Lecture no: 7

Channel Coding

Ove Edfors, Department of Electrical and Information Technology
Ove.Edfors@eit.lth.se

Contents (CHANNEL CODING)

- Overview
- Block codes
- Convolution codes
- Fading channel and interleaving

Coding is a much more complicated topic than this. Anyone interested should follow a course on channel coding.

OVERVIEW

Channel coding
Basic types of codes

Channel codes are used to add protection against errors in the channel.

It can be seen as a way of increasing the distance between transmitted alternatives, so that a receiver has a better chance of detecting the correct one in a noisy channel.

We can classify channel codes in two principal groups:

BLOCK CODES

Encodes data in blocks of k, using code words of length n.

CONVOLUTION CODES

Encodes data in a stream, without breaking it into blocks, creating code sequences.

Channel coding
Information and redundancy

EXAMPLE

Is the English language protected by a code, allowing us to correct transmission errors?

When receiving the following sentence with errors marked by ´-´:

“D- n-t w-rr- -b--t ---r d-ff-cult--s -n M-th-m-t-cs.
- c-n -ss-r- --- m-n- -r- st-ll gr--t-r.”

it can still be “decoded” properly.

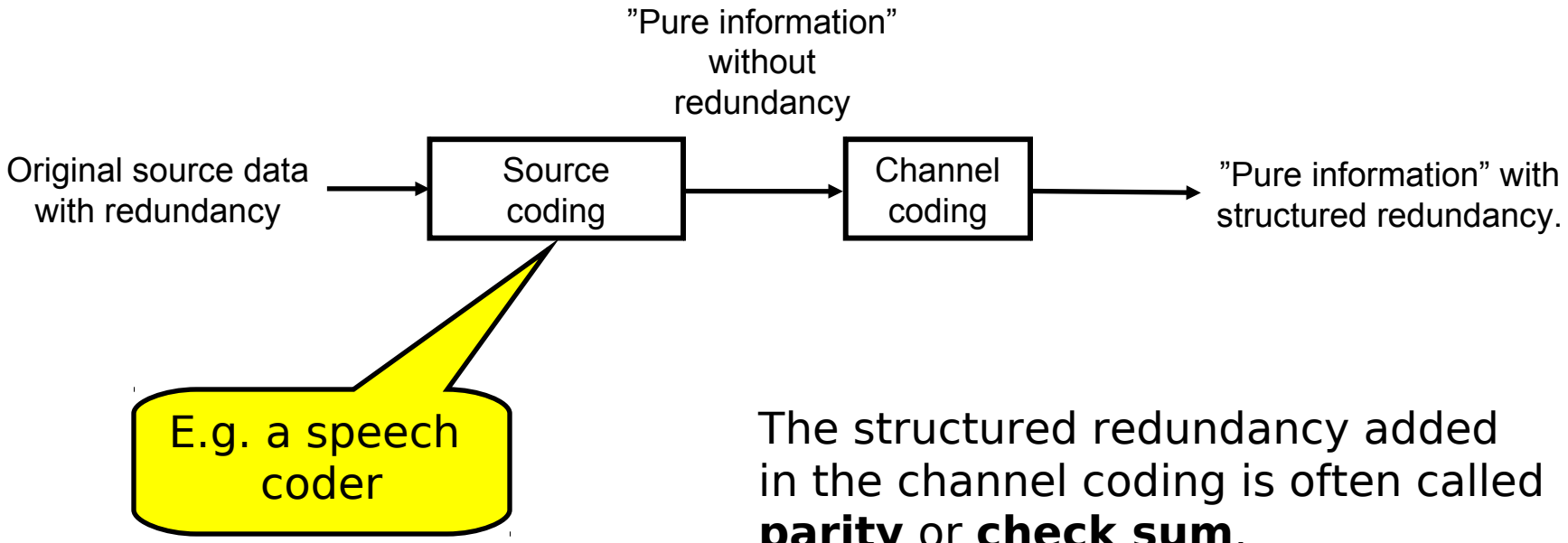
What does it say, and who is quoted?

There is something more than **information** in the original sentence that allows us to decode it properly, **redundancy**.

Redundancy is available in almost all “natural” data, such as text, music, images, etc.

Channel coding
Information and redundancy, cont.

Electronic circuits do not have the power of the human brain and needs more structured redundancy to be able to decode “noisy” messages.

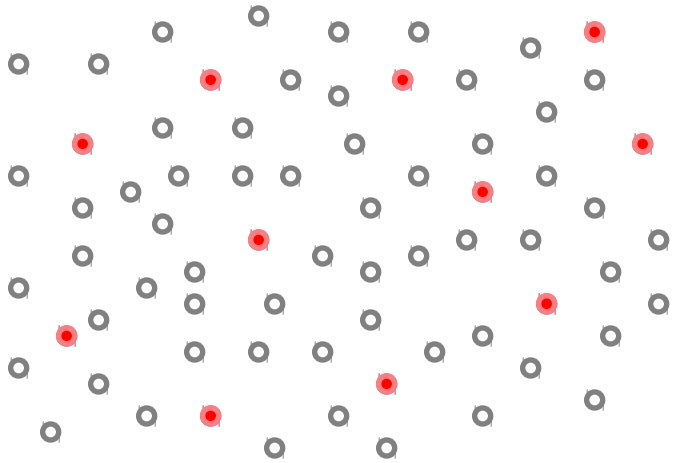


The structured redundancy added in the channel coding is often called **parity** or **check sum**.

Channel coding
Illustration of code words

Assume that we have a block code, which consists of k information bits per n bit code word ($n > k$).

Since there are only 2^k different information sequences, there can be only 2^k different code words.



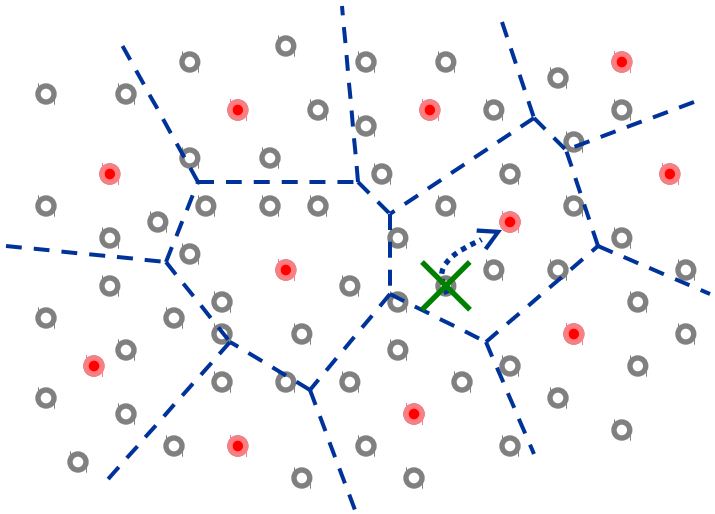
2^n different binary sequences of length n.

