}(D)\) denote "z-transforms" (D-transforms) of the incoming and outgoing bit-streams.

If \(M\) denotes the constraint length, the maximum order of binary polynomials in \(G(D)\) is \(M - 1\).

The connection between the generator matrix \(G(D)\) (and thereon the code words) and the parity check matrix \(H(D)\) is similar to those of block codes:

\[
C(D)H^T(D) = 0 \quad \text{or equivalently} \quad G(D)H^T(D) = 0
\]

Examples will follow, illustrating the connection between the encoder structure and the generator & parity check matrices.

We also show how a non-systematic code can be transformed into an equivalent systematic code with certain elementary manipulations

- though systematic codes are no better or worse than non-systematic ones – they are just, perhaps, more intuitive
Example 1: Rate 2/3 code

Code rate \( r_c = k/n = 2/3 \).
Constraint length \( M = 2 \).
Generator matrix:

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

Code is clearly systematic.

Parity check matrix is clearly \((C(D)H^T(D) = 0)\)

\[
H(D) = \begin{bmatrix}
1 & 0 & D & 1
\end{bmatrix}
\]

Well, let's verify that this really is the parity check matrix:

\[
C(D)H^T(D) = \begin{bmatrix}
B^{(1)}(D), B^{(2)}(D), (1 \oplus D)B^{(1)}(D) \oplus DB^{(2)}(D)
\end{bmatrix}
\begin{bmatrix}
1 \oplus D \\
D \\
1
\end{bmatrix}
\]

\[
= (1 \oplus D)B^{(1)}(D) \oplus DB^{(2)}(D) \oplus (1 \oplus D)B^{(1)}(D) \oplus DB^{(2)}(D)
\]

\[
= 0
\]

This applies with any input bit combination.

Example 1: Rate 2/3 code (cont’d)

Starting from the previous parity check matrix alone, it is easy to develop a corresponding systematic encoder. You simply pick

\[
C^{(1)}(D) = B^{(1)}(D), C^{(2)}(D) = B^{(2)}(D)
\]

and based on parity check matrix, \( C^{(3)}(D) \) is then:

\[
C(D)H^T(D) = (1 \oplus D)C^{(1)}(D) \oplus DC^{(2)}(D) \oplus C^{(3)}(D) = 0
\]

\[
\Rightarrow C^{(3)}(D) = (1 \oplus D)C^{(1)}(D) \oplus DC^{(2)}(D)
\]

Obviously we obtained the original code.
**Example 2: Rate 1/2 code**

Code rate $r_c = k/n = 1/2$.

Constraint length $M = 3$.

Generator matrix:

$G(D) = \begin{bmatrix} 1 \oplus D^2, & 1 \oplus D \oplus D^2 \end{bmatrix}$

This code is not systematic.

Parity check matrix is clearly $(C(D)H^T(D) = 0)$

$H(D) = \left[ (1 \oplus D \oplus D^2), (1 \oplus D^2) \right]$  

Using the previous parity check matrix, we can construct an equivalent systematic encoder as follows. Since

$$C(D)H^T(D) = (1 \oplus D \oplus D^2)C^{(1)}(D) \oplus (1 \oplus D^2)C^{(2)}(D) = 0$$

and choosing $C^{(1)}(D) = B(D)$ (looking for a systematic code), we get

$$(1 \oplus D \oplus D^2)B(D) = (1 \oplus D^2)C^{(2)}(D)$$

This can be implemented with the following recursive structure.

Since this is a linear system, we can also change the order of the recursive and non-recursive parts, which yields the structure in part (b).

The obtained encoder has identical parity check matrix with the original non-systematic code. Thus these two codes are fully equivalent from error control point of view.
Convolutional codes are clearly systems which contain memory. We can model them as Markov-chains (state-machines), as they are essentially shift-register processes.

Thus the decoding can then be built on the Maximum Likelihood (ML) sequence detection principle, implemented with Viterbi-algorithm:

- the very same processing engine that we used for maximum likelihood sequence detection under ISI, earlier

In the following, we present both hard and soft Viterbi decoding. In general, ML detector chooses the code word (here coded sequence) which is closest to the observation in the sense of:

- Euclidian distance to soft observation, in case of soft decoder
  - corresponding branch metrics in Viterbi
- Hamming distance to raw bit detection, in case of hard decoder
  - corresponding branch metrics in Viterbi

Notice that due to memory, the term code word means here the whole encoded sequence. In case of soft encoder, of course also the symbol mapping needs to be taken into account.

In decoding context, it is useful to represent the encoder using a trellis diagram:

- States correspond to different possible bit combinations inside the shift-registers.

Decoding complexity depends essentially on the number of different states, which in this case is essentially related to the total number of internal shift-registers (maximum of \( k(M - 1) \)).

As a concrete example, we consider the previous rate 1/2 convolutional code whose block-diagram is reproduced below:

Clearly, this encoder has two state variables, namely \( B_{k-1} \) and \( B_{k-2} \).

Hence the state at time \( k \) is \( \Psi_k = \{ B_{k-1}, B_{k-2} \} \).

Below graphs then show the one stage of the corresponding trellis diagrams as follows:

- left: one stage of the trellis with state-transitions labeled with input bit and two coded output bits (used by hard decoder)
- right: one stage of the trellis with state-transitions labeled with input bit and two coded output symbols (assuming binary modulation here; used by soft decoder)
Example: Trellis Diagram for Previous Rate 1/2 Code

(a): complete trellis diagram corresponding to encoding of $K$ information bits, initial and final states assumed all zero; branch metrics in Viterbi processing using either Hamming or Euclidean distance.

(b): one stage of bit-level trellis, used by hard decoder

(c): one stage of symbol-level trellis (assuming binary modulation $\{+a,-a\}$), used by soft decoder

Concrete examples will be provided in the classroom exercises.

Performance of Convolutional Codes

Usual assumption is that channel noise (say $N_k$ in above transmission chain) is Gaussian distributed and white. Then Viterbi-algorithm with soft metrics is the optimum ML-sequence detector.

Suppose we further assume again that binary modulation is used (for simplicity of analysis), with following notations:

- Symbol alphabet
  - Uncoded case: $\pm a_u$
  - Coded case: $\pm a_c$

- Noise variance
  - Uncoded case: $\sigma_u^2$
  - Coded case: $\sigma_c^2$

Depending whether information bit rate or bandwidth is fixed, then $\sigma_c^2 = (n/k)\sigma_u^2$ or $\sigma_c^2 = \sigma_u^2$.

In the following, we compare the performance of uncoded and coded digital communication chain in case of soft decoding. The previous rate 1/2 code is used as a practical example code.

Analysis is again based on bit error rate of information bits, building on

$$P[\text{bit error, uncoded}] = Q\left(\frac{a_u}{\sigma_u}\right) = Q\left(\sqrt{\frac{d_{u,\min}}{\sigma_u^2}}\right)$$

$$P[\text{bit error, coded}] \approx Q\left(\frac{d_{E,\min}}{2\sigma_c}\right) = Q\left(\sqrt{\frac{d_{H,\min}a_c^2}{\sigma_c^2}}\right)$$
Example: Rate 1/2 code and Soft Decoding

Notation:
- Soft noisy observations: \( y_k, y_{k+1}, \ldots \)
- Detected symbols: \( \hat{a}_k, \hat{a}_{k+1}, \ldots \)

In the considered example code, each stage in the trellis diagram corresponds to two symbol instants. They are decoded to a single information bit. Branch metric in soft ML Viterbi-decoder is

\[
d^2_{k,E} = (y_k - \hat{a}_k)^2 + (y_{k+1} - \hat{a}_{k+1})^2
\]

Decoder chooses the path which minimizes the cumulative metric (weighted total path length).

In terms of decoding error probability, the key thing is the minimum distance between the correct path and other paths. For the rate 1/2 code, one example of such minimum distance is (assuming that the correct path is all zeros):

\[
d^2_{E,\text{min}} = 20a_c^2
\]

It is easy to verify that this really is the minimum distance. Also each path has one other path at this distance.

Example: Rate 1/2 code and Soft Decoding (cont’d)

Thus the information bit error rate is now approximately

\[
P[\text{bit error, coded}] \approx Q\left(\frac{\sqrt{20}a_c}{2\sigma_c}\right) = Q\left(\sqrt{\frac{5a_c^2}{\sigma_c^2}}\right)
\]

Then the bit error rate without and with coding are approximately equal when

\[
\frac{a_c^2}{\sigma_u^2} = \frac{5a_c^2}{\sigma_c^2} \iff SNR_u = 5 \times SNR_c
\]

Thus the Coding Gain in terms of SNR is directly

\[
\text{Coding Gain}_1 = 10\log_{10}(5) \approx 7 \text{ dB}
\]

This is the true gain also from transmit/received useful signal power point of view if the bandwidth is fixed (and thereon the noise power, \( \sigma_c^2 = \sigma_u^2 \)).

