Protection de l'information

Channel coding

Haïfa Farès

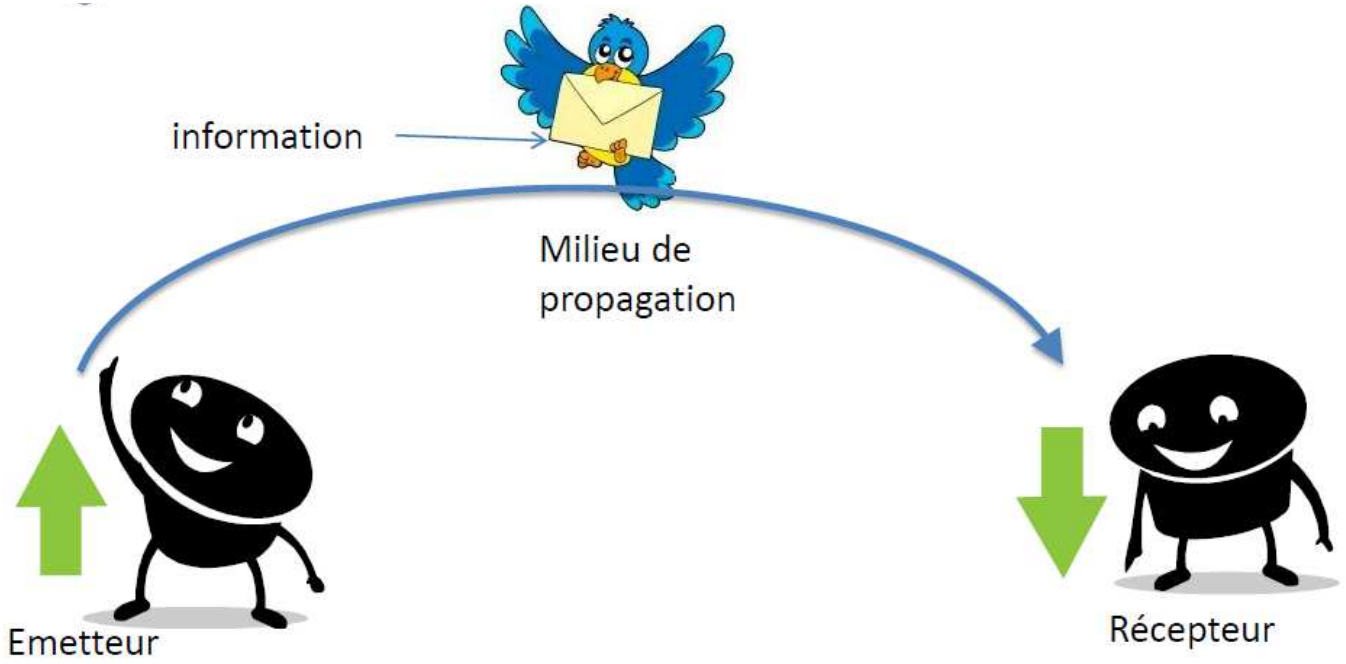
CentraleSupélec

I- Introduction

1. *History*
2. *Communication Chain*

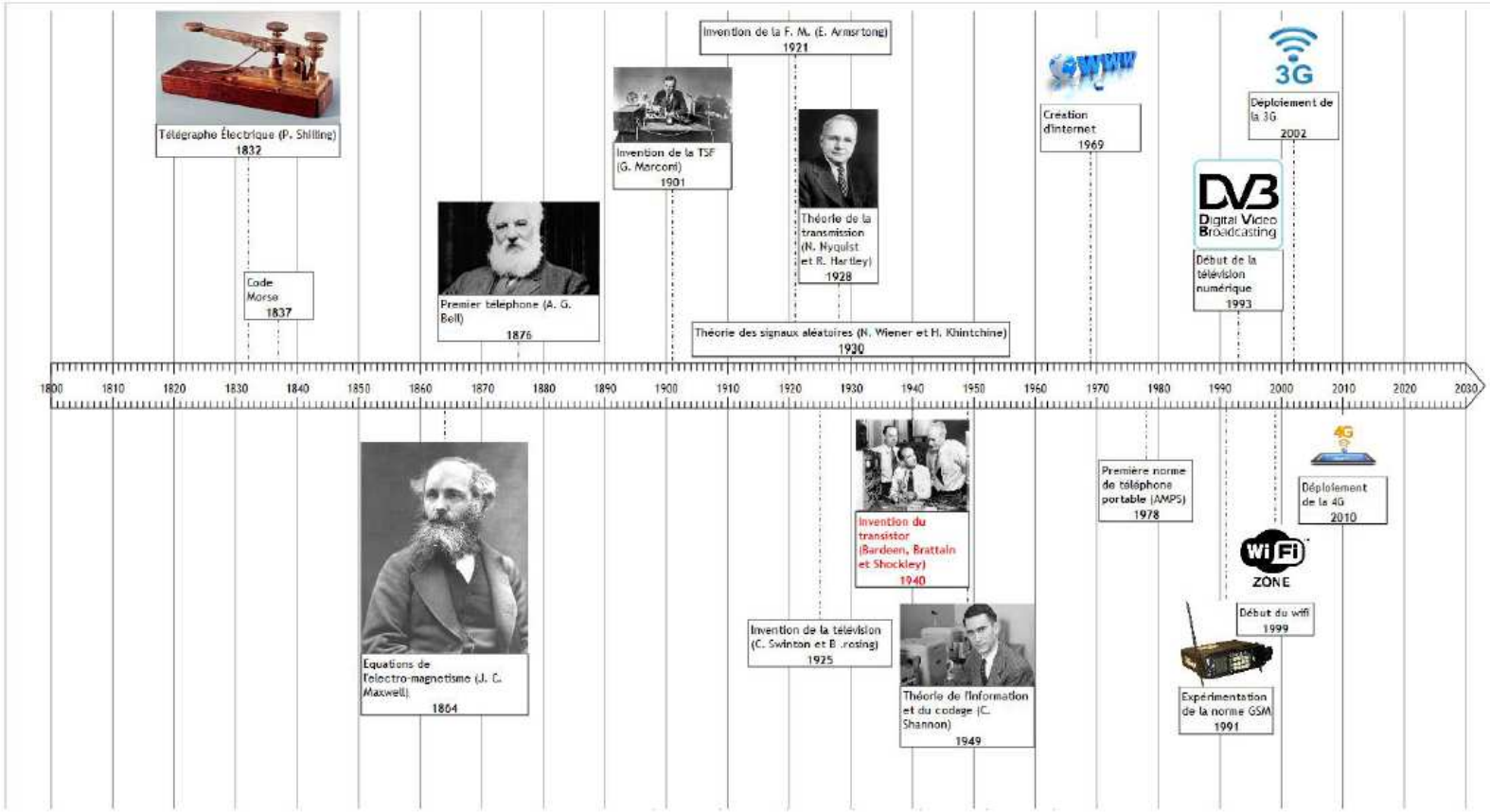
What is a *communication*?