Only 2^k are valid code words in our code.

This leads to a larger **distance** between the valid code words than between arbitrary binary sequences of length n, which increases our chance of selecting the correct one after receiving a noisy version.

Channel coding
Illustration of decoding

If we receive a sequence that is not a valid code word, we decode to the **closest** one.



Received word

Using this “rule” we can create decision boundaries like we did for signal constellations.

One thing remains ... what do we mean by **closest**?
We need a distance measure!

Channel coding
Distances

The distance measure used depends on the channel over which we transmit our code words (if we want the rule of decoding to the *closest* code word to give a low probability of error).

Two common ones:

Hamming distance Measures the number of bits being different between two binary words.

Used for binary channels with random bit errors.

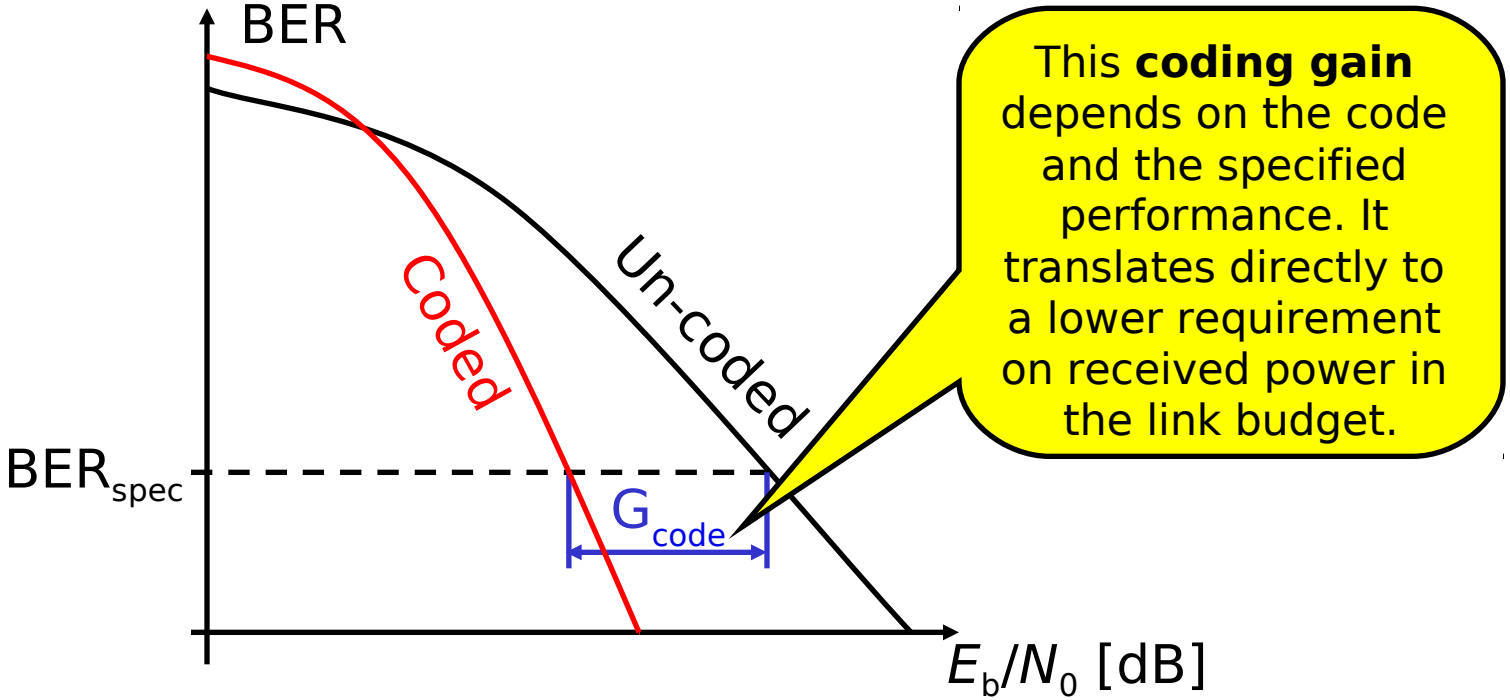
Euclidean distance Same measure we have used for signal constellations.

Used for AWGN channels.

We will look at this in more detail later!

Channel coding
Coding gain

When applying channel codes we decrease the E_b/N_0 required to obtain some specified performance (BER).



NOTE: $\boldsymbol{E_b}$ denotes **energy per information bit**, even for the coded case.

Channel coding
Bandwidth

When introducing coding we have essentially two ways of handling the indreased number of (code) bits that need to be transmitted:

1) Accept that the raw bit rate will increase the required radio bandwidth proportionally.

   This is the simplest way, but may not be possible, since we may have a limited bandwidth available.

2) Increase the signal constellation size to compensate for the increased number of bits, thus keeping the same bandwidth.

   Increasing the number of signal constellation points will decrease the distance between them. This decrease in distance will have to be compensated by the introduced coding.

BLOCK CODES

Channel coding
Linear block codes

The encoding process of a linear block code can be written as

$$\vec{x} = \underline{G}\,\vec{u}$$

where

$\vec{u}$ k - dimensional information vector

$\underline{G}$ n x k - dimensional generator matrix

$\vec{x}$ n - dimensional code word vector

The matrix calculations are done in an appropriate arithmetic.
We will primarily assume **binary codes** and **modulo-2** arithmetic.