If, in turn, we assume, that the information bit rate is fixed (in comparing uncoded and coded chains), the noise powers are related as \( \sigma_c^2 = (n/k)\sigma_u^2 = 2\sigma_u^2 \), and the Coding Gain is

\[
\text{Coding Gain}_2 = 10\log_{10}(5/2) \approx 4 \text{ dB}
\]
**Example: Rate 1/2 code and Hard Decoding**

In **hard decoding**, the mapping from transmitted coded bits to received raw bits is essentially a binary symmetric channel (BSC):

- Raw bit error probability \( p \)

Analysis proceeds then essentially similar to soft decoding, the differences are:

- Distance metrics build on Hamming-distance
- Instead of Gaussian \( Q(\cdot) \)-function, the error probabilities follow binomial distribution (notation \( Q(\cdot, \cdot) \), like we have learned earlier)

It is easy to see that for the earlier rate 1/2 code, one example of paths at minimum distance is as follows:

The direct (correct) path corresponds to coded bits (0,0) (0,0) (0,0) while the other path corresponds to (1,1) (0,1) (1,1)

The Hamming-distance is 5 which yields the error probability

\[
P\left[ \text{bit error, coded} \right] = Q(5, p) = 10p^3(1 - p)^2 + 5p^4(1 - p) + p^5
\approx 10p^3
\]

In above, the last form applies with reasonably small \( p \). Note that the above expression is, by default, for decoding error but we approximate also the actual bit error probability with it.

---

**Example: Rate 1/2 code and Hard Decoding** (cont’d)

We can now compare the uncoded and coded transmission chain at some target information bit error probability, say \( 10^{-5} \). Then in the coded case we have

\[
10p^3 = 10^{-5} \implies p = 0.01
\]

This can now be easily mapped to SNR requirement, or equivalently needed signal level \( a_c \) such that above raw error rate level is achieved (here assumed that \( \sigma_c^2 = (n/k)\sigma_u^2 = 2\sigma_u^2 \)):

\[
Q(a_c/\sigma_c) = Q(a_c/\sqrt{2}\sigma_u) = 0.01 \implies a_c \approx 3.4\sigma_u
\]

Similarly in the uncoded case, we can reach the error probability of \( 10^{-5} \) when

\[
Q(a_u/\sigma_u) = 10^{-5} \implies a_u \approx 4.3\sigma_u
\]

Thus the **Coding Gain** is

\[
\text{Coding Gain}_{1/2} = 10\log_{10}(4.3/3.4)^2 \approx 2 \text{ dB}
\]

Note that this is considerably smaller than the corresponding Coding Gain in soft decoding (which was 4 dB).

Despite this, hard decoding is also widely applied due to its very simple implementation.
A bit more about Hard Decoding

Sometimes in the literature, the concept free distance, \( d_{\text{free}} \), is used in convolutional coding context. It is essentially identical to minimum Hamming distance \((d_{H,\text{min}})\), and thus describes the coding performance especially in hard decoding. Previous example rate 1/2 code has free distance of 5.

In the following table, we list some widely used convolutional codes from literature.

Table below gives the generator matrix elements in octal form (for presentation compactness - e.g. 5,7 corresponds to previous rate 1/2 example code).

<table>
<thead>
<tr>
<th>( n )</th>
<th>( k )</th>
<th>( d_{\text{free}} )</th>
<th>Generator matrix elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>5,7</td>
</tr>
<tr>
<td>6</td>
<td>15,17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>23,35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>65,57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>133,171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>345,237</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>561,753</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1167,1545</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2335,3661</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5,7,7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>13,15,17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>25,33,37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>47,53,75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>133,145,175</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>113,155,173</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>557,663,711</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1117,1365,1633</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>2353,2671,3175</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>5,7,7</td>
</tr>
<tr>
<td>10</td>
<td>13,15,17,15,17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>25,33,37,33,37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>47,53,75,75,75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>133,145,175,175</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>113,155,173,173</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>557,663,711,711</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1117,1365,1633,1633</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>2353,2671,3175,3175</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Example: 3G WCDMA/UMTS

- Rate 1/2 code
  \([561, 753]\)
- Rate 1/3 code
  \([557, 663, 711]\)
- Other code rates available through puncturing (see next slides)

Puncturing and Punctured Convolutional Codes

Punctured convolutional code means that some encoded bits are systematically removed from the output of a standard encoder. This is done of course such that the information bits can still be decoded correctly, at least in the case of error free transmission.

In puncturing, the code rate is increased and free distance gets usually smaller.

The principal purpose of puncturing is that we can easily and flexibly obtain many different codes with different code rates from a single mother code.

The following tables show examples of punctured convolutional codes with code rates 2/3 and 3/4, starting from a rate 1/2 mother code.

\[
\begin{align*}
\text{rate} & = \frac{2}{3} & \text{rate} & = \frac{3}{4} \\
\text{Generator matrix} & \quad \text{Generator matrix} \\
\begin{array}{c|c}
\hline
\text{d}_{\text{free}} & \text{Generator matrix} \\
\hline
3 & 7,5,7,X \\
4 & 15,13,15,X \\
5 & 31,33,31,X \\
6 & 73,41,73,X \\
6 & 163,135,163,X \\
8 & 337,251,337,X \\
8 & 661,473,661,X \\
\hline
\end{array}
\end{align*}
\]

\[
\begin{align*}
\text{generator matrix} & \quad \text{generator matrix} \\
\begin{array}{c|c}
\hline
\text{d}_{\text{free}} & \text{generator matrix} \\
\hline
3 & 5,7,5,X,X,7 \\
4 & 15,17,15,X,X,17 \\
5 & 25,37,X,37,X \\
5 & 31,33,X,33,X,33 \\
6 & 135,163,X,163 \\
6 & 205,307,X,307 \\
6 & 515,737,X,737 \\
\hline
\end{array}
\end{align*}
\]
**Punctured Convolutional Codes (cont’d)**

**Example:**

Code whose generator matrix is \( (7,5,7,X) \) is obtained from the basic rate 1/2 code \( (7,5) \) such that the second output bit is transmitted only at every other coding cycle.

The code \( (5,7,5,X,X,7) \), in turn, is obtained from basic rate 1/2 code \( (5,7) \) such that we transmit:
- Both output bits at first coding cycle
- Only the first output bit at second coding cycle
- Only the second output bit at third coding cycle
- Then repeat

At overall bit level, this means
\[
C_1^{(1)}, C_1^{(2)}, C_2^{(1)}, C_3^{(2)}, C_4^{(1)}, C_4^{(2)}, C_5^{(1)}, C_6^{(2)}, C_7^{(1)}, \ldots
\]

Puncturing is used in almost all existing and emerging digital communication systems.

Proper puncturing level, and thus code rate and free-distance, is chosen based on the required information bitrate, on one side, and on (assumed) channel quality, on the other side. The receiver can anyway always decode the chosen punctured code.

---

**Some Performance Comparisons from Literature**

Some Coding Gains (or upper bounds for Coding Gain) reported in the literature, as functions of constraint length and free distance:

<table>
<thead>
<tr>
<th>Coding Rate 1/2</th>
<th>( M )</th>
<th>( d_f )</th>
<th>Upper Bound (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1/2 )</td>
<td>3</td>
<td>5</td>
<td>3.97</td>
</tr>
<tr>
<td>( 1/3 )</td>
<td>4</td>
<td>6</td>
<td>4.76</td>
</tr>
<tr>
<td>( 2/3 )</td>
<td>5</td>
<td>7</td>
<td>5.43</td>
</tr>
<tr>
<td>( 3/4 )</td>
<td>6</td>
<td>8</td>
<td>6.00</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>7</td>
<td>10</td>
<td>6.99</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>8</td>
<td>10</td>
<td>6.99</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>9</td>
<td>12</td>
<td>7.78</td>
</tr>
</tbody>
</table>

Comparison of systematic and non-systematic codes in terms of free distance, code rate 1/2:

<table>
<thead>
<tr>
<th>Constraint Length</th>
<th>Free Distance</th>
<th>Free Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2 )</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( 3 )</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>( 4 )</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>( 5 )</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>( 6 )</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>( 7 )</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>( 8 )</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

Soft Coding Gains (dB) in Viterbi decoding, different coding rates and constraint lengths:

<table>
<thead>
<tr>
<th>Uncoded ( E_b/N_0 ) (dB)</th>
<th>Coding Rate ( P_B )</th>
<th>( M )</th>
<th>( 1/3 )</th>
<th>( 1/2 )</th>
<th>( 2/3 )</th>
<th>( 3/4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.8 ( 10^{-3} )</td>
<td>4.2</td>
<td>3.3</td>
<td>3.5</td>
<td>3.8</td>
<td>2.9</td>
<td>3.1</td>
</tr>
<tr>
<td>9.6 ( 10^{-3} )</td>
<td>5.7</td>
<td>4.3</td>
<td>4.6</td>
<td>5.1</td>
<td>4.2</td>
<td>4.6</td>
</tr>
<tr>
<td>11.3 ( 10^{-7} )</td>
<td>6.2</td>
<td>4.9</td>
<td>5.3</td>
<td>5.8</td>
<td>4.7</td>
<td>5.2</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>7.0</td>
<td>5.4</td>
<td>6.0</td>
<td>7.0</td>
<td>5.2</td>
<td>6.7</td>
</tr>
</tbody>
</table>
Concatenated Coding

In concatenated coding, the idea is to combine two separate encoding stages, with a process called interleaving in between. The benefit is that the computational complexity is typically much lower compared to a single coding stage (with similar performance as a whole).

**Inner code** can be e.g. a convolutional code, and decoding on the receiver side is done using soft Viterbi-decoding. Different coding rates can be obtained, e.g., with puncturing.