A **communication** is to transmit an **information** from a **source** to a **destination** through a **propagation medium**



CentraleSupélec

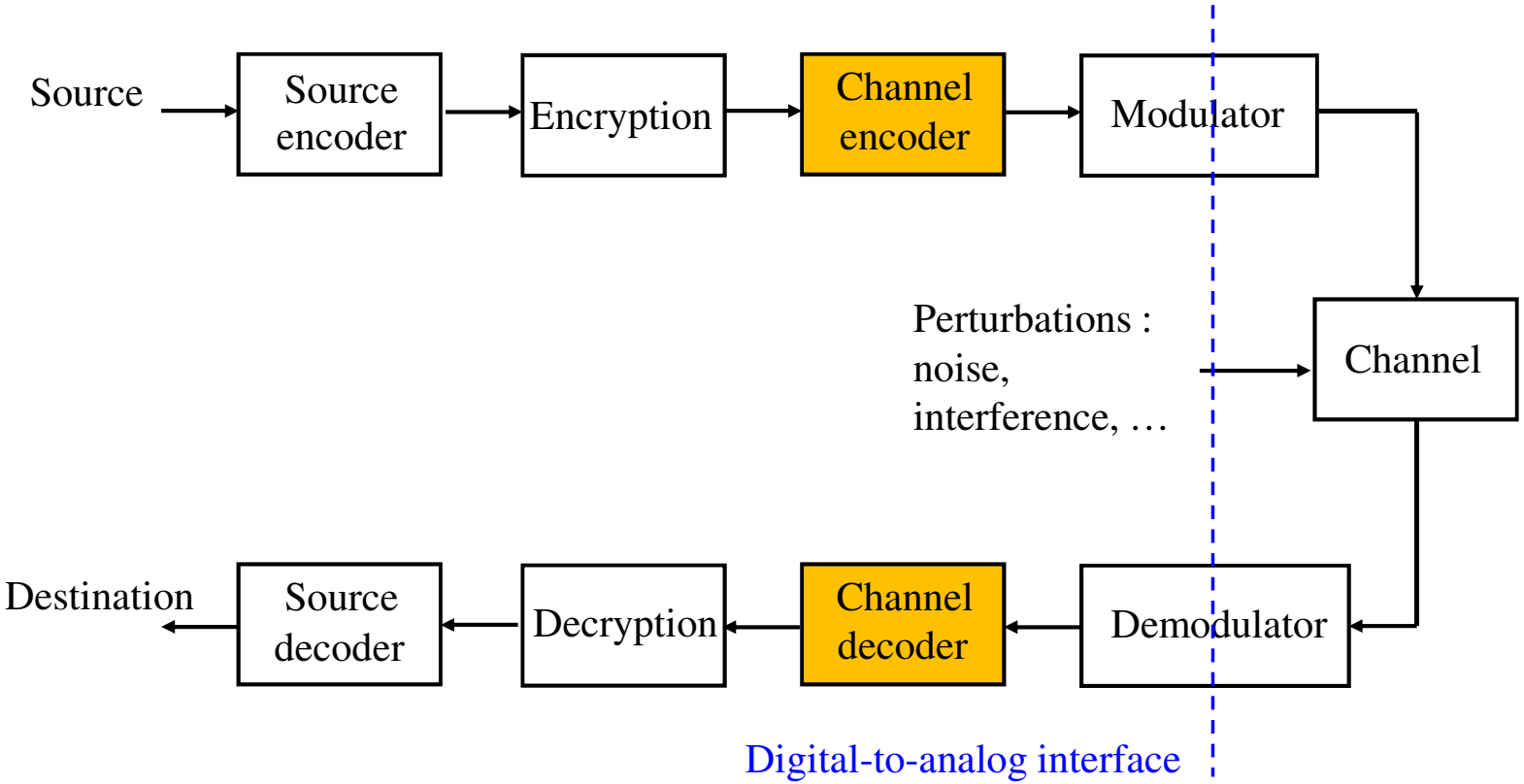
History of communications



CentraleSupélec

Communication chain

- Diagram of a point-to-point digital transmission

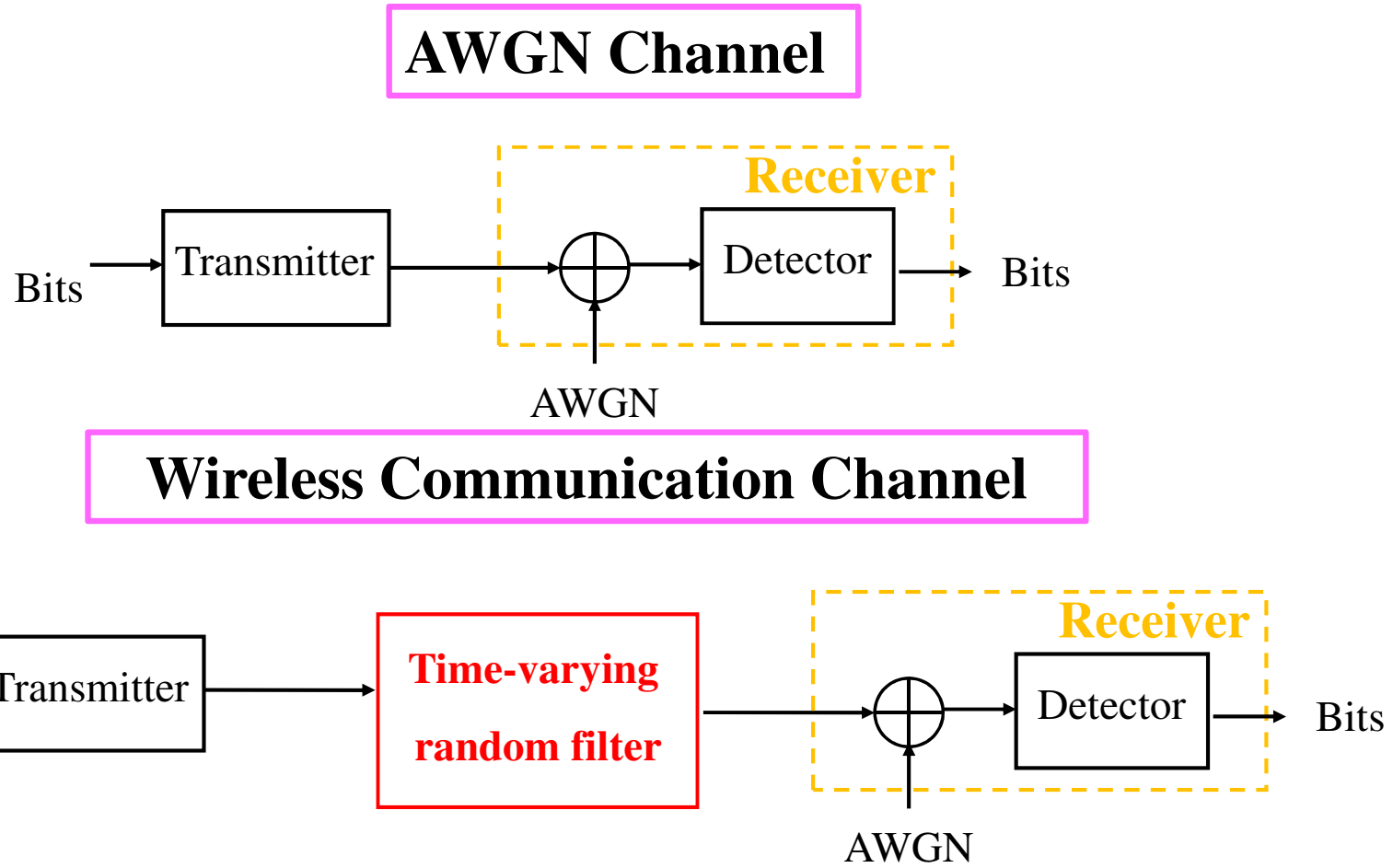


CentraleSupélec

II- Wireless channel

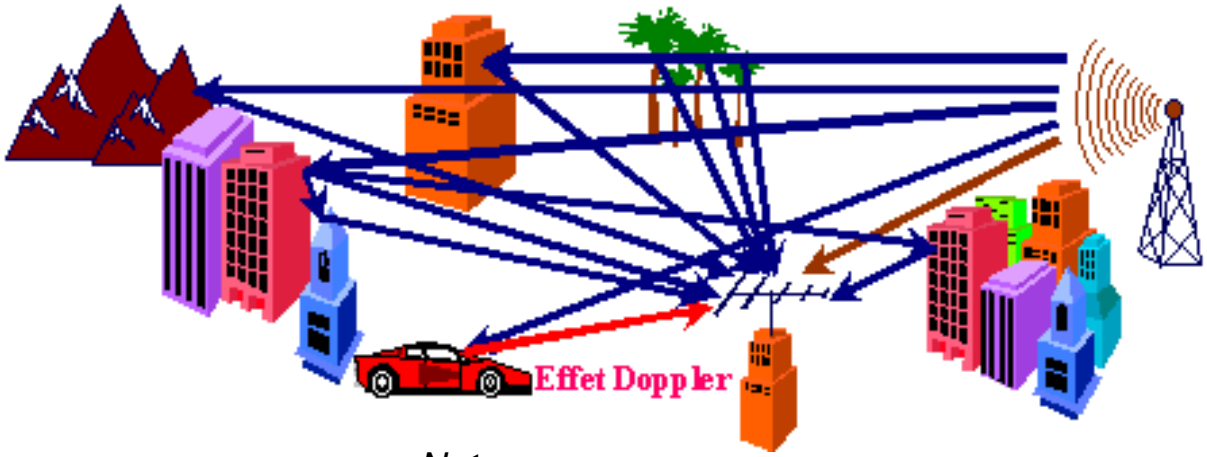
1. *AWGN channel vs. Wireless channel*
2. *Fading*
3. *Multi-path cannel*

AWGN channel vs. Wireless channel



CentraleSupélec

Multi-path channel



$$r(t) = \sum_{n=0}^{N-1} h_n s(t - \tau_n) + v(t)$$

n^{th} path attenuation

n^{th} path delay

No line of sight → Rayleigh distribution

line of sight → Rice distribution

CentraleSupélec

Fading distribution

- In a classical environment (theoretical perfect model) , noise is generally considered to be AWGN (Additive White Gaussian Noise), modeled by the normal distribution.

$$f_h(h) = \frac{e^{-\frac{h^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

- In the case of a fading channel, the probability density of the attenuations will follow a law of the form:

$$f_h(h) = \frac{h}{\sigma^2} e^{-\left(\frac{h^2+A^2}{2\sigma^2}\right)} I_0\left(\frac{hA}{\sigma^2}\right)$$ Rice distribution

  - If A=0, we come to the Rayleigh distribution

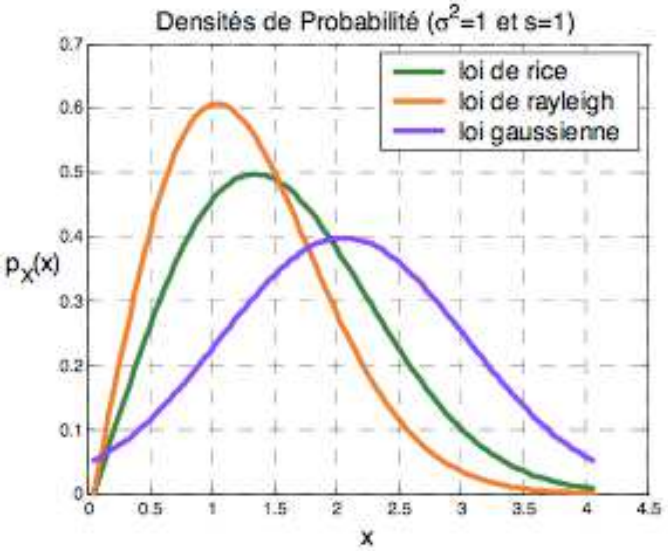
- Rayleigh distribution is the preferred fading model because it models the *most severe conditions* with the *simplest expression*.

CentraleSupélec

Rice vs. Rayleigh

- The Rayleigh model is used more than Rice model for the following reasons:

  - The Rayleigh model corresponds to a propagation without line of sight, and therefore more constraining. This allows to rely on the worst case
  - The Rayleigh model corresponds to many practical cases of propagation, in an urban environment or in indoor propagation (indoor)
  - The mathematical expression of a Rayleigh distribution is simpler than that of a Rice one



Time variations of the channel: Coherence time

- The coherence time T_c gives an approximation of the time during which the behavior of the channel is relatively constant
- The coherence timeT_c allows to characterize the time variation of the channel in the time domain
- Principle : compare T_c to T, symbol period.

	Fading variation
$T_c > T$	Slow fading
$T_c < T$	Fast fading

CentraleSupélec

Slow fading vs. fast fading

- Slow fading :
  - The channel changes but *slowly*
  - Channel coefficients remain constant all the time of a frame transmission

$$r(t) = \sum_{n=0}^{N-1} h_n s(t - \tau_n) + v(t)$$

- Fast fading :
  - The channel changes *very quickly*
  - It is impossible to consider the gains of the paths as constant on an observation window

$$r(t) = \sum_{n=0}^{N-1} h_n(t) s(t - \tau_n(t)) + v(t)$$

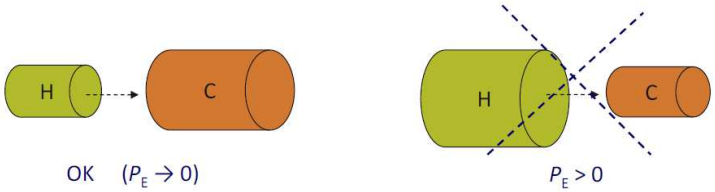
III- Compensation

1. *Information theory review: channel capacity*

« The theory concerned the performance of the very best system possible and how to approach that performance (without explicitely designing the system). An understanding of each piece of the system was necessary to achieve this objective. »

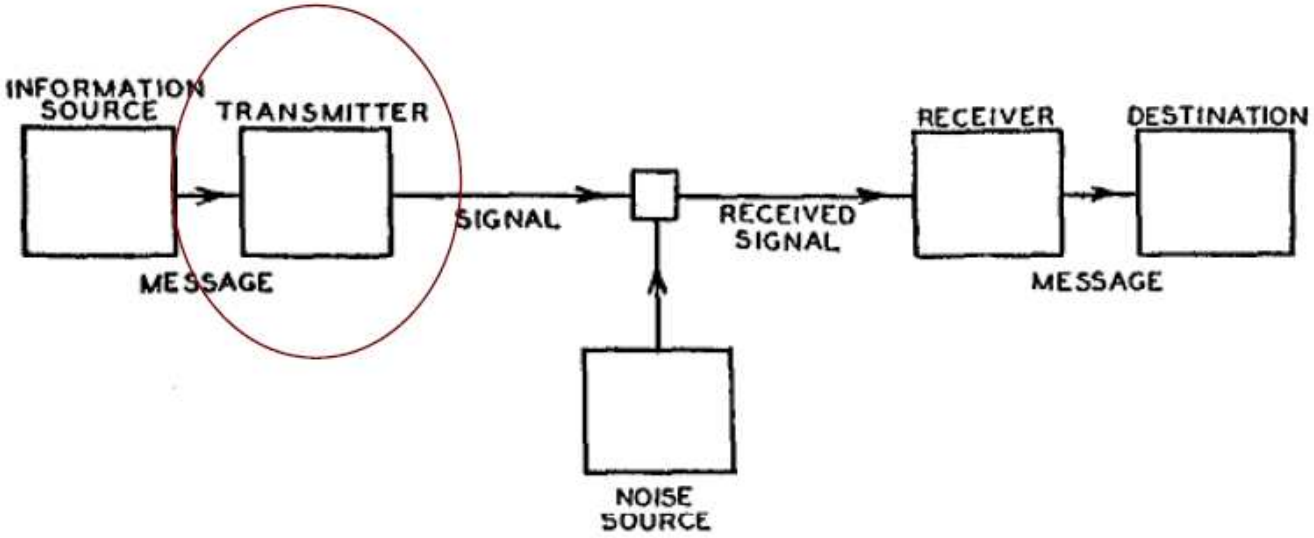
2. *Channel coding principle and classification*