Channel coding
Some definitions

Code rate:

$$R = \frac{\text{bits in}}{\text{bits out}} = \frac{k}{n}$$

Modulo-2 arithmetic (XOR):

$$\vec{x}_i + \vec{x}_j = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Hamming weight:

$$w(\vec{x}) = \text{number of ones in } \vec{x}$$

Hamming distance:

$$d(\vec{x}_i, \vec{x}_j) = w(\vec{x}_i + \vec{x}_j)$$

Minimum distance of code:

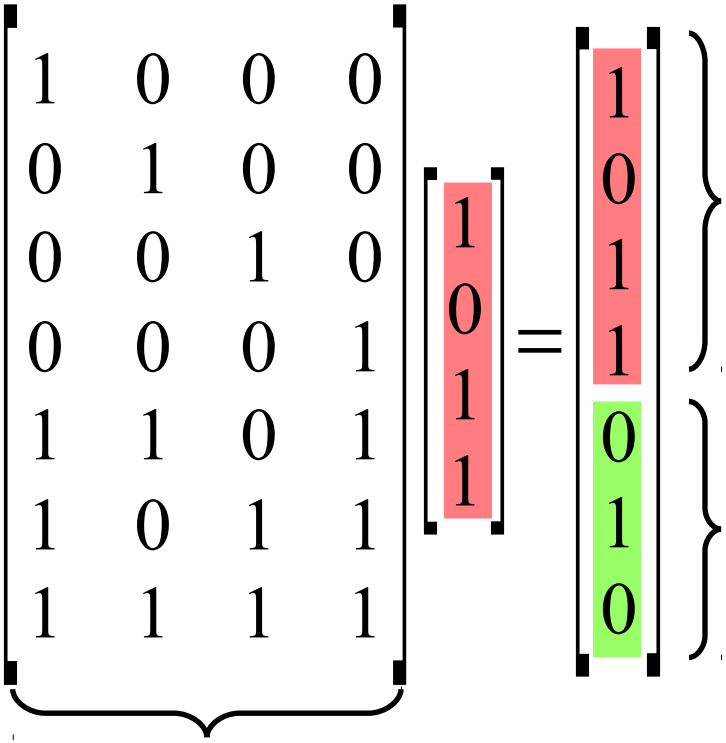
$$d_{min} = \min_{i \neq j} d(\vec{x}_i, \vec{x}_j)$$
$$= \min_{i \neq j} w(\vec{x}_i + \vec{x}_j)$$

The minimum distance of a code determines its error correcting performance in non-fading channels.

Note: The textbook sometimes use the name "**Hamming distance of the code**" ($\boldsymbol{d_H}$) to denote its minimum distance.

Channel coding
Encoding example

For a specific (n,k) = (7,4) code we encode the information sequence 1 0 1 1 as



If the information is directly visible in the code word, we say that the code is **systematic**.

In addition to the *k* information bits, there are *n-k* = 3 **parity** bits.

Channel coding
Encoding example, cont.

Encoding all possible 4 bit information sequences gives:

Information	Code word	Hamming weight
0 0 0 0	0 0 0 0 0 0 0	0
0 0 0 1	0 0 0 1 1 1 1	4
0 0 1 0	0 0 1 0 0 1 1	3
0 0 1 1	0 0 1 1 1 0 0	3
0 1 0 0	0 1 0 0 1 0 1	3
0 1 0 1	0 1 0 1 0 1 0	3
0 1 1 0	0 1 1 0 1 1 0	4
0 1 1 1	0 1 1 1 0 0 1	4
1 0 0 0	1 0 0 0 1 1 0	3
1 0 0 1	1 0 0 1 0 0 1	3
1 0 1 0	1 0 1 0 1 0 1	4
1 0 1 1	1 0 1 1 0 1 0	4
1 1 0 0	1 1 0 0 0 1 1	4
1 1 0 1	1 1 0 1 1 0 0	4
1 1 1 0	1 1 1 0 0 0 0	3
1 1 1 1	1 1 1 1 1 1 1	7

This code has a minimum distance of 3.
(Minimum code word weight of a linear code, excluding the all-zero code word.)

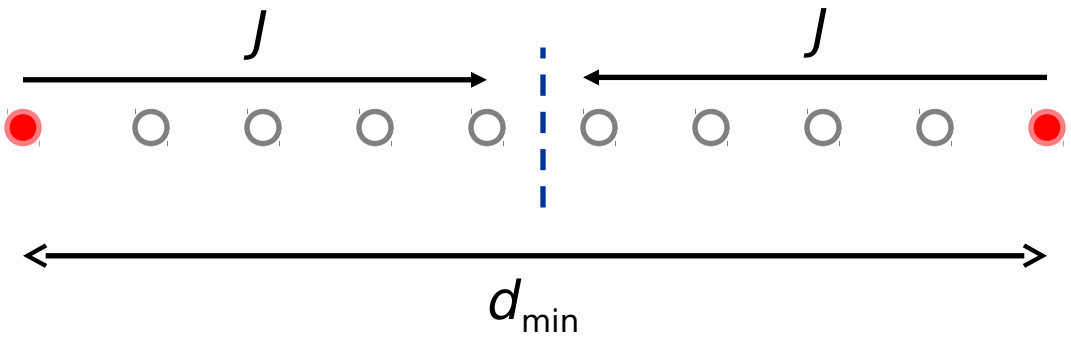
This is a (7,4) Hamming code, capable of correcting one bit error.

Channel coding
Error correction capability

A binary block code with minimum distance $d_{\min}$ can correct J bit errors, where

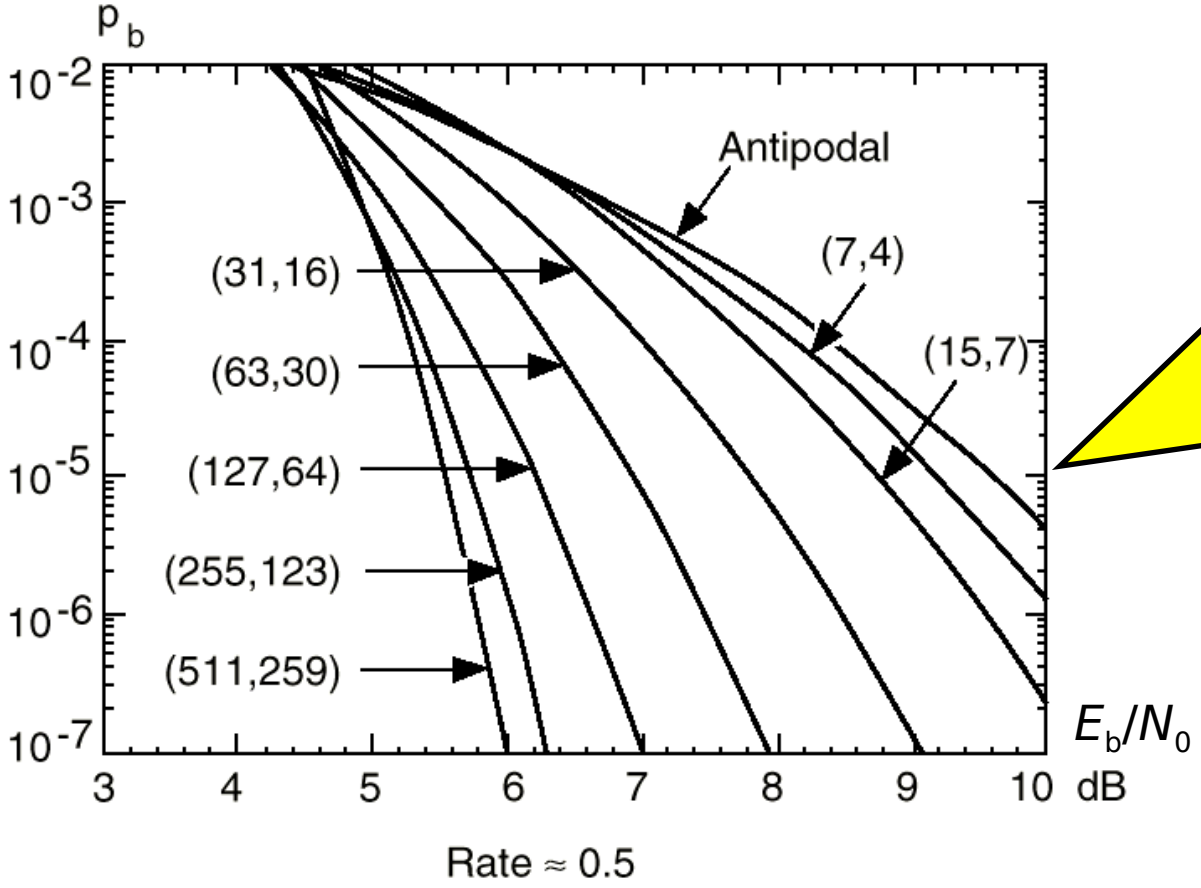
$$J = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