**Outer code** could be then a powerful block code (e.g. Reed-Solomon code), possibly shortened.

Inner decoder may yield *error bursts*. By interleaving (and deinterleaving), the effects can be reduced since deinterleaving spreads the error burst to a wider time window (and thus the outer decoder can correct the errors better).

An example of typical error rates in different interfaces of the receiver chain is:

- raw BER $\sim 10^{-1}$
- BER after inner decoder $\sim 10^{-4}$
- BER after outer decoder $\sim 10^{-9} \ldots 10^{-12}$

Interleaving

Interleaving mixes the order of the bits in a deterministic manner. The purpose is to spread consecutive bits as far as possible. This way, the corresponding deinterleaving on receiver side spreads possible error burst efficiently.

Implementation challenges are related to needed memory. Also the overall delay in the transmission chain is increased if the interleaving window is increased.

Interleaving is typically used together with concatenated coding, but it can also be used with only one coding stage to break error bursts (e.g. at hard decoder input).

There are two common interleaving principles:

- **Block interleaving**
  A table of $M \times N$ elements is read in column-wise and read out row-wise. Deinterleaving works then vise versa.

- **Convolutional interleaving**
  A bit more complicated structure which reduces the memory requirements.
RETRANSMISSION CONCEPTS, ARQ AND HARQ, in short

The basic retransmission approach is Automatic Repeat reQuest (ARQ) where the system adopts error detection code (e.g. CRC based) and signals the outcome to the transmitter through feedback/control channel

- **ACK**: positive acknowledgment; CRC check OK, packet or block interpreted error-free
- **NACK**: negative acknowledgment; CRC check failed, packet or block contains errors

In case of NACK, the receiver discards the packet (in the very basic ARQ-based approaches), and the TX retransmits the packet

Some variants are

- **TX** waits for ACK before sending any new packet, if not received until time-out resend the earlier packet
- **TX** can send multiple packets with identifiers, receiver sends ACKs with identifiers for next expected packets
- **TX** can send multiple packets, individual ACK's per packet

Retransmission methods, particularly hybrid ARQ (HARQ, next page) allows for more aggressive link adaptation

- first transmission target packet/block error rate can be even 10%, which allows for using aggressive (high) modulation order and high(er) code rate, than what could be used if there were no retransmissions
- reliability is then improved through HARQ, as long as it works fast enough such that the latency does not become too high

ARQ and HARQ, cont'd

While ARQ combines error detection and retransmissions, the concept of Hybrid ARQ (HARQ) is more advanced and combines error correction capabilities of the error control codes with retransmissions

Most simple version of HARQ is that the error correction codes correct for the errors that they can, and if the CRC check fails after that, then retransmission of the packet is requested

This, however, is not yet the best or most efficient way to process the received signal.

In HARQ with soft combining, the original received packet, even if not correctly decoded in full, is stored in RX memory and later combined with the retransmitted signal to obtain a more reliable observation of the original information

While the information bits behind the retransmitted signal must naturally be the same as in the original transmission, the exact set of coded bits can actually be different

- **Chase combining**: same set of coded bits used, in retransmission
  - simple implementation, can be seen as additional repetition coding
  - RX can combine the original and retransmitted signals directly, particularly if also the modulation scheme is the same between different (re)transmissions
- **Incremental redundancy**: here different sets of coded bits are used, in retransmission
  - in particular with systematic codes, this means new and different parity bits (redundancy part) and one can view that after the receiver has combined the receptions at coded bit level, the code rate has decreased
  - individual code rate and modulation scheme, per transmission can vary
ARQ and HARQ, cont’d

Below illustrates LTE downlink protocol architecture, pinpointing also ARQ at radio link control (RLC) layer and HARQ at MAC layer
- since HARQ operates close to physical layer, it is fast and thus many services allow for multiple retransmissions
- each UE can also have multiple simultaneous HARQ processes, basically one per transport block (up to 8, in parallel, in baseline LTE)
- HARQ is only applied in the shared channel (not for broadcast transmissions)

MORE ADVANCED CHANNEL CODING METHODS:
TURBO, LDPC AND POLAR CODES

In this part of the course, a short introductory to modern, binary channel codes is given. We discuss in particular
- turbo codes (LTE shared channel, Wimax)
- LDPC codes (WLAN 802.11n and onwards, Wimax, 5G shared data channel)
- Polar codes (5G control channel)

As discussed already in the early parts of the course, the Hartley-Shannon law defines the channel capacity (theoretical maximum mutual information) between TX and RX, in the form of $C \equiv W \log_2 (1 + SINR)$. Shannon also proved that we can achieve the channel capacity with very long, random like codes.

The question ever since has been how to build such capacity approaching codes.

In principle generating random like codes is actually easy, but the decoding complexity is the issue. As we make the code more random or increase the code word length, the decoder complexity increases also.
- Finding codes with required randomness and still to have simple decoding algorithms is difficult.

Turbo and LDPC codes rely on very simple component codes, which are then decoded iteratively by exchanging information between component decoders.

Polar code, in turn, is a “new” rival, which currently relies on list detector to achieve good performance.

Quote of today by A. Viterbi, 1991: “Never prematurely discard the information that may be useful in making a decision until after all decisions related to that information have been completed.”
Motivation / Nice to know – before we kickoff

How good is turbo code?

In the given example*, the original turbo code and the code’s free
distance asymptote, and the simulated performance of the (2,1,14)
maximum free distance (MFD) convolutional code and its free distance
asymptote are given.

The (2,1,14) MFD code has $2^{14}$ (16384!) states and belongs to the
family of codes chosen for the Galileo mission and is decoded by using the Big Viterbi decoder built by Jet Propulsion Laboratory, which is the
most complex decoder ever realized in hardware.

Materials in this part are prepared by Dr. Toni Levanen (thanks
very much Toni !), and from course requirements perspectives,
understanding only the very basic concepts is sufficient.


TURBO CODES: ENCODING

Turbo encoder uses simple convolutional encoders and an interleaver
to build a systematic code word, which is then decoded iteratively by
exchanging information between component decoders.

The original encoder structure* was a systematic encoder based on two
recursive systematic convolutional codes and a pseudorandom
interleaver. Examples of systematic and non-systematic, and recursive
and non-recursive encoders below ($G = [7; 5]$).

**The component encoders can be different or the number of encoders
can be increased, but it has been shown that most of the coding gain is
achieved by using two identical component codes. Below the LTE turbo
coder is given** as an example:


Turbo codes: examples of parallel and serial concatenation

In parallel concatenation, as used in LTE, the same information sequence acts as an input to both component encoders. The second component encoder receives an interleaved (permuted) version of the same data sequence. In the decoder, extrinsic information related to data bits is exchanged between component decoders.

![Diagram of parallel concatenation](image)

In serial concatenation, the first component encoder acts as the outer code and the second as the inner code. The inner code provides all soft information related to its data bits to outer code, and outer code provides extrinsic information related all coded bits to inner code.

![Diagram of serial concatenation](image)

Extrinsic information and SISO MAP decoding are described in more detail in turbo decoding section, and in the complementary materials part at the end of the lecture notes.

Turbo codes: inner interleaver

The main tasks of the interleaver is to reduce correlation between the inputs of the encoders and to efficiently reduce the probability of a data sequence which can generate code words of low Hamming weight in all component encoders. Several different interleaver designs exists, from which we discuss two candidates, pseudorandom and the Quadrature Permutation Polynomial (QPP) interleaver.

Pseudorandom interleaver defines a random like permutation between the input and output bit indices per codeword, and is changed between each codeword. Random interleavers provide the best performance. This is intuitive from the channel capacity point of view, which is achieved with random codes.

The Quadrature Permutation Polynomial (QPP) used in LTE is maximum contention free, indicating that all integer window lengths, $W$, dividing the interleaver length, $I$, will be contention free. By contention free is meant that as the $N = I/W$ parallel processors access the memory regarding extrinsic information for $j^{th}$ bit, all the processors read different memory cells. QPP allows parallelized decoder design for high rate turbo decoders. Below example for $j = 2^*$. 

![Diagram of QPP interleaver](image)

**Turbo codes: inner interleaver**

One aspect related to interleavers and the data block length is the so-called interleaver gain. As the interleaver length (data block length) increases, the turbo code performs better. This is due to the improved decorrelation of the soft information by the interleaver.

Given below is an example of the performance in an AWGN channel with different interleaver (data block) sizes. The code rate is $R=1/2$ and S-interleaver was used. In the decoder 18 decoding iterations were assumed. We can see how the interleaver size affects the steepness of the slope in the waterfall region and decreases the error floor. Also, for shorter block sizes the performance saturates faster (with less iterations) than with larger block sizes (why?).

For reference, in LTE the maximum code block size is 6144 bits. If the transport block is larger than this, then it is divided into multiple code blocks.

**Turbo codes: decoding**

Already in 1965 Forney proposed concatenated coding, which has been successfully applied in several applications, so there’s nothing new in this sense. The “turbo” in turbo codes is the iterative soft information exchange in the decoder, where the soft information exchange provides that no information is lost in the decoding process. Example block diagram given below, where SISO MAP (Soft-Input Soft-Output Maximum A Posteriori probability) decoders are used.

Extrinsic information exchange between component decoders.