Channel capacity



- It is the maximum data rate that may be carried under certain conditions on a transmission channel with an error probability tending to zero
  - *The maximization is over all sources which might be used as input to the channel*
- Concept closely related to the following
  - Bit rate: expressed in bits / s, at which data can be transmitted
  - Bandwidth: frequency band occupied by the transmitted signal
  - Noise: Channel perturbation
  - Error probability
- Target: exploit bandwidth as efficiently as possible
- Interpretation : If Di <C, there is at least one code allowing to have an error-free transmission.

Channel capacity: the under hidden property



« Choosing the encoding relationship between the source output and the channel input is the most important degree of freedom that we have in designing a system for reliable communication. »

CentraleSupélec

Nyquist theorem

- For bandwidth *B*, the highest practical rate is *2B* bits/s.
- With a multilevel signal, where M is the number of discrete elements, Nyquist's theorem becomes:

$$C = 2B \log_2 M$$

- Be careful to the overload for the receiver!

Shannon Theorem

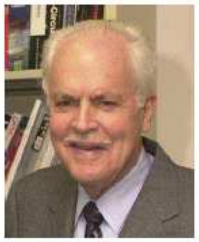
- The higher the data rate, the more disruptive random noise will be detrimental
- Key parameter: Signal-to-noise ratio

$$SNR_{dB} = 10\log_{10}\frac{\text{Puissancedu signal}}{\text{Puissancedu bruit}}$$

- Gaussian channel capacity (resources are bandwidth B and SNR) in bits/s:

$$C = B\log_2(1+SNR)$$

Coding theory



Irving Reed

David Muller

Codes de Reed-Muller (1954)

Richard Hamming

Code de Hamming (1947)

Marcel Golay

Code de Golay (1949)

Peter Elias

Codes convolutifs

Codes produits

(1954)

Robert Gallager

Codes LDPC (1960)

David Forney

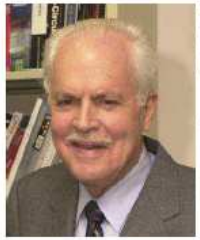
Codes concaténés (1965)

Andrew Viterbi

Algorithme de Viterbi (1967)

Gottfried Ungerboeck

Modulations codées en treillis (1978)



Irving Reed Gustave Solomon

Codes Reed-Solomon (1960)

Elwin Berlekamp

Décodage des codes RS (1968)

Claude Berrou

Alain Glavieux

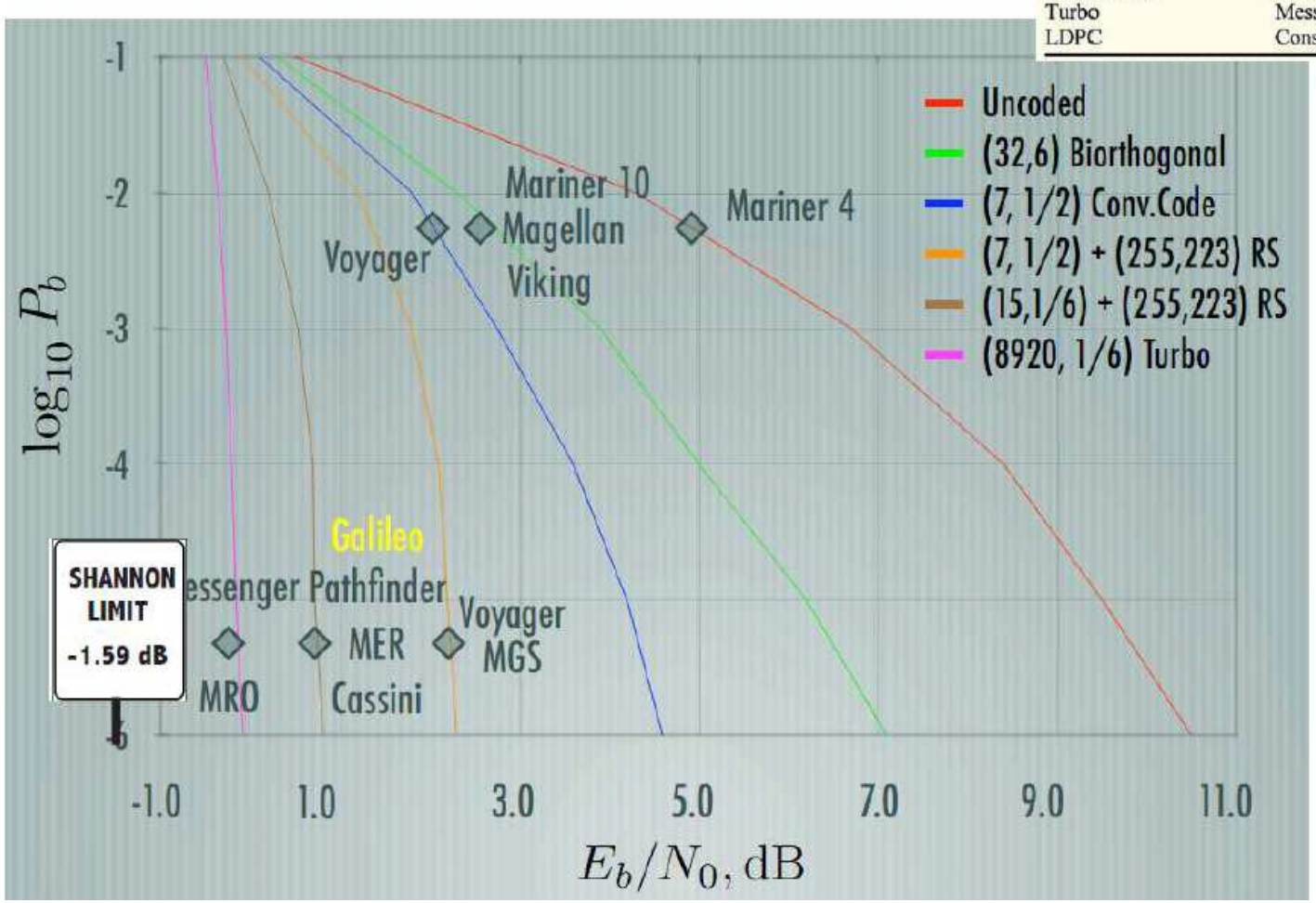
Turbo-codes (1993)

CentraleSupélec

60 years of coding…

Code	Mission(s)	Years
Uncoded	Explorer, Mariner, many others	1958-present
(25,1/2) convolutional	Pioneer, Venus	1968-1978
(32,6) Reed-Muller	Mariner, Viking	1969-1975
Golay	Voyager	1977-present
RS(255,223)+(7,1/2)	Voyager, Galileo, many others	1977-present
RS(255,223)+(7,1/3)	Voyager	1977-present
RS(255,var)+(14,1/4)	Galileo	1989-2003
RS+(15,1/6)	Cassini, Mars Pathfinder, others	1996-present
Turbo	Messenger, Stereo, MRO, others	2004-present
LDPC	Constellation, MSL	est. 2009

Table 1 Codes Used by NASA Missions



R. J. McEliece, « Viterbi's impact on the exploration of the solar system », 2005

CentraleSupélec

Minimum Distance

- The Hamming distance between two binary codewords: number of different bits.