Rounded down to nearest integer.



Channel coding
Performance and code length

Longer codes (with same rate) usually have better performance!



This example is for a non-fading channel! Not in textbook

Drawbacks with long codes is complexity and delay.

Channel coding
Undetected errors

Not in textbook

If the bit-errors introduced in the channel change the transmitted code word into another valid code word, the receiver is unable to detect that errors have occured in the channel.

With a minimum distance of d_{min}, there must be at least d_{min} bit-errors in the code word for this to happen.

Given the probability of (uncoded/raw) bit-error p in the channel, we can upper-bound the probability of **undetected errors** as:

$$P_{\text{ue}} \leq \left(2^k - 1\right) p^{d_{\text{min}}} \left(1-p\right)^{n-d_{\text{min}}}$$

The bounding comes from: We assume that ALL code words are at the minimum distance d_{min}.

Channel coding
Bit error probability after decoding

Not in textbook

The probability that m bit errors occur in a code word of length n, with a raw/undoced bit-error probability p is

$$\Pr\{m \text{ bit errors}\} = \binom{n}{m} p^m (1-p)^{n-m}$$

If our code can correct J bit-errors, the probability of decoding to the wrong code word is

$$\Pr\{\text{code word error}\} = \sum_{m=J+1}^{n} \Pr\{m \text{ bit errors}\} = \sum_{m=J+1}^{n} \binom{n}{m} p^m (1-p)^{n-m}$$

Assuming that $d_{\min}$ of the n bits are in error when we decode to a wrong code word, we approximate the coded bit-error probability

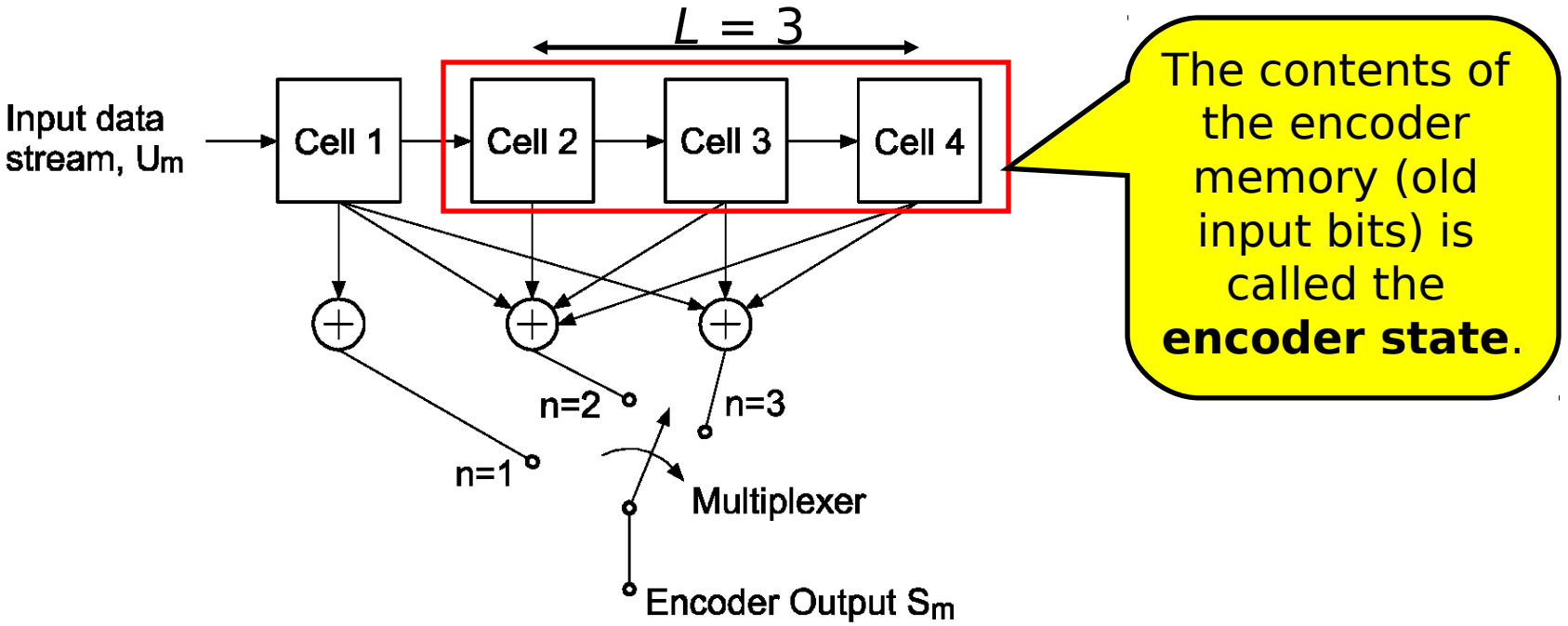
$$p_{\text{b, coded}} \approx \frac{d_{\min}}{n} \Pr\{\text{code word error}\} = \frac{d_{\min}}{n} \sum_{m=J+1}^{n} \binom{n}{m} p^m (1-p)^{n-m}$$