Next, let’s look at an example performance for a turbo code in an AWGN channel.

Error floor is not visible, but it is there. Error floor is a more typical phenomenon for parallel concatenated turbo codes when compared to serial concatenated turbo codes.
Turbo codes: decoding

To describe the SISO MAP decoding, we start with bit-wise MAP criterion given as

\[ L(\hat{u}_i) = \ln \frac{P(u_i = 0, y)}{P(u_i = 1, y)} = \ln \frac{P(u_i = 0 | y)}{P(u_i = 1 | y)} \]

After some manipulations, we obtain for a systematic code, bit-wise MAP probability given as

\[
L(\hat{u}_i) = \ln \frac{P(y_j | c_j = +1)}{P(y_j | c_j = -1)} \cdot \ln \frac{P(u_i = 0)}{P(u_i = 1)} + \ln \frac{\sum_{c_j} P(y_j | c_j) \cdot \prod_{j \neq i} P(u_j)}{\sum_{c_j} P(y_j | c_j) \cdot \prod_{j \neq i} P(u_j)}
\]

\[ = L_{ch,y}(u_i) + L_a(u_i) + L_e(u_i) \]

The soft-output per data bit \( u_i \) can be split into 3 statistically independent log-likelihood ratios (LLRs)

- Systematic part \( L_{ch,y}(u_i) \) (received through the channel)
- a-priori information \( L_a(u_i) \) (from other component decoders). This is interleaved extrinsic information received from other decoders!
- Extrinsic information \( L_e(u_i) \) (generated in this decoder). This is a priori information for other decoders after interleaving!

Turbo codes, summary and discussion

Interleavers are an important part of turbo code design

- Decrease correlation between component decoders
- Decrease the probability of simultaneous low Hamming weight code words from all component encoders (break “bad” input sequences)

Component codes should be recursive, in order to provide interleaver gain, and the feedback polynomial should be primitive polynomial.

Extrinsic information is exchanged between component decoders. This is information generated by a component decoder excluding all the information in the component decoder input. After interleaving, extrinsic information is considered as a priori information for the next component decoder. Extrinsic information evolution is analyzed with the help of EXIT-charts (more information available in the complementary material section at the very end of the lecture notes).

Efficient implementations use log-domain decoding together with sliding window decoders and contention-free interleaver designs. Some approximation of the Jacobian algorithm is applied to simplify forward, backward and transition probability computation. Efficient decoder first calculates backward probabilities, and then per time instant new forward probabilities and extrinsic information to the output.

Turbo decoder has no inherent structure to stop iterating if the codeword has been decoded correctly. Several methods exists. In LTE, CRC is added to each codeword and evaluated after each iteration for early stopping.

For similar length codewords, one turbo decoder iteration is typically clearly more complex than one LDPC decoder iteration. For similar performance, although LDPC uses more decoding iterations, the overall complexity is typically similar. Do not compare turbo and LDPC only by the number of iterations!

In general, more information regarding Turbo codes available in the complementary material section at the very end of the lecture notes.
Turbo principle on a larger scope

Generally speaking iterative decoding or receiver design is based on exchanging information between soft information processors, given that the exchanged information is not otherwise available to the soft information processor and not generated by the receiving soft information processor. Examples of soft information processors in RX are

- Synchronization (fine tuning)
- Channel estimator
- Soft interference cancellation
- Channel decoder
- Phase noise, carrier frequency offset, Doppler compensation

where $\pi$ represents an interleaver function.

LOW-DENSITY PARITY-CHECK (LDPC) CODES

Low-density parity-check (LDPC) codes were originally presented by Gallager* in his Ph.D. thesis already in 1963 and re-invented by MacKay and Neil** in the late 1990s, at the same time being also actively researched by Richardson and Urbanke independently.

LDPC codes are linear block codes with sparse parity check matrix $H$

- Sparse means that $H$ contains relatively few ones among many zeros (in the case of binary codes)
- Complexity of the decoding algorithm is proportional to the number of ones in the parity check matrix $H$. This is why we want a sparse parity check matrix.

Efficient decoding is based on the Tanner graph*** of the parity check matrix.

Recall that for every linear binary $(n, k)$ code $C$ with code rate $R = k/n$

- there is a generator matrix $G$ such that code words $x$ and data words $u$ are related by $x = uG$ (... or $c = bG$ like we wrote it earlier, sorry for slightly revised notation)
- there is a parity-check matrix $H$ of rank $n - k$ such that $xH^T = 0$
- the relation between the generator and parity-check matrix is $GH^T = 0$

Main advantages of LDPC codes

- Very good codes (can achieve performance close to capacity)
- Low decoding complexity
- Old age => patents related to main idea dissolved (although lots of new ones, thanks to Wimax, WLAN 802.11, and 5G)

---

LDPC codes: encoding

LDPC codes can be systematically encoded by a systematic generator matrix

\[ G = [I_{k \times k} | P_{k \times n-k}] \]

The matrix \( P \) can be found by transforming \( H \) into another check matrix of the code that has the form

\[ H' = [P_{k \times n-k}^T | I_{n-k \times n-k}] \]

The systematic format of \( H \) can always be found, e.g., via Gauss-Jordan elimination.

The problem with encoding via \( G \) is that the submatrix \( P \) is generally not sparse, so for codes of practical interest the encoding complexity is high.

- Modern LDPC codes have a structure which enables efficient encoding and decoding.

Irregular quasi-cyclic LDPC (QC-LDPC) codes are used in WLAN 802.11n and onwards (.11ac, etc) and have been approved (currently a working assumption) for 5G as shared channel codes. These codes allow efficient encoding through a method proposed by Richardson and Urbanke*. Furthermore, these codes have been shown to achieve similar performance to random LDPC codes**. In the next page, an example of the quasi-cyclic LDPC code parameterization from IEEE 802.11n WLAN standard is given.

QC-LDPC codes are based on parity check matrices which are built from cyclically shifted identity matrices. This design reduces the memory required to store different parity check matrices defined for different block sizes and coding rates.

In the 5G standardization, special care also on the aspects of rate matching, chase combining, and incremental redundancy have been taken into account in the design of the proposed LDPC codes.

---

** S. Myung, K. Yang, and J. Kim, “Quasi-Cyclic LDPC Codes for Fast Encoding,” 2005
**LDPC codes: regular vs irregular designs**

A regular \((d_v, d_c)\) LDPC code of length \(n\) is defined by a parity-check matrix \(H\) with \(d_v\) ones in each column and \(d_c\) ones in each row. These are the simplest form of LDPC codes. Good for pen and paper exercises, but we can do better.

Irregular LDPC codes are a generalization of regular LDPC codes, which provide lower error rates

- Can be optimized to either improve the waterfall-region or the error floor, depending on the application
- Irregular (varying) number of ones per column and per row, which leads to variable and check nodes of different degree

A Tanner graph of a code is a graphical representation of the code constraints defined by a parity-check matrix with

- a variable node (\(v\)-node) for each code symbol
- a check node (\(c\)-node) for each check equation
- an edge between a variable node and a check node if the code symbol participates in the check equation (each edge in the graph corresponds to a 1 in the parity-check matrix)

Nodes of the same type cannot be connected! (bipartite graph)

**Example:**

A regular \((d_v, d_c)\) LDPC code of length \(n\) is defined by a parity-check matrix \(H\) with \(d_v\) ones in each column and \(d_c\) ones in each row. These are the simplest form of LDPC codes. Good for pen and paper exercises, but we can do better.

Irregular LDPC codes are a generalization of regular LDPC codes, which provide lower error rates

- Can be optimized to either improve the waterfall-region or the error floor, depending on the application
- Irregular (varying) number of ones per column and per row, which leads to variable and check nodes of different degree

A Tanner graph of a code is a graphical representation of the code constraints defined by a parity-check matrix with

- a variable node (\(v\)-node) for each code symbol
- a check node (\(c\)-node) for each check equation
- an edge between a variable node and a check node if the code symbol participates in the check equation (each edge in the graph corresponds to a 1 in the parity-check matrix)

Nodes of the same type cannot be connected! (bipartite graph)

**Example:**

A Tanner graph of a code is a graphical representation of the code constraints defined by a parity-check matrix with

- a variable node (\(v\)-node) for each code symbol
- a check node (\(c\)-node) for each check equation
- an edge between a variable node and a check node if the code symbol participates in the check equation (each edge in the graph corresponds to a 1 in the parity-check matrix)

Nodes of the same type cannot be connected! (bipartite graph)

**LDPC codes: decoding**

In LDPC decoding the aim is to compute the marginal a posterior probabilities \(P(x_n = 1 | y, H)\) for each \(n\)

- note that even with systematic LDPC code the parity bit related marginal posterior probabilities are updated during the decoding process although not needed in the output

Strictly speaking, the exact decoding of these marginal posterior probabilities is very difficult if the graph contains cycles. However, it is interesting and efficient to implement a decoding algorithm that would be appropriate if there were no cycles (in other words, they also get the job done).

Short cycles in an LDPC code have similar effect as short interleaver in turbo codes, the extrinsic information exchanged between nodes is not anymore independent of the input and decoder performance does not improve with additional iterations.