  For example, d(0001, 0111) = 2
- Minimum distande of a code $d_{\min} = \min_{\mathbf{y}_1, \mathbf{y}_2 \,;\, \mathbf{y}_1 \neq \mathbf{y}_2} d(\mathbf{y}_1, \mathbf{y}_2)$

- Since the sum of two codewords is another codeword, and the distance between two codewords is the weight of their sum, we also have:

$$d_{\min} = \min_{\mathbf{y} \,;\, \mathbf{y} \neq \mathbf{0}} P(\mathbf{y})$$

- *Determining* $d_{\min}$ for block codes
  - If the number of codewords is not too high: exhaustive search
  - Otherwise : $d_{\min}$ = number of linearly dependent (zero sum) columns of **H** (generator matrix).

Minimum Distance

$d(a,b) = d(b,a)$	Symmetry
$d(a,b) \geq 0$	Positivity
$d(a,b) = 0$	if and only if $a = b$
$d(a,b) + d(b,c) \geq d(a,c)$	Triangular inequality
$d(a \oplus x, b \oplus x) = d(a,b)$	Invariance

CentraleSupélec

Minimum Distance: fundamental parameter

- *Detect an error*

  ✓ How many errors can be detected with certainty ? $d - 1$

  The receiver definitely detects ALL false messages, as long as the number of errors N verifies 0 <N <d.
  Some false messages with d errors (or more) are not detected.

- *Correct an error*

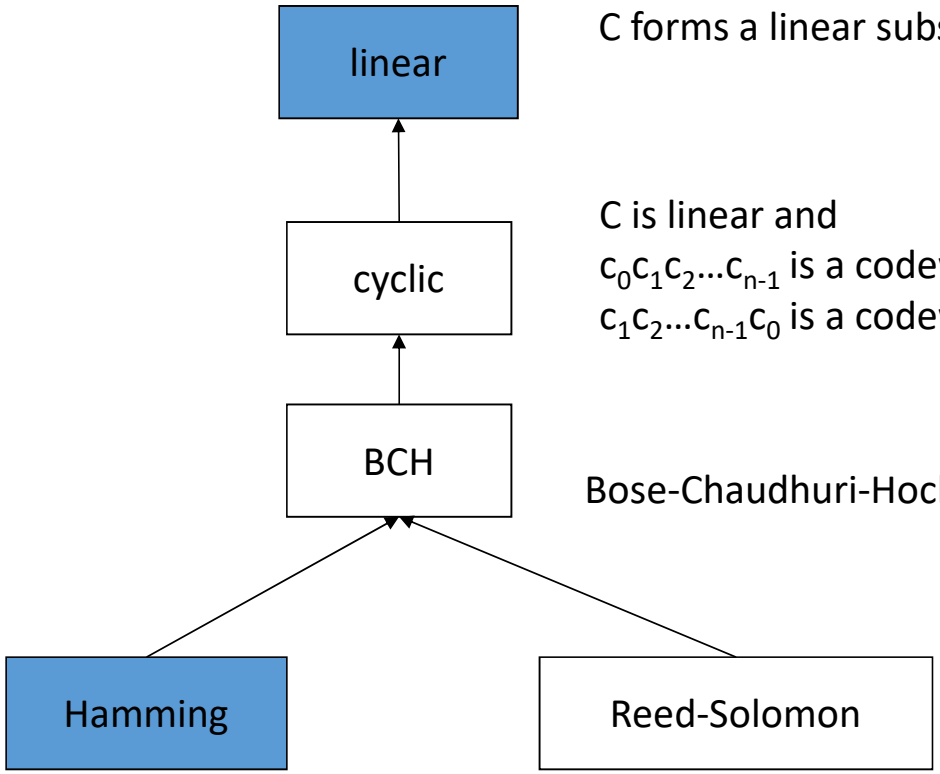
  ✓How many errors can be corrected with certainty? $\left\lfloor \frac{d-1}{2} \right\rfloor$

  The messages are corrected as long as the number of errors N satisfies 0 ≤ N <d/2 (strict).
  False messages such as d/2 ≤ N are not always well corrected.

Coding - Examples

- Examples of linear block codes
  - Parity codes
  - Repetition code
  - Hamming code
- Cyclic codes
  - We study here a subclass of the linear codes defined on *GC(2)*.
  - For each codeword of length *n*, a polynomial of degree is associated with the most coefficients equal to the number of bits of the codeword.
  - Cyclic codes are mainly used for error detection.
  - *Definition* : A linear code *C*(*n*, .) is cyclic if any cyclic permutation of a codeword is also a codeword.

Hierarchy of codes



C forms a linear subspace of $\sum^n$ of dimension k

C is linear and
$c_0c_1c_2…c_{n-1}$ is a codeword implies
$c_1c_2…c_{n-1}c_0$ is a codeword

Bose-Chaudhuri-Hochquenghem

These are all block codes.

Block linear code

- A linear code is simply defined by a generator matrix.
- Message x generator matrix = coded message, for example:

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

- So the matrix $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ generates the code:

Information bits	00	01	10	11
Corresponding codeword	000	111	001	110

CentraleSupélec

Systematic block linear code

- A systematic codeword contains the information bits as a prefix.
- For this case, the generator matrix contains an identity block on the left.
- Example of the systematic linear code of even parity:

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{1} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{1} & 1 \\ \mathbf{1} & \mathbf{0} & 1 \\ \mathbf{1} & \mathbf{1} & 0 \end{pmatrix}$$

- In block representation, it gives $M * \begin{pmatrix} I & H \end{pmatrix} = \begin{pmatrix} M & C \end{pmatrix}$

Error detection – Syndrom method

1. We receive the coded message with a possible error $R = M * \begin{pmatrix} I & H \end{pmatrix} + E.$
2. We multiply by the control matrix $\begin{pmatrix} -H \\ I \end{pmatrix}$

$$R * \begin{pmatrix} -H \\ I \end{pmatrix} = M * \begin{pmatrix} I & H \end{pmatrix} * \begin{pmatrix} -H \\ I \end{pmatrix} + E * \begin{pmatrix} -H \\ I \end{pmatrix}$$

3. Important property for decoding in the case where the H block is square

$$\begin{pmatrix} I & H \end{pmatrix} * \begin{pmatrix} -H \\ I \end{pmatrix} = 0$$

4. By the preceding property, we obtain the syndrome: $S = R * \begin{pmatrix} -H \\ I \end{pmatrix} = E * \begin{pmatrix} -H \\ I \end{pmatrix}$
5. An error is detected if and only if $S \neq 0$.

CentraleSupélec

Cyclic block linear code

- Why use a cyclic code?

  - The problem for all linear codes is to find a good generator matrix.
  - Cyclic codes have a particularly strong structure to help on this point: any cyclic systematic code (n; k) is generated by the polynomials of degree $n - k$ dividing $x^n - 1$.