CONVOLUTION CODES

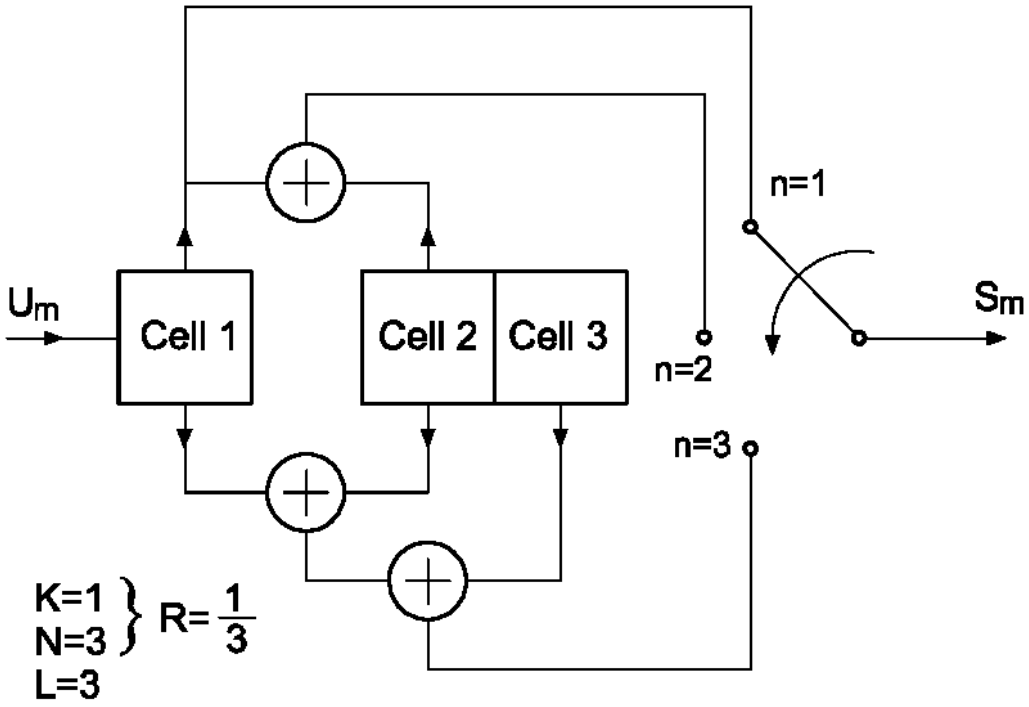
Channel coding
Encoder structure

In convolution codes, the coded bits are formed as convolutions between the incoming bits and a number of generator sequences.

We will view the encoder as a shift register with memory *L* and *N* generator sequences (convolution sums).



Channel coding
Encoding example



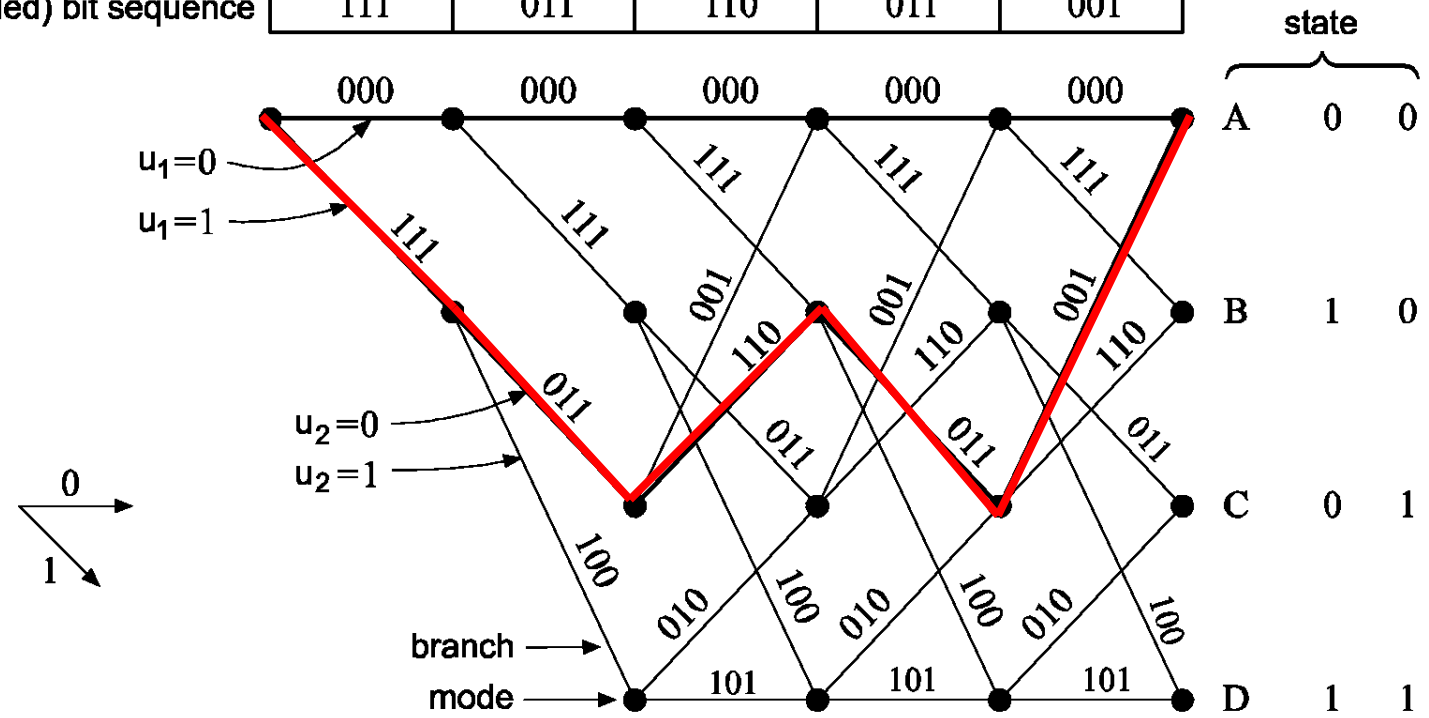
Input	State	Output	Next state
0	00	000	00
1	00	111	10
0	01	001	00
1	01	110	10
0	10	011	01
1	10	100	11
0	11	010	01
1	11	101	11

We usually start the encoder in the **all-zero** state!

Channel coding
Encoding example, cont.

We can view the encoding process in a trellis created from the table on the previous slide.