**Example of a length 4 cycle (worst case):**

Some further details available for LDPC decoding in the complementary material section at the very end of the lecture notes.
LDPC codes, some conclusions

Linear block codes defined through a sparse parity-check matrix $H$. The sparseness of the parity-check matrix defines the decoding complexity.

$H$ can be regular or irregular. Typically irregular designs are used due to their better performance. Irregular designs can be optimized in terms of waterfall-region or error floor performance, depending on the application.

**Advantages of LDPC over turbo codes:**
- Allow decoder structures with higher degree of parallelization
- Have been shown to provide lower memory requirements and lower chip/IC area than turbo codes
- More amenable to high code rates
  - Turbo codes require heavy puncturing of the $R = 1/3$ mother code to achieve high code rates. LDPC designs are typically optimized separately for different rates.
- Generally possess lower error floor (for certain block length and code rate)
- Can work without outer interleaving
- In-built detection of correctly decoded code words for early stopping
- Dissolved patent issues for the basic design

As with turbo codes, the basic theory and good practices for design, analysis, and decoding exist.

If interest in LDPC codes was invoked, the spatially coupled LDPC codes are an interesting topic to look at. They have been proven to achieve channel capacity (although not as simply as with polar codes) and have been shown to have smaller finite length capacity gap than polar codes.

Another view on iterative turbo and LDPC codes

Codes are constructed so that the relationship between their bits (the structure of their redundancy) is locally simple, admitting simple local decoding. The local descriptions of the codes are interconnected in a complex manner (e.g., in pseudorandom), introducing long-range relationships between code bits.

Relatively high global description complexity is thereby introduced in the interconnection between the simple local structures
- Turbo codes: short convolutional codes are the simple local codes, and interleaver introduces the complex interconnections
- LDPC codes: parity check equations are the simple local codes, and the Tanner graph of the connections between variable nodes and parity check nodes represents the complex interconnections between parity check nodes

Iterative decoding proceeds by performing simple local decoding and then exchanging the results, passing messages between locales across the “complex” interconnections.

Thus, description complexity is introduced without adding computational complexity per se, but rather as wiring, routing, or design complexity.*

The implementation question is how to handle this routing complexity (?).

---

Polar codes were introduced by Erdal Arikan in 2009*. They are based on the channel polarization phenomenon. The encoding matrix can be considered as a series of synthetical channels, and in these channels the mutual information tends to polarize into 1 or 0, when the code block size becomes large.

Polar codes were the first provably capacity achieving codes (basic turbo and LDPC are capacity approaching) which have deterministic construction. Also spatially coupled LDPC have been shown to be capacity achieving and also Reed-Muller codes have been proven to be capacity achieving in certain channels. Reed-Muller and Polar codes are very closely related. The difference is in the way the bit indices used for transmission are selected.

Polar codes are linear block codes with conceptually simple construction. The main drawback of original polar codes is that in their design a design-SNR is used, which relates the optimal code structure to the neighborhood of a specific SNR. Thus, operating at different SNR points requires different code designs!

Differences compared to other codes:
- Turbo codes: parallel concatenation of state machines driven by input bits
- LDPC: linear block code with sparse parity-check matrix

The polar codes have been approved as the 5G control channel codes for DL and UL (currently working assumption)

Research question: How to improve performance and reduce detection complexity of short polar codes (5G control channel)

From [http://www.polarcodes.com/](http://www.polarcodes.com/) an online-tutorial and downloadable matlab implementation of a polar codec can be found.

Polar codes: encoding

Let us now consider an 8 bit example. The information bits, $d_i$, are indexed in the decoding order. Well get to this in the next page. The stages illustrated in the figure correspond to the stages/levels in the binary tree that is used in decoding.

In the encoding process, the XOR-operation reduces the capacity of a particular channel. But, the main drawback of basic polar codes is that ordering of the input bits in terms of capacity depends on the channel! So, the codes are not universal. This means, e.g., that for a code of a certain length, different set of frozen bits is optimal at different SNR. Recently, there has been proposals to overcome this problem, e.g., by concatenating polar codes with LDPC codes.*

Systematic polar codes can be constructed, e.g., by placing the information bits into data bearing $x_i$, instead of $d_i$.**

---

Polar codes: successive cancellation decoding

The decoding of polar codes was originally based on successive cancellation decoding (SCD). Although the SCD is a very simple and error prone algorithm, it is a provably capacity achieving solution for long codes. In SCD, we assume that when decoding a bit $d_i$, bits $d_k$, $k < i$ are known and bits $d_l$, $l > i$, are completely undetermined ($p(d_l = 0) = p(d_l = 1) = 0.5$). Recently, efficient list detection algorithms have been proposed to improve the performance with short block lengths*.

Let us now decode first two bits (in conceptual level) for the given 8-bit example code. First bit $d_0$

Here, on the left hand side the full encoding structure is shown, on the right hand side the paths participating on the decoding of $d_0$ are shown. In the bottom the corresponding binary tree is shown.

---


* I. Tal and A. Vardy, "List Decoding of Polar Codes," 2015
Polar codes: successive cancellation decoding

And then bit $d_1$

And so on for all the bits. In the next page all of the binary trees related to different bit indices are illustrated. The order of the binary trees in terms of mutual information is not completely clear and depends on the channel.

Now, freezing the “bad” channels allows us to transmit information only through the “good” channels.
Polar codes: Performance under Successive Cancellation Decoding (SCD)

Polar code performance* with SCD and $R = 1/2$

[Graph showing performance over blocklength vs. BER]

and for $R = 9/10$

[Graph showing performance over blocklength vs. BER]

SCD does not perform very good with high code rates. Improved performance can be achieved by more complex decoders, e.g., list decoder.

* www.polarcodes.com

Polar codes, some conclusions

Polar codes are new, provable capacity achieving linear block codes which have gained lots of interest and have been accepted (currently working assumption) as the DL and UL control channel codes in 5G eMBB physical layer.

Polar codes are based on the channel polarization phenomenon, which drives the virtual channels to either capacity $\sim 0$ or $\sim 1$. Then, intuitively, information bits are transmitted only in those virtual channels which have capacity $\sim 1$, and those virtual channels which have capacity $\sim 0$ are frozen. Frozen bits are always zeros and known by the TX and RX.

The main benefits are that they are provably capacity achieving codes with SCD. The main drawbacks of the basic polar code design are the channel dependent frozen bit indices, slow decoding with SCD, performance with short block lengths, and detection complexity with more sophisticated decoders (e.g., list decoders). Polar codes can be converted to good universal codes by concatenating them, e.g., LDPC code (as inner code) or shift registers to generate additional parity bits on the code word.

The main open research problem is the performance with short code block lengths? How to improve the performance and how to efficiently decode the code with improved performance. The good thing is that we still have some problems to solve =)
TRELLIS CODES

The basic block codes and convolutional codes create the redundancy to transmit signal by adding the amount of transmitted bits:
- code rate \( r_c = k/n \) => the amount of bits is \( n/k \) times that of an uncoded system.

Thus, if the symbol alphabet size is independent of code rate, and we want to keep the information bit rate unchanged, the coded system needs \( n/k \) times bigger bandwidth (compared to uncoded system):
- this implies also bigger noise power in the receiver which eats out part of the true Coding Gain, like we have analyzed earlier.

In trellis coding or coded modulation, the idea is to build the redundancy such that symbol alphabet design depends on the code rate.

If the symbol alphabet size in uncoded reference system is \( 2^k \) and code rate to be used is \( r_c = k/n \), then we deploy an alphabet size \( 2^n \)

In this manner, the information bit rate (over a given bandwidth) is independent of the code rate, and we can capitalize fully the Coding Gain.

On the other hand, in Coding Gain analysis, bigger noise sensitivity of larger symbol alphabet needs to be taken into account (no free lunch, sorry).

Underlying code can be either a block code or convolutional code, but convolutional codes are typically deployed. In this particular case, the overall method is called trellis coding.

Concept of Coded Modulation

In the following, we sketch the basics of coded modulation and trellis coding in a compact manner.

The idea is to design code rate and symbol alphabet size together.

The underlying motivation comes from the information theoretic basis (capacity analysis addressed in the very early parts of the course):

Example: target is to communicate at rate 3 information bits per channel use:
- uncoded 8-level modulation:
  - zero error probability needs infinite SNR
  - error probability \( 10^{-5} \) needs SNR around \( \sim 26 \) dB
- 16-level modulation
  - Should in theory be possible with zero error probability at SNR around \( \sim 18 \) dB

=> coded modulation Coding Gain in theory around 8 dB
Coded Modulation (cont’d)

Continuing the previous example – with bigger modulation size than 16, the additional gain is then only max around 1dB (why?).

Thus in practice, by doubling the constellation size, we can capitalize most of the available gain.

At bit level processing, doubling the constellation size (“$M \Rightarrow 2M$”) enables adding one coded bit per symbol:

$$\log_2(2M) = \log_2(M) + 1$$

Thus code rates of the form

$$r_c = k/n = k/(k+1)$$

are in practice a well-working (yet economical) solution. This means code rates of the form

$$r_c = 1/2, \; 2/3, \; 3/4, \; 4/5, \; 5/6, \; ...$$

Example Trellis Code

Here we use the previous rate $r_c = 1/2$ convolutional code as the example code, and the 4-level symbol alphabet is here QPSK (alphabet $\{+a, +ja, -a, -ja\}$).

