Background

The material of the last two lectures of the course are based on

[**MF**] J. C. Moreira and P. G. Farrell, Essentials of Error-Control Coding, Wiley, Chichester, 2006, pp. 209–325.

- Traditional paradigm: Try to construct codes with large minimum distance.
- Shannon: Random codes are good.
- Modern paradigm: "Distance isn't everything". Two code types with good practical performance are *turbo* and *LDPC* codes. Iterative decoding.

©Patric Östergård

Turbo Codes

Turbo(charger): A supercharger driven by a turbine powered by the engine's exhaust gases.

The term *turbo* in the coding context origins from some similarity between decoding and the turbo engine principle (and not from the fact that turbo can be imagined as something better and/or faster, although these codes indeed have good practical performance).

Turbo codes were discovered in 1993 by Berrou, Glavieux and Thitimajshima.

©Patric Östergård

Turbo Encoder

A basic turbo encoder is a rate-1/3 systematic encoder consisting of three parallel parts whose output is [MF, Fig. 7.1]

1. the data stream unaltered,
2. data from a convolutional encoder,
3. data from a convolutional encoder combined with a random interleaver.

Puncturing (which is usually not applied to the systematic bits) can be used to obtain higher rates than 1/3.

©Patric Östergård

Turbo Decoder

Turbo codes are decoded in an iterative soft-decision decoding process. Two different decoders—corresponding to the two different encoders—get progressively better estimates of the message bits through an iterative exchange of information. More precisely, each of the decoders obtain a new estimate based on

- the systematic bits,
- the parity bits (produced by the corresponding encoder),
- the previous estimate of the other decoder.

This scheme is described in [MF, Fig. 7.2].

©Patric Östergård

Log Likelihood Ratio

The estimates are usually communicated from one decoder to the other in the form of *log likelihood ratio* (LLR).

Instead of elements from $\{0, 1\}$, it is often more convenient to consider elements from $\{-1, 1\}$. The LLR is defined as

$$L(b_i) = \ln\left(\frac{P(b_i = +1)}{P(b_i = -1)}\right).$$

The sign of the LLR can be used as the hard decision of the estimate; the absolute value gives the reliability of the estimate.

©Patric Östergård

BCJR Algorithm

The *BCJR algorithm* (Bahl-Cocke-Jelinek-Raviv) determines an estimate for a given sequence element by observing the output sequence of a discrete memoryless channel that corresponds to a given input sequence.

The BCJR algorithm is the core part of iterative decoding of turbo codes. The BCJR and other related algorithm are considered in detail in the course S-72.3280 Advanced Radio Transmission Methods (from 2008: S-72.3281 Advanced Transmission Methods).

©Patric Östergård

Iterative Decoding

A scheme for iterative decoding of turbo codes is shown in [MF, Fig. 7.11]. LLR estimates in an example case ([MF, Example 7.3]) are presented in [MF. Tables 7.8–7.14].

A priori information: Information provided by the other encoder. (In the beginning $P(b_i = +1) = P(b_i = -1) = 0.5$ so the a priori LLR is 0 for the first encoder in the first iteration.)

Extrinsic information: The component of the generated reliability value that depends on redundant information introduced by the considered constituent code.

©Patric Östergård

Interleavers for Turbo Codes

Interleavers, which permute positions in a stream of symbols, are useful for several purposes in coding, for example, to

- combat burst errors by distributing adjacent symbols (and thereby errors) among many words;
- obtain statistically independent sequences of symbols for turbo encoders.

Major types are *block*, *convolutional*, *random*, and *linear* interleavers.

©Patric Östergård

Block Interleavers

In a block interleaver, the symbols are written rowwise into an $M \times N$ matrix and read columnwise. With a table

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

the symbols are output in the order $1 \to 6 \to 11 \to 16 \to 2 \to \cdots$.

Design rule: Let $M = N$ and let M and N be odd.

©Patric Östergård

Convolutional Interleavers

A convolutional interleaver is implemented using N shift registers, of length $0, L, 2L, \ldots, (N-1)L$ as in [MF, Fig. 7.16].

With $N = 3$ and $L = 1$, the symbols are output in the order $0 \to -2 \to -4 \to 3 \to 1 \to -1 \to 6 \to 4 \to \cdots$.

Easy implementation is a strength of convolutional interleavers. Note, however, that no block of consecutive symbols is mapped to consecutive symbols.

©Patric Östergård

Random Interleavers

A random interleaver gives a random permutation of a block of symbols. The permutation of a (pseudo-)random interleaver needs to be stored in memory.

- BER performance improves when interleaver size increases. **Note.** Applications do not always allow large interleavers, implying long delays.
- If the interleaver size is small, then block interleavers are better than random interleavers.
- If the interleaver size is large, then random interleavers are better than block interleavers.

See [MF, Fig. 7.17].

©Patric Östergård

Linear Interleavers

A permutation of an interleaver given by a concise mathematical formula can save a lot of memory. A linear interleaver of length L maps a symbol in position i, $0 \le i \le L-1$, into position

$$pi + s \pmod{L}.$$

for some $0 \le p, s \le L-1$. It is required that $\gcd(p, L) = 1$.

©Patric Östergård

A Practical Aspect of Decoding

Implementing decoding algorithms (for turbo and other codes) often means handling values of very different orders of magnitude ⇒ overflow and underflow problems may occur.

Solution: Convert the values as well as the calculations into logarithmic form. Then

$$\begin{aligned} e^A \times e^B &= e^{A+B}, \\ e^A + e^B &= e^{\max(A,B)+\ln(1+e^{-|A-B|})}. \end{aligned}$$

©Patric Östergård

Performance of Turbo Codes

The performance of turbo codes depends on the signal-to-noise ratio (SNR):

- At low values of SNR, iterative decoding performs worse than uncoded transmission, even for a large number of iterations.
- At low to medium values of SNR, iterative decoding is very effective. The performance increases with an increase in the number of iterations.
- A medium to high values of SNR, iterative decoding converges in few iterations. Performance increases only slowly as SNR increases.

©Patric Östergård