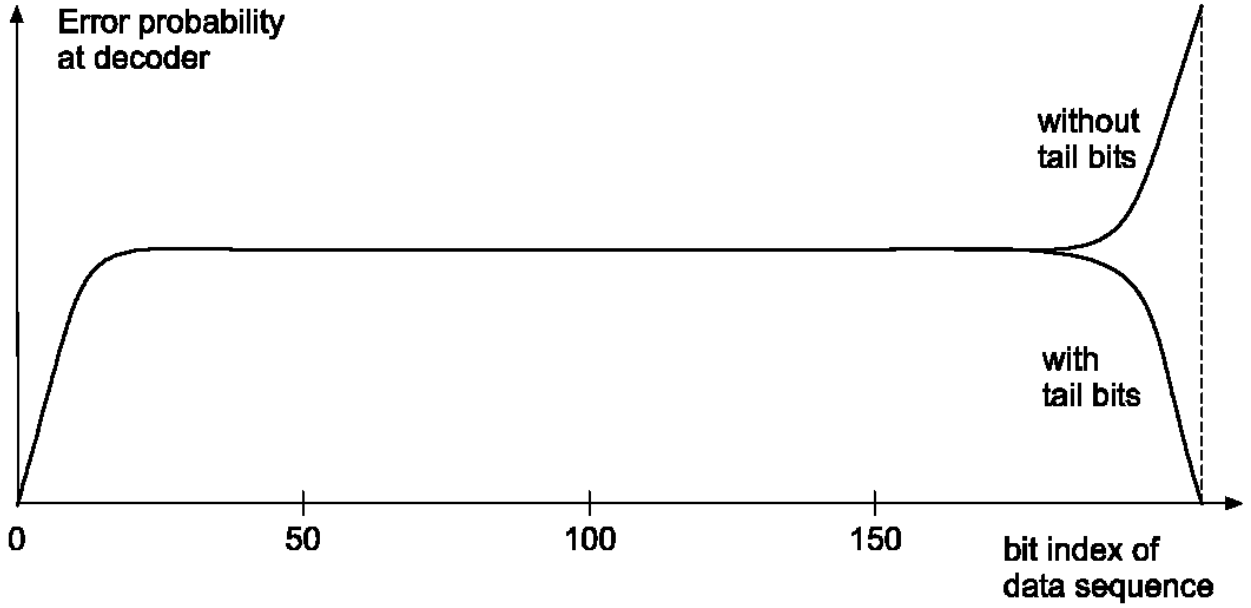
Input (information) bit sequence	1	0	1	0	0
Output (coded) bit sequence	111	011	110	011	001



Channel coding
Termination

At the end of the information sequence, it is common to add a **tail** of L zeros to force the encoder to end (terminate) in the zero state.

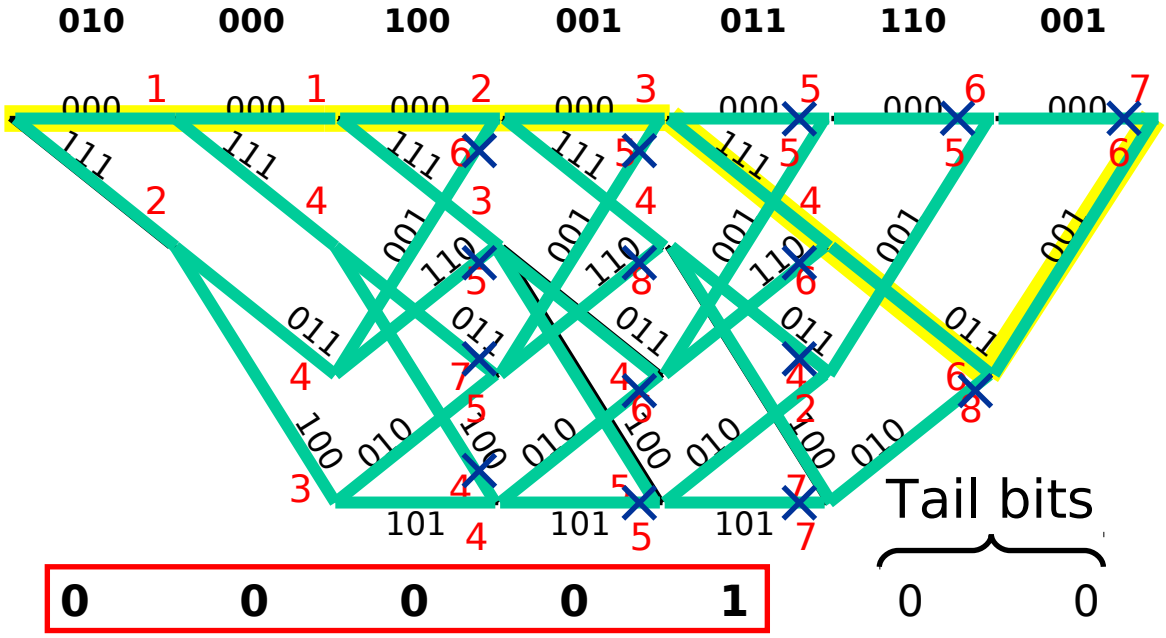
This improved performance, since a decoder knows both the starting state and ending state.



Channel coding
A Viterbi decoding example

We want to find the path in the trellis (the code sequence) that is closest to our received sequence. This can be done efficiently, using the **Viterbi algorithm** (search trellis, accumulate distances, discard path with highest distance whenever they “collide” and back-trace from the end).

Received sequence:



Channel coding
Soft decoding

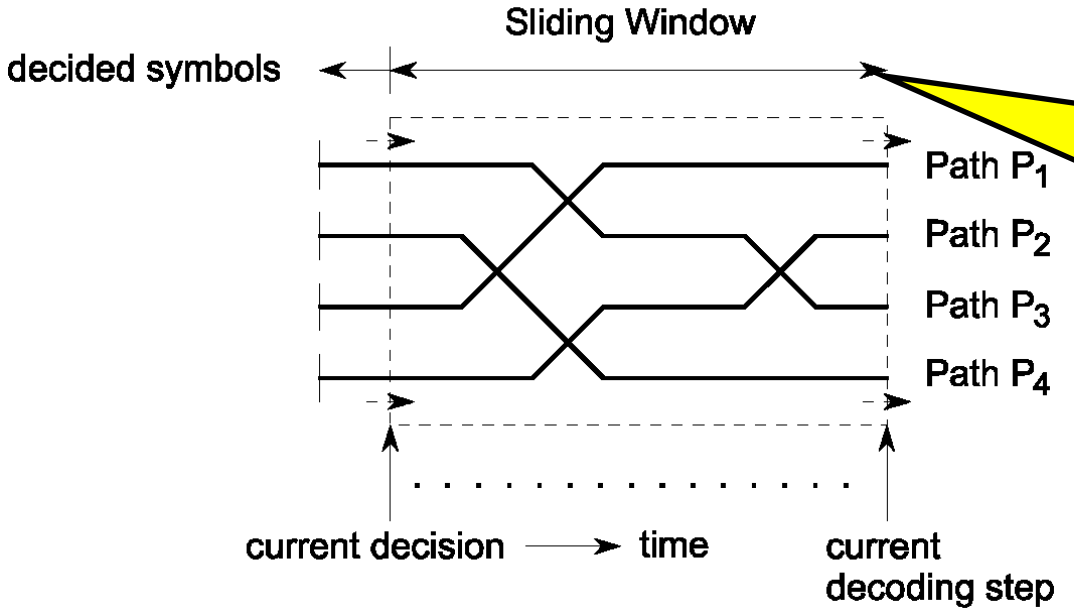
We have given examples of hard decoding, using the **Hamming distance**.