As a whole, we are thus using QPSK at information rate of 1 information bit per symbol.
Decoding of Trellis Codes

Stemming from the memory of underlying convolutional code, also trellis codes have memory. Thus decoding can be built on sequence detection principle implemented using Viterbi-algorithm.

In case of soft decoding, the only difference to actual convolutional codes is the increased symbol alphabet size and thus increased noise sensitivity.

Below example of one stage of trellis for
- 1/2 convolutional code with binary alphabet (left)
- 1/2 trellis code with QPSK alphabet (right)

As learned earlier, the performance of Viterbi-decoding (Coding Gain) depends essentially on minimum Euclidian distance of coded symbol streams:

\[
P[\text{bit error, coded}] \simeq Q\left[\frac{d_{E,\min}}{2\sigma_c}\right]
\]

Example will follow.

Performance Example

Figure below illustrates an example of minimum distance error event when the true (correct) path is all zeros (at bit level):

The corresponding minimum distance is (alphabet is here \{+a_c, ja_c, a_c, -ja_c\})

\[
d_{E,\min} = \sqrt{|a_c + a_c|^2 + |a_c - ja_c|^2 + |a_c + a_c|^2} = \sqrt{10}a_c
\]

That this really is the minimum distance is confirmed by the facts that
- When two paths diverge, the first branch distance is always \(2a_c\)
- After diverging, the second branches have always distance \(\sqrt{2}a_c\)
- When two paths merge, last branch distance (before merging) is \(2a_c\)

Thus the error probability is

\[
P[\text{bit error, coded}] \simeq Q\left[\frac{\sqrt{10}a_c}{2\sigma_c}\right] = Q\left[\frac{10a_c^2}{4\sigma_c^2}\right] = Q\left[\frac{10}{4} SNR_c\right]
\]
Performance Example (cont’d)

Uncoded reference case is binary transmission with alphabet $\pm a_u$. The reference bit error rate is then

$$P[\text{bit error, uncoded}] = Q\left(\frac{a_u}{\sigma_u}\right) = Q\left(\frac{a_u^2}{\sigma_u^2}\right) = Q\left(\sqrt{SNR_u}\right)$$

Thus the coded and uncoded transmission chains have identical error rates when

$$SNR_u = \frac{10}{4}SNR_c$$

Thus the Coding Gain from SNR perspective is directly

$$\text{Coding Gain} = 10 \log_{10}\left(\frac{10}{4}\right) \approx 4 \text{ dB}$$

This is also the true gain from the needed received signal power point of view, since the bandwidth (and thus noise power) is now fixed.

Notice also that similar gain of 4dB can be achieved with the same convolutional code, in the classical sense with binary modulation and double bandwidth (compared to uncoded case).

Thus the coding gain with trellis code is the same, and both systems have also the same information bit rate. The difference is, however, that the trellis coded system is using only half the bandwidth, and thus has double spectral efficiency compared to traditional case.

Trellis Code Performance, more generally

In general, the achievable gain depends on the number of states in the trellis diagram (essentially the amount of memory).

Following table illustrates the orders of magnitude of the achievable Coding Gain vs. the number of states:

<table>
<thead>
<tr>
<th>#states</th>
<th>coding gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3-4dB</td>
</tr>
<tr>
<td>8</td>
<td>4-5dB</td>
</tr>
<tr>
<td>16</td>
<td>close to 5dB</td>
</tr>
<tr>
<td>128</td>
<td>close to 6dB</td>
</tr>
</tbody>
</table>

Error probability, and thus the performance in terms of Coding Gain, are always dominated by the minimum distance error events.

In error probability analysis, care must be taken though. This is because even if the underlying convolutional code is linear, the trellis code may be nonlinear.

For nonlinear codes, in general, the minimum distances of all individual paths should be checked separately. This can be done systematically in practice using, e.g., the Viterbi-algorithm (by feeding in the true coded sequences, one by one, and following the second closest path).
Scrambling is a method to “randomize” the transmitted bit sequence, either the uncoded or coded one. This helps e.g. different synchronization algorithms in the receiver and also reduces intercell interference in cellular systems (when different cells use different scrambling codes).

Scrambling does not increase the amount of transmitted bits and thus does not add any redundancy - it is in principle a one-to-one mapping between input and output bit sequences.

Scrambling is used in all communication systems, and thus here we sketch the basics of it.

Example: in LTE cellular network, scrambling is applied to coded bits, i.e., at the output of the channel coder prior to the symbol mapper. The exact details vary though, a lot, depending on which of the channels in the LTE system we talk about (shared data channel, control channels, etc).

On transmitter side, the bit sequence $b_k$ is summed with another bit sequence $x_k$, the so called scrambling sequence. All arithmetics in modulo 2, thus the scrambled sequence is another bit sequence.

On receiver side, the scrambling is removed by adding again the same sequence, again in modulo 2, to the received bits. This requires proper frame synchronization.

The scrambling sequences that are commonly used are deterministic known sequences (such that both TX and RX knows them) but that are pseudo-random like in order to carry out the randomization efficiently.

Commonly, these sequences are periodic (i.e., they repeat at certain cycle) but the period length is very long, e.g., $2^{31}$ bits.

Such sequences can be easily generated using feedback shift-registers.

Some more details in the following.
**Pseudo-random sequences**

Pseudo-random sequences used in scrambling can be generated with the following type of feedback shift-registers:

Here \( D \) denotes a unit delay, and all sums are modulo 2. The initial state of the shift-register can be anything other than all-zeros (why ?). The binary feedback coefficients \( h_1, \ldots, h_n \) define a so called generator polynomial (or vice versa).

With \( n \) delay elements, the maximum period length of the sequence is \( 2^n - 1 \) (why ?) This type of so called maximum length sequences can be generated with certain feedback coefficients (generator polynomials) that are well known in applied math:

<table>
<thead>
<tr>
<th>order ( n )</th>
<th>coefficients in octal form</th>
<th>order ( n )</th>
<th>coefficients in octal form</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>19</td>
<td>20000047</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>20</td>
<td>4000011</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>21</td>
<td>10000005</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>22</td>
<td>20000003</td>
</tr>
<tr>
<td>6</td>
<td>103</td>
<td>23</td>
<td>40000041</td>
</tr>
<tr>
<td>7</td>
<td>211</td>
<td>24</td>
<td>100000207</td>
</tr>
<tr>
<td>8</td>
<td>435</td>
<td>25</td>
<td>200000011</td>
</tr>
<tr>
<td>9</td>
<td>1021</td>
<td>26</td>
<td>400000107</td>
</tr>
<tr>
<td>10</td>
<td>2011</td>
<td>27</td>
<td>1000000047</td>
</tr>
<tr>
<td>11</td>
<td>4005</td>
<td>28</td>
<td>2000000011</td>
</tr>
<tr>
<td>12</td>
<td>10123</td>
<td>29</td>
<td>4000000005</td>
</tr>
<tr>
<td>13</td>
<td>20033</td>
<td>30</td>
<td>10040000007</td>
</tr>
<tr>
<td>14</td>
<td>42103</td>
<td>31</td>
<td>20000000011</td>
</tr>
<tr>
<td>15</td>
<td>100003</td>
<td>32</td>
<td>40020000007</td>
</tr>
<tr>
<td>16</td>
<td>210013</td>
<td>33</td>
<td>100000020001</td>
</tr>
<tr>
<td>17</td>
<td>400011</td>
<td>34</td>
<td>201000000007</td>
</tr>
<tr>
<td>18</td>
<td>1000201</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table on the previous page shows the generator polynomials for shift register lengths of \( n = 2 \cdots 34 \).

The generator polynomials are given in octal form - when converted to binary form the describe the generator polynomial \( h(D) \) and thus thereon the feedback coefficients of the shift-register structure.

**Example:** \( n = 15: 1 \ 0 \ 0 \ 0 \ 0 \ 3 \Rightarrow 001 \ 000 \ 000 \ 000 \ 000 \ 011 \) which corresponds to polynomial

\[
h(D) = 1 \oplus D^{14} \oplus D^{15}
\]

Example scrambling polynomials from commercial systems:

\[
\begin{align*}
\text{DVB} & : h(D) = 1 \oplus D^{14} \oplus D^{15} \\
\text{VDSL} & : h(D) = 1 \oplus D^{18} \oplus D^{23}
\end{align*}
\]
Self-Synchronized Scrambler

Compared to basic and simple frame-synchronized scrambler concept, this approach does not basically require frame synchronization.

If the bits $c_k$ are identical at both ends, original bits are recovered.

On the other hand, if and when there are bit errors in the system, this structure has error accumulation/propagation problems which are avoided in the basic frame synchronized scrambling approach.