Cyclic block linear code: Encoding from a polynomial

1. We start for example from the polynomial
$$x^7 - 1 = (x+1)(x^3+x+1)(x^3+x^2+1)$$
2. For the codewords of length $n = 7$, we have 3 possible codes:
   - The one generated by x + 1 encoding messages of length $k = n - 1 = 6$,
   - Those generated by $x^3 + x + 1$ and by $x^3 + x^2 + 1$ encoding messages of length $k = n-3 = 4$.
3. We encode by multiplying the polynomial of the message to be sent by x^{n-k}, then by carrying out the Euclidean division by the generator polynomial g(x). The rest of the division gives the redundancy.
4. For example, for polynomial $g(x) = x^3 + x^2 + 1$ and message $(1111) \dashrightarrow x^3 + x^2 + x + 1$:
$$(x^3+x^2+x+1)x^3 = (x^3+x^2+1)(x^3+x+1) + x^2 + x + 1.$$
5. The encoded message will be (1111111)

CentraleSupélec

Cyclic block linear code: Error detection with polynomials

- It is possible to obtain a matrix representation but it is more efficient to work with polynomials
  1. We have R = M G + E with each row of G being a multiple of g (x). So M G is a multiple of g (x).
  2. The rest of the Euclidean division of R by g (x) is the syndrome.
- For the example of $g(x) = x^3 + x^2 + 1$
  - ✓ Message $M = (1 \quad 0 \quad 1 \quad 0)$, after encoding $(1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1)$.
  - ✓ Without error, $r(x) = x^6 + x^4 + 1 = (x^3 + x^2 + 1)g(x)$. The rest is null, so there is no mistake.
  - ✓ With error, the received message is $R = (1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1)$.
  - ✓ $r(x) = x^6 + x^4 + x^2 + 1 = (x^3 + x^2 + 1)g(x) + x^2$. The rest is not null, so there is an error.

CentraleSupélec

Particular cyclic block linear code: Hamming code

- Family of cyclic codes with $n = 2^m - 1$ and $k = 2^m - 1 - m = n - m$ always having a minimum distance $d = 3$.

- Perfect code for $d = 3$, that is, it contains no ***unnecessary*** redundancy.

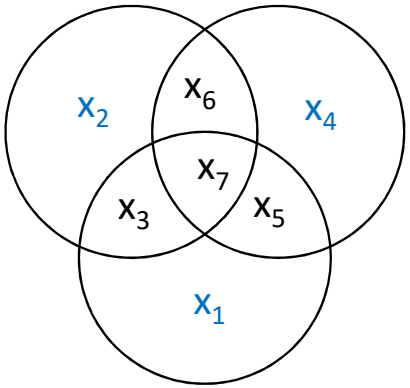
Hamming Code

- In the late 1940's Richard Hamming recognized that the further evolution of computers required greater reliability, in particular the ability to not only detect errors, but correct them. His search for error-correcting codes led to the Hamming Codes
- Hamming Codes are still widely used in computing, telecommunication, and other applications.
- Hamming Codes also applied in:
  - Data compression
  - Block Turbo Codes

A [7,4] binary Hamming Code (1)

- Let our codeword be $(x_1\ x_2\ \ldots\ x_7) \in F_2^7$
- x_3, x_5, x_6, x_7 are chosen according to the message (perhaps the message itself is $(x_3\ x_5\ x_6\ x_7)$).
- $x_4 := x_5 + x_6 + x_7 \pmod 2$
- $x_2 := x_3 + x_6 + x_7$
- $x_1 := x_3 + x_5 + x_7$
- Codewords

$$
\begin{array}{ccc}
(0\ 0\ 0\ 0) & \rightarrow & (0\ 0\ 0\ 0\ 0\ 0\ 0) \\
(0\ 0\ 0\ 1) & \rightarrow & (1\ 1\ 0\ 1\ 0\ 0\ 1) \\
(0\ 0\ 1\ 0) & \rightarrow & (0\ 1\ 0\ 1\ 0\ 1\ 0) \\
(0\ 0\ 1\ 1) & \rightarrow & (1\ 0\ 0\ 0\ 0\ 1\ 1) \\
(0\ 1\ 0\ 0) & \rightarrow & (1\ 0\ 0\ 1\ 1\ 0\ 0) \\
(0\ 1\ 0\ 1) & \rightarrow & (0\ 1\ 0\ 0\ 1\ 0\ 1) \\
(0\ 1\ 1\ 0) & \rightarrow & (1\ 1\ 0\ 0\ 1\ 1\ 0) \\
(0\ 1\ 1\ 1) & \rightarrow & (0\ 0\ 0\ 0\ 1\ 1\ 1) \\
\vdots & & \vdots
\end{array}
$$



A [7,4] binary Hamming Code (2)

- Let a = $x_4 + x_5 + x_6 + x_7$ (=1 iff one of these bits is in error)
- Let b = $x_2 + x_3 + x_6 + x_7$
- Let c = $x_1 + x_3 + x_5 + x_7$

- If there is an error (assuming at most one) then abc will be binary representation of the subscript of the offending bit.

- If (y1 y2 … y7) is received and abc ≠ 000, then we assume the bit abc is in error and switch it. If abc=000, we assume there were no errors (so if there are three or more errors we may recover the wrong codeword).

Parity check matrix

- A check matrix for a [7,4] binary Hamming Code:

$$L_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

- Let $\mathbf{y} = (y_1\ y_2\ \ldots\ y_n)$ be a received codeword.
- The syndrome of y is $\mathbf{S} := \mathbf{L}_r\mathbf{y}^T$. If $\mathbf{S}=0$ then there was no error. If $\mathbf{S} \neq 0$ then S is the binary representation of some integer $1 \leq t \leq n=2^r-1$ and the intended codeword is $\mathbf{x} = (y_1 \ldots y_r+1 \ldots y_n)$.

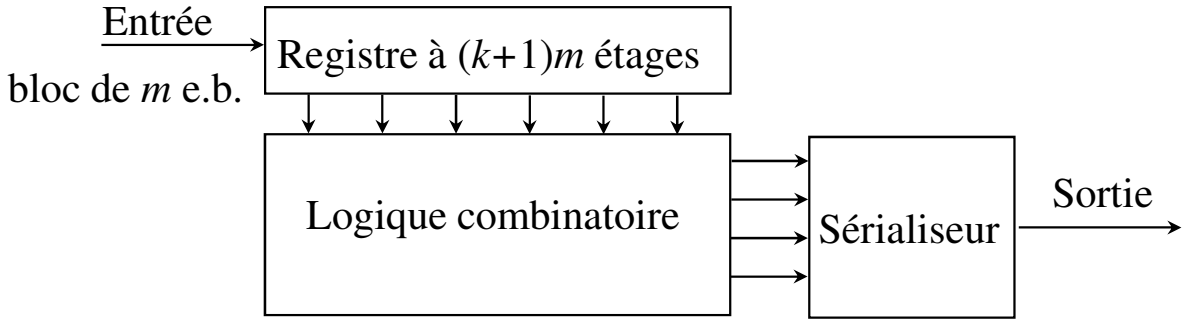
CentraleSupélec

III- Convolutional Codes

1. *Definition*
2. *Systematic Convolutional (SC) codes*
3. *Recursive Systematic Convolutional (RSC) Codes*
4. *Trellis termination*

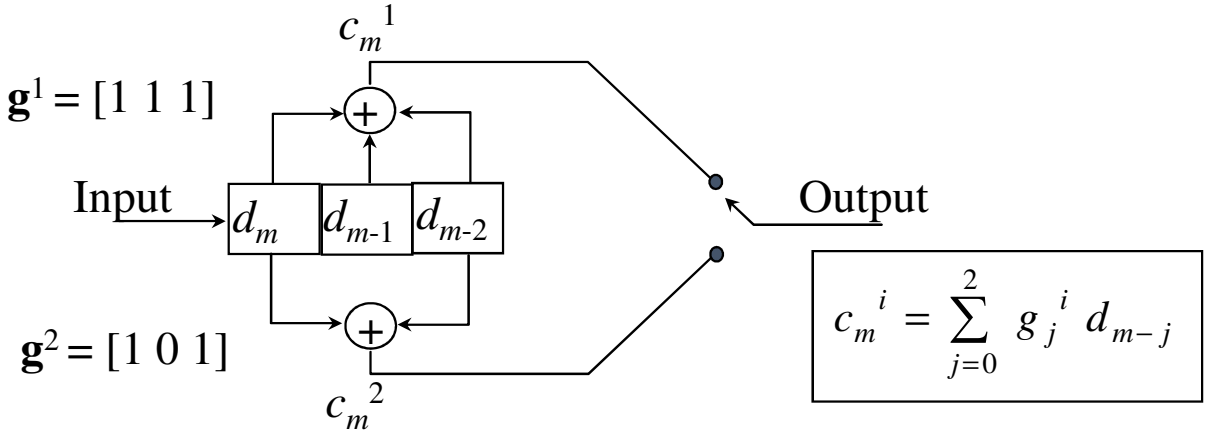
Convolutional codes

- A convolutional code is composed of
  - A register of (k+1).m stages storing (k+1) previous symbols,
  - a combinatorial logic calculating the blocks of n *bits* at the output.
- *Constraint length*: nombre of shifts needed in order to output an information block.
- Code rate: $R = m / n$.
- If m input bits are explicitly found in the block of n output bits, the code is then systematic.
- Convolutional coding is also used when the input bitstream is not separated into blocks after source coding.



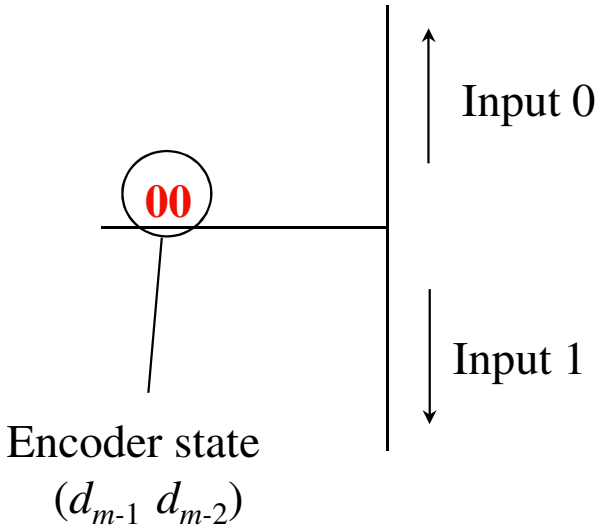
Example

- Code rate $R = 1/2$ and constraint length $(k + 1) = 3$.



- The introduction of a memory effect complicates the representation of convolutional codes.
- We then adopt the representation in the form:
  - tree,
  - state *diagram*,
  - *treillis.*

Tree representation

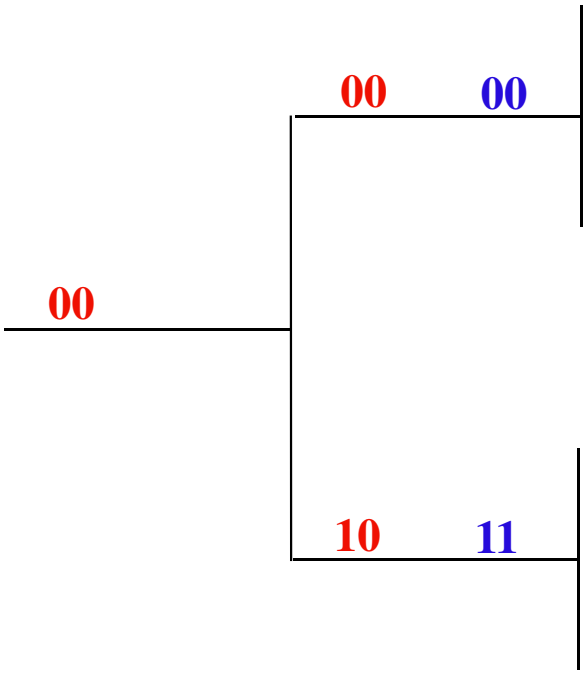


E

$t = 0$

CentraleSupélec

Tree representation

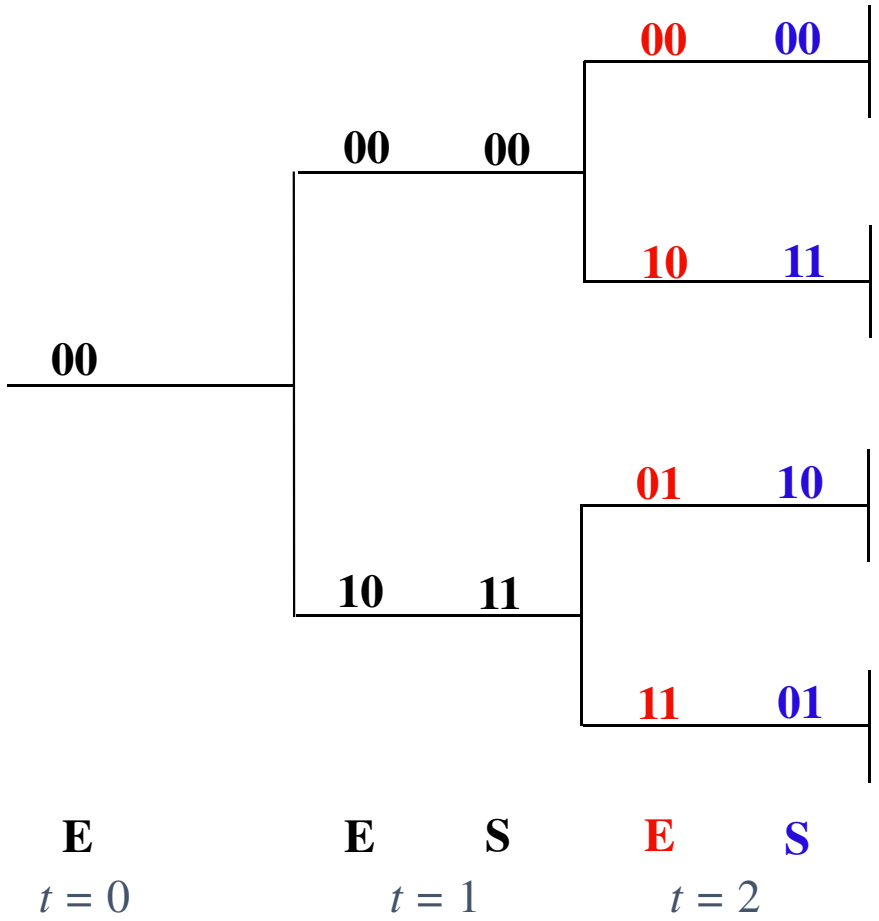


E E S

$t = 0$ $t = 1$

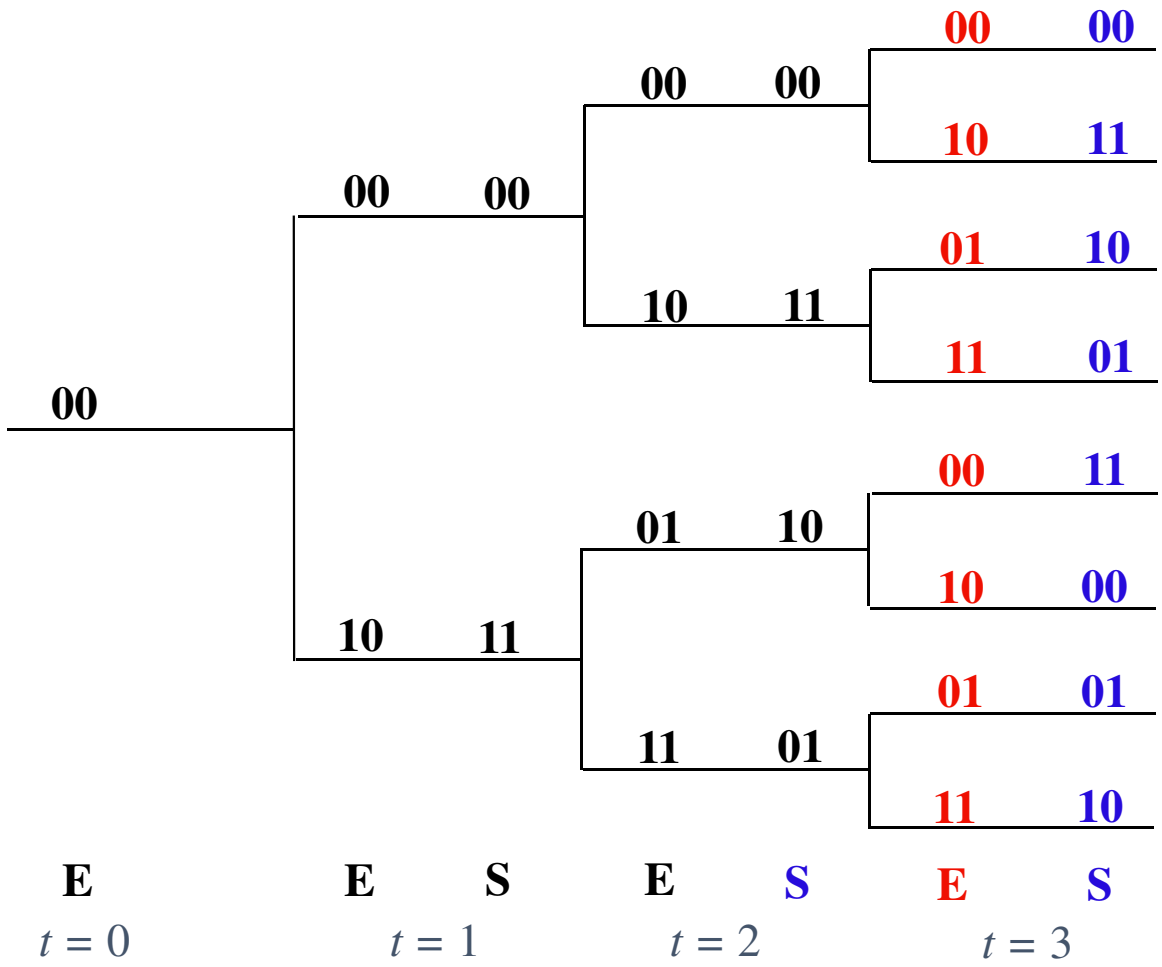
CentraleSupélec

Tree representation



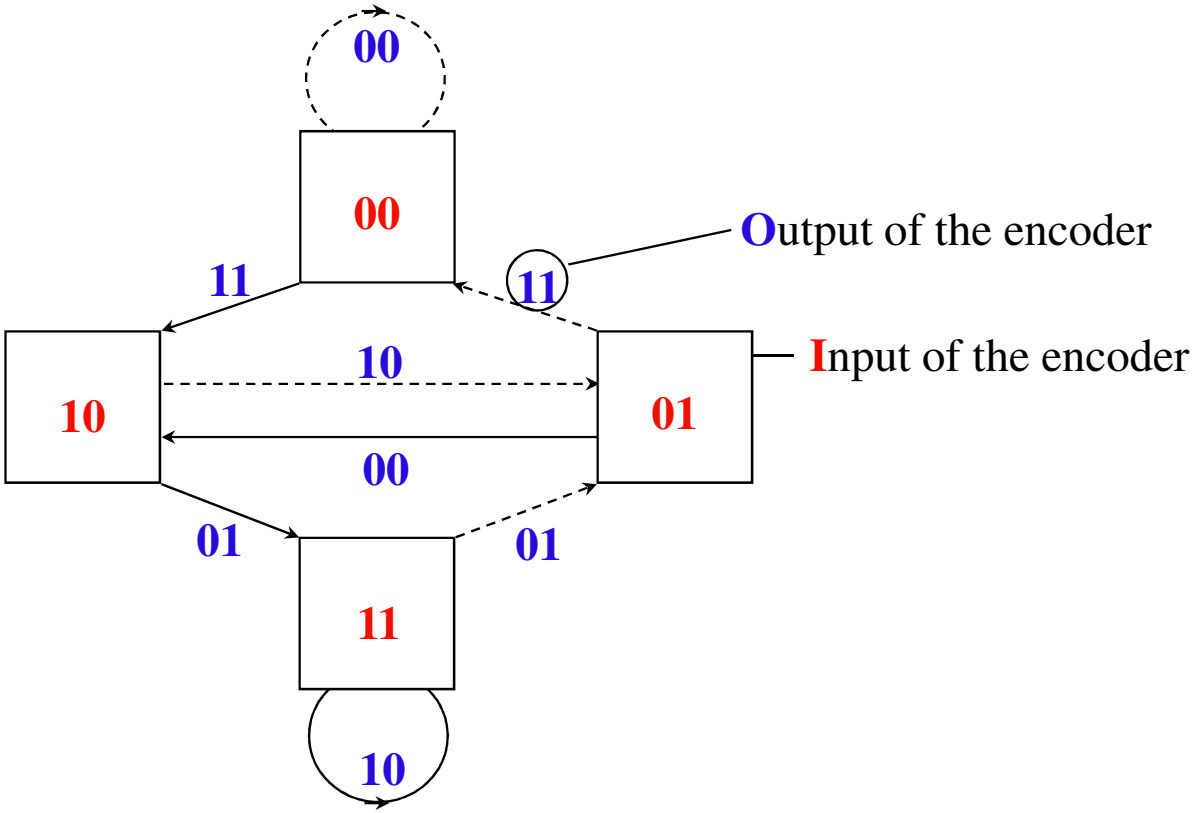
CentraleSupélec

Tree representation



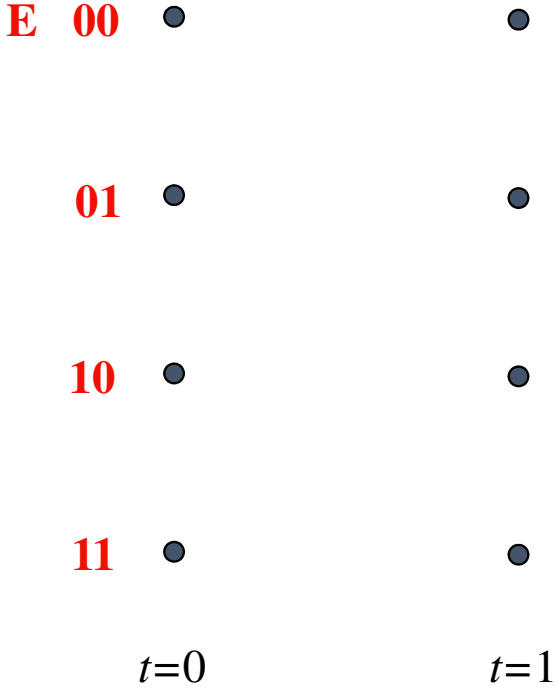
CentraleSupélec

State diagram