If we do not detect ones and zeros before decoding our channel code, we can use **soft decoding**. In the AWGN channel, this means comparing Euclidean distances instead.

Channel coding
Surviving paths

The Viterbi algorithm needs to keep track of one surviving path per state in the trellis. For long code sequences this causes a memory problem.

In practice we only keep track of surviving paths in a window consisting of a certain number of trellis steps. At the end of this window we enforce decisions on bits, based on the metric in the latest decoding step.



Experience shows that a window length of **6 times the encoder memory** only lead to minor performance losses.

FADING CHANNELS AND INTERLEAVING

Channel coding
Fading channels and interleaving

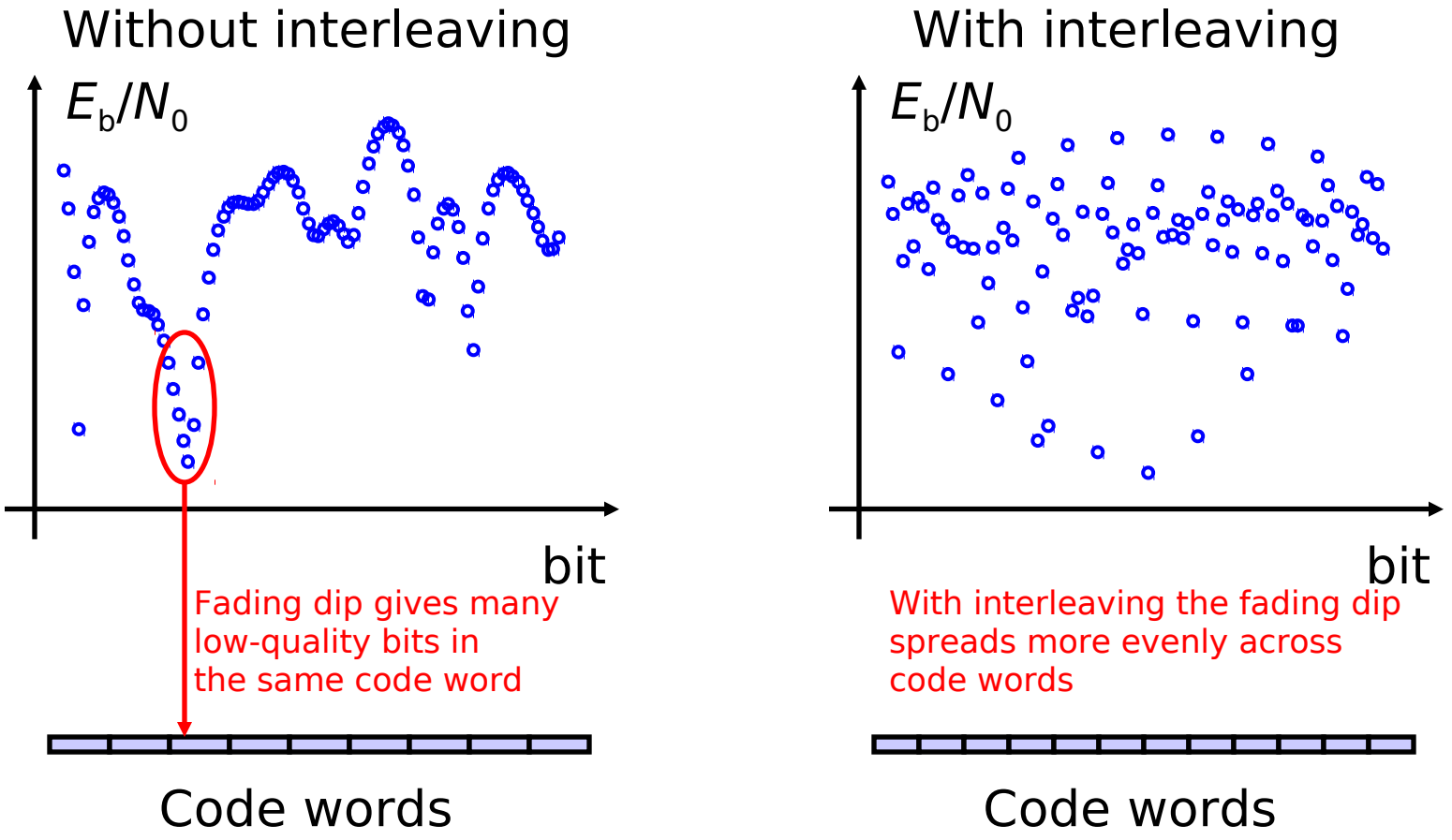
In fading channels, many received bits will be of “low quality” when we hit a fading dip.

Coding may suffer greatly, since many “low quality” bits in a code word may lead to a decoding error.