Basic frame synchronized scrambler is the de-facto solution.

summary: quite a bit processing involved, or what do you think ? ☺
GOOD FOLLOW-UP COURSES (examples, there are also more)

ELT-43106 Multicarrier and Multiantenna Techniques
- all the essentials and more regarding multicarrier/OFDM systems and associated signal processing in transmitters and receivers
- all the essentials regarding multiantenna/MIMO and associated signal processing in transmitters and receivers
- both are key technologies in modern wireless systems and networks
- lectured in spring terms

ELT-45306 Advanced Course on Wireless Communications
- extensive course on LTE/LTE-Advanced mobile cellular radio network and involved radio protocols
- how multicarrier and multiantenna technologies, coding methods, etc are applied in LTE/LTE-Advanced
- LTE/LTE-Advanced network radio protocols, system structure, radio resource management principles, etc
- lectured in spring terms

ELT-43406 5G Mobile Communications
- more applied course on wireless communications solutions for 5G mobile networks and applications
- lectured in spring terms

ELT-43456 Internet-of-Things Wireless Communications
- more applied course on wireless communications solutions for IoT systems and applications
- both cellular and ISM-band based techniques covered
- lectured in spring terms, first implementation on spring 2018

COMPLEMENTARY MATERIALS

ADVANCED CHANNEL CODING SCHEMES

- for those who are interested in further details
- builds on the basics of Turbo, LDPC & Polar codes described within the actual lecture notes
TURBO CODES & DECODING

The MAP algorithm used to decode turbo codes is typically so called BCJR algorithm*. It defines the symbol-by-symbol (bit-wise in the case of binary code) MAP decoding as

\[
L(u_i) = \ln \frac{P(u_i = 0, y)}{P(u_i = 1, y)} = \ln \sum_{s' \in S} P(s', s, y) = \ln \sum_{s' \in S} P(s', s, y_{k-1}, y_k)
\]

which allows efficient calculation of the log-likelihood ratio (LLR) based on the convolutional code’s Trellis diagram.


After some manipulation,

\[
L(u_i) = \ln \sum_{s' \in S} \alpha_{i-1}(s') \cdot \gamma_i(s', s) \cdot \beta_i(s)
\]

where

- \( \alpha_{i-1} \) is the Forward state probability for state \( s' \) at time instant \( i \)
- \( \gamma_i(s', s) \) is the Transition probability from state \( s' \) to state \( s \) at time instant \( i \). This is where the a priori information is brought to the equation!
- \( \beta_i(s) \) is the Backward probability for state \( s \) at time instant \( i \)

The beauty of this equation is that forward and backward probabilities can be calculated recursively, similar to Viterbi-algorithm. The forward probability is evaluated by starting from the left hand side of the code trellis and the backward probability by starting from the right hand side of the code trellis.
Turbo codes & decoding

The decoding is typically done with LLR values, because using probabilities leads to numerical instability. The log-domain transition variable is defined for the first component decoder as

\[
\bar{\gamma}(s', s) = \ln \left( \gamma(s', s) \right) = \ln P(y_i | s', s) + \ln P(s | s')
\]

\[
= C - \frac{1}{2\sigma^2} (y_i - c_{i,s})^2 + \ln P(u_i = u(s', s)) + \ln P(u_i = u(s', s)) + \ldots
\]

\[
= u_i y_{i,s} / \sigma_{iu}^2 + p_i y_{i,s} / \sigma_{iu}^2 + u_i L_i(u_i) / 2
\]

the forward variable as

\[
\bar{\alpha}_i(s) = \ln \alpha_i(s) = \ln \left( \sum_s \gamma_i(s', s) \alpha_{i,s}(s') \right) \ln \left( \sum_s \exp(\bar{\gamma}(s', s) + \bar{\alpha}_{i,s}(s')) \right)
\]

and the backward variable as

\[
\bar{\beta}_i(s') = \ln \beta_i(s') = \ln \left( \sum_s \gamma_i(s', s) \beta_{i,s}(s) \right) \ln \left( \sum_s \exp(\bar{\gamma}(s', s) + \bar{\beta}_{i,s}(s)) \right)
\]

with the initialization parameters

\[
\bar{\alpha}_i(s') = \begin{cases}
0, & \text{if } s' = 0 \\
-\infty, & \text{if } s' \neq 0
\end{cases}
\]

\[
\bar{\beta}_i(s) = \begin{cases}
0, & \text{if } s = 0 \\
-\infty, & \text{if } s \neq 0
\end{cases}
\]

\[
\bar{\beta}_i(s) = \text{constant}
\]

Terminated code Non-terminated code

Still, we have a logarithmic sum terms which are not easily computed!

- Jacobian algorithm is commonly used to simplify the logarithmic sums

\[
\ln(e^{x_1} + e^{x_2}) = \max^*(x_1, x_2)
\]

\[
= \max(x_1, x_2) + \ln(1 + e^{h(x_2)}) = \max(x_1, x_2) + f_c(-|x_1 - x_2|)
\]

The function \(\max^*(x_1, x_2)\) can be approximated in several different ways

- Look up table for \(f_c(-|x_1 - x_2|)\) (e.g. 8 different values sufficient*)
- (Piecewise) linear approximation of \(f_c(-|x_1 - x_2|)\)
- Constant approximation of \(f_c(-|x_1 - x_2|)\)
- Weighted max \((k \max(x_1, x_2) \approx \max^*(x_1, x_2), \text{where } k \approx 0.75)\)
- Plain vanilla max (which relates to SOVA!)

Turbo Codes: EXIT charts intro

The BER curve of a turbo code (or LDPC) typically shows two different regions:

- **Waterfall-region**: At low to medium SNR the iterative decoding is extremely efficient and BER drops fast as a function of SNR.
- **Error floor region**: At high SNR the decoding converges in few iterations and BER improves slowly as a function of SNR.

How to understand this behavior? Somehow related to the extrinsic information exchanged between decoders.

EXIT-charts* are analysis of iterative process by semi-analytic approach:

- Determine analytically mutual information between information bits and a-priori information.
- Determine by simulation mutual information between information bits and extrinsic information for a specific a-priori information level.
- Picture relationship between both mutual informations.
- Combine diagrams of both contributing decoders into one chart.

First, let's look at the transfer characteristics function for a (37,23f)-RSCC with rate R=2/3. The illustrated results show how much extrinsic information, $I_e$, is observed in the component decoder output with given a priori information, $I_a$, in the component decoder input.

Below an EXIT-chart with snapshot decoding trajectory at $E_b/N_0 = 1$ dB is given. The second trajectory (red one below) is obtained by taking one transfer characteristic function and changing the axis $I_a$ and $I_e$. Then, both functions have been plotted to the same figure. Changing the axis of the second transfer characteristic function is now showing how the extrinsic information of the first component decoder is transformed to a priori information by the second component decoder, and how the extrinsic information from the second component decoder is transformed into a priori information for the first component decoder.

In the example figure, the "staircase" trajectory is an example of the extrinsic information evolution from a single simulation with maximum of 10 iterations. It can be observed that the transfer characteristic functions accurately bound the information transfer behavior, and provide an accurate model for an iterative decoder performance.

* S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," 2001
**Turbo Codes: EXIT charts intro**

Pinch-off SNR is the SNR value at which the two transfer characteristic functions start to overlap. Because at this point the path for mutual information to grow towards unity is closed, it gives an accurate SNR estimate for the start of the waterfall-region.

EXIT-charts provide an important tool to design, analyze, and understand iterative codes and systems, estimate the performance without extensive simulations, and provide the pinch-off SNR.

**LDPC Codes & L-Algebra for Decoding**

Here the log-domain algebra commonly used with LDPC codes is reviewed. This is more or less the same as LLR used with turbo codes, but with LDPC codes we concentrate on the LLR related to modulo-2 sum of certain information bits.

**Basic properties of LLR/L-values**

- Sign of the value equals to the hard decision
- Magnitude indicates the reliability of the decision
- For an uncoded transmission the LLR consists of two components
  - \( L(y|x) \) depends on the received signal \( y \) and therefore on the channel statistics
  - \( L_d(x) \) represents the a priori information about symbol \( x \)
- LLR is a sufficient statistics
  - Computed function that retains all information needed for making an optimal decision

Parity bits are generated by modulo-2 sums of certain information bits. How can we calculate the L-value of a parity bit \( p = u_1 \oplus u_2 \)?

\[
L(u_1 \oplus u_2) = \ln \frac{P[u_1 \oplus u_2 = 0]}{P[u_1 \oplus u_2 = 1]} = \ln \frac{P[x_1 \cdot x_2 = +1]}{P[x_1 \cdot x_2 = -1]} = L(x_1 \cdot x_2)
\]

\[
L(x_1 \cdot x_2) = \ln \frac{\exp(L(x_1) + L(x_2)) + 1}{\exp(L(x_1)) + \exp(L(x_2))} = 2 \tanh^{-1} \left[ \tanh(L(x_1)/2) \cdot \tanh(L(x_2)/2) \right] = L(x_1) + L(x_2)
\]

Thus, the likelihood value of the modulo-2 sum of two statistically independent random variables is defined based on the box-plus operator with rules:

\[
L(x) \Box 0 = 0, \quad L(x) \Box \infty = L(x), \quad L(x) \Box -\infty = -L(x)
\]
LDPC codes & decoding

Here a short description of the logarithmic sum-product algorithm (SPA) is given. Some notation for defining the SPA:

- $V_j$ = v-nodes connected to c-node $c_j$ = neighborhood of $c_j$
- $V_j/i$ = {v-nodes connected to c-node $c_j$ \{v-node $v_i$\} = v-nodes connected to c-node $c_j$ except $v_i$
- $C_i$ = c-nodes connected to v-node $v_i$ = neighborhood of $v_i$
- $C_i/j$ = {c-nodes connected to v-node $v_i$ \{c-node $c_j$\} = c-nodes connected to v-node $v_i$ except $c_j$
- $P_i = P(v_i = 1 | y_i)$ = probability of v-node $v_i$ having value 1 given the channel observation $y_i$
- $q_{ij}(b) = P(v_i = b | S_i, y_i, M_c(\sim j))$, where $b = \{0, 1\}$ = probability that v-node $v_i$ having a bit value $b$ satisfies all check equations in set $C_i/j$
- $r_{ij}(b) = P(\text{check equation } c_j \text{ is satisfied } | v_i = b, M_c(\sim i))$, where $b = \{0, 1\}$ = probability that check equation related to c-node $c_j$ is satisfied if v-node $v_i$ has value $b$. (Meaning that from all the other nodes in set $V_j/i$ we obtain an even or odd number of ones, for $v_i = 0$ or $v_i = 1$, respectively)
- $Q_i(b) = \text{probability that v-node } v_i \text{ has value } b, \text{ where } b = \{0, 1\}$

Note that the messages $q_{ij}(b)$, while interpreted as probabilities here, are random variables (rv’s) as they are functions of the rv’s $y_i$ and other messages which are themselves rv’s. Similarly, by the virtue of the message passing algorithm, the messages $r_{ij}(b)$ are rv’s

Let us now define the related L-values

$$L(v_i) = \ln \left( \frac{P(v_i = 0 | y_i)}{P(v_i = 1 | y_i)} \right), L(r_{ij}) = \ln \left( \frac{r_{ij} = 0}{r_{ij} = 1} \right),$$

$$L(q_{ij}) = \ln \left( \frac{q_{ij} = 0}{q_{ij} = 1} \right), L(Q_i) = \ln \left( \frac{Q_i = 0}{Q_i = 1} \right)$$

Summary of the Log-Domain SPA Decoder (skipping some derivations):

1) For $i = 0, 1, ..., n - 1$, initialize $L(q_{ij}) = L(v_i) = 2y_i/\sigma_i^2$ for all $i, j$ for which $h_{ji} = 1$ (assuming memoryless Gaussian channel)
2) Update $L(r_{ij}) = \sum_{i \in V_j/i} L(q_{ij})$
3) Update $L(q_{ij}) = L(v_i) + \sum_{j \in C_i/j} L(r_{ij})$
4) Update $L(Q_i) = L(v_i) + \sum_{j \in C_i} L(r_{ij})$
5) For $i = 0, 1, ..., n - 1$, set $\hat{v} = 1$ if $L(Q_i) < 0$, otherwise $\hat{v} = 0$
6) If $\hat{v}H^T = 0$ or the number of iterations equals the maximum limit, stop; else go to Step 2)

Notice that for linear codes, the decoding algorithm is able to detect an uncorrect codeword with near-unity (very high) probability! In other words, LDPC codes have an inherent property to detect correctly received code word and stop iterating. (How about turbo codes?)
LDPC codes & decoding

We notice that assuming independent random variables, we can define the $L(r_{ij})$ with the earlier mentioned box-plus operator. As with the Jacobian function used with turbo codes, this operator can be approximated in several ways. Actually, the box-plus operator and Jacobian function can be related by

$$L_1 \boxplus L_2 = \max \{ 0, L_1 + L_2 \} - \max \{ L_1, L_2 \}$$

Another way to look at the product of complex tanh-functions, is to factor

$$L(\alpha_{ij}) = \alpha_{ij} \beta_{ij}$$

into its sign, $\alpha_{ij}$, and magnitude $\beta_{ij}$, so that we can rewrite $L(r_{ij})$ as

$$L(r_{ij}) = \sum_{i \in V_{ij}} L(q_{ij}) = 2 \tanh^{-1} \left[ \prod_{i \in V_{ij}} \tanh \left( \frac{1}{2} L(q_{ij}) \right) \right]$$

$$L(r_{ij}) = 2 \tanh^{-1} \left[ \prod_{i \in V_{ij}} \tanh \left( \frac{1}{2} \alpha_{ij} \beta_{ij} \right) \right] = \prod_{i \in V_{ij}} \alpha_{ij} 2 \tanh^{-1} \left[ \prod_{i \in V_{ij}} \tanh \left( \frac{1}{2} \beta_{ij} \right) \right]$$

Let us now define

$$\phi(x) = - \log \left[ \tanh \left( \frac{x}{2} \right) \right] = \log \left( \frac{e^x + 1}{e^x - 1} \right)$$

$$\phi^{-1}(x) = \phi(x), \text{ if } x > 0$$

We finally obtain

$$L(r_{ij}) = \prod_{i \in V_{ij}} \alpha_{ij} \phi \left[ \sum_{i \in V_{ij}} \phi \left( \beta_{ij} \right) \right]$$


The function $\phi(x)$ can be now similarly approximated as proposed with Jacobian algorithm (look-up table, piecewise linear, constant, etc.).

So we started from the product of probabilities, converted it into product of complex tanh-functions (box-plus), and from these we obtained a form with product of signs times a function that can be implemented through a look-up table. This is quite important from the implementation point of view.

Simplest form of the box-plus operation approximation leads to so called min-sum decoder. Note from the shape of $\phi(x)$ that the smallest $\beta_{ij}$ term dominates in the summation in the definition of $L(r_{ij})$, so that

$$\phi \left( \sum_{i} \phi(\beta_{ij}) \right) \sim \phi \left( \phi(\min \beta_{ij}) \right) = \min \beta_{ij}$$

This solution is comparable with the max-log-MAP simplification used with turbo codes.
Designing irregular codes is to find parity-check matrices having sufficient structure for efficient encoding, good performance, and small number of edges. This field of research requires its own lecture, and has not been discussed here in detail. Shortly:

- Irregular LDPC codes optimized via density evolution or extrinsic information transfer (EXIT) analysis
  - Provides probability density functions describing the distribution of degrees of variable and check nodes
- Different design methods (placing the edges between \( v \)-nodes and \( c \)-nodes) include
  - progressive edge growth (PEG), which aims to maximize the local cycle lengths
  - approximate cycle extrinsic message degree (ACE), which maximizes the extrinsic information flowing into stopping sets
  - protograph design, where protograph defines the properties of the final LDPC code that is a combination of protographs with interleaved connections between \( v \)-nodes and \( c \)-nodes of different protographs (=> spatially coupled LDPC codes)
- Different design methods are not exclusive and can be used together in design and analysis

This field is developing and still under active research. Interested reader may use the given key words to find out more on this topic.

**POLAR CODES & ENCODING**

Let’s have another way of defining polar codes. (Just for the fun of it!)

An \((N, K, I)\), where \(N = 2^m\) is the code length in bits and \(K\) is the information length in bits, and \(I\) is the index set of data bits. The complementary set \(I^C\) is called frozen bit indices.

The kernel of the polar code is defined as \(F^\otimes n = F \otimes F \otimes \ldots F \otimes F\) \((n\ times)\), where \(\otimes\) is the Kronecker product operator.

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

Now the code word is generated by \(x = F^\otimes n d\), where \(d_{i,c} = 0\) and \(d_i = u\). Example: \(x = F^\otimes 2 d, N = 4, K = 3, I = \{0,2,3\}, I^C = \{1\}\)

\[
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 = u_0 \\ d_1 = 0 \\ d_2 = u_1 \\ d_3 = u_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = Gu
\]

Note that frozen zeros can be ignored because they do not affect the modulo-2 sum (XOR). The efficient implementation of polar codes is based on a XOR network with complexity \(O(N \log_2 N)\). One kernel can be represented as

And for the example the XOR network corresponds to
**Polar codes & decoding**

How are the probabilities evaluated in the XOR and repetition nodes? Let’s look at the information flow through a kernel. $L_0$ and $L_1$ are now LLR values related to $d_0$ and $d_1$, respectively. This process is then easily generalized to solve all the bit values in the polar code.

Now we have two likelihood operations in use. $f(L_0, L_1)$ is easy, because we remember the box-plus operator. The second term is repetition, $g(L_0, L_1)$, because you have the same information in two branches, but the sign of the likelihood $L_0$ depends on the bit decision made from $f(L_0, L_1)$ due to the nature of the XOR operation. We can write

$$
\begin{pmatrix}
    f(L_0, L_1) \\
g(L_0, L_1)
\end{pmatrix}
= \begin{pmatrix}
    L_0 \boxplus L_1 \\
    (L_0 \boxplus L_{u_0|L_0,L_1}) + L_1
\end{pmatrix}
$$

where $L_{u_0|L_0,L_1} = \begin{cases} 
-\infty, u_0 = 1 \\
\infty, u_0 = 0 
\end{cases}$

The decoding order of the kernel (and a larger code) relates to the mutual information in the different virtual channels. The total mutual information can be written with the help of chain rule decomposition as

$$
I(D_0, D_1; Y_0, Y_1) = I(D_0; Y_0, Y_1) + I(D_1; Y_0, Y_1|D_0) = 2I(X; Y)
$$

which implies that one can also achieve the rate $2I(X; Y)$ using two steps. First, information transmitted through the virtual channel $W^- : D_0 \rightarrow (Y_0, Y_1)$ is decoded to $\tilde{D}_0$. Then, information transmitted through the virtual channel $W^+ : D_1 \rightarrow (Y_0, Y_1, D_0)$ is decoded based on the side information $\tilde{D}_0$. 