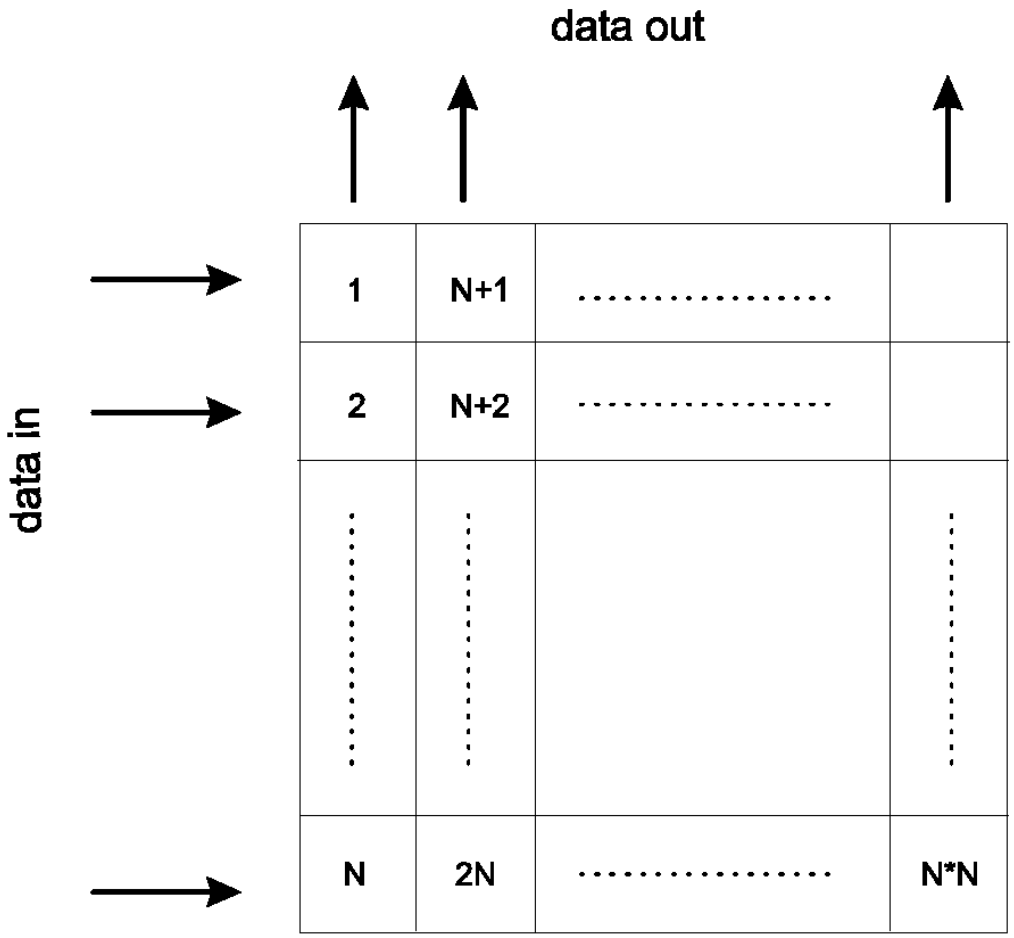
To prevent all “low quality” bits in a fading dip from ending up in the same code word, we rearrange the bits between several code words before transmission ... and rearrange them again at the receiver, before decoding.

This strategy of breaking up fading dips is called **interleaving**.

Channel coding
Distribution of low-quality bits



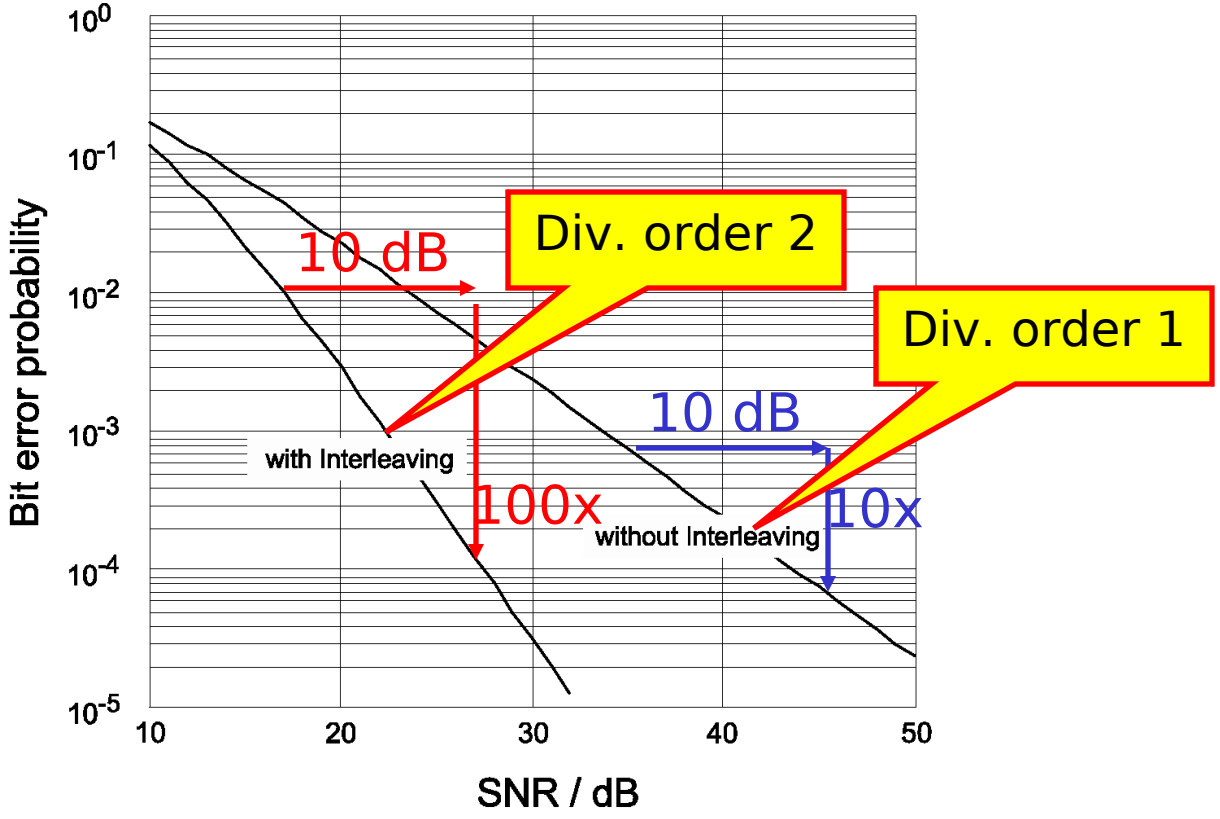
Channel coding
Block interleaver



The writing and reading of data in interleavers cause a delay in the system, which may cause other problems.

Channel coding
Interleaving - BER example

BER of a R=1/3 repetition code over a Rayleigh-fading channel, with and without interleaving. Decoding strategy: majority selection.



Hard decoding of block codes with minimum distance $d_{\min}$ gives a code diversity order of

$$\left\lfloor \frac{d_{\min}-1}{2} \right\rfloor + 1$$

if the interleaving works properly.

Summary (CHANNEL CODING)

- Channel coding is used to improve error performance
- For a fixed requirement, we get a **coding gain** that translates to a lower received power requirement.
- The two main types of codes are **block codes** and **convolution codes**
- Depending on the channel, we use different **metrics** to measure the **distances**
- Decoding of convolution codes is efficiently done with the **Viterbi algorithm**
- In fading channels we need **interleaving** in order to break up fading dips (but causes delay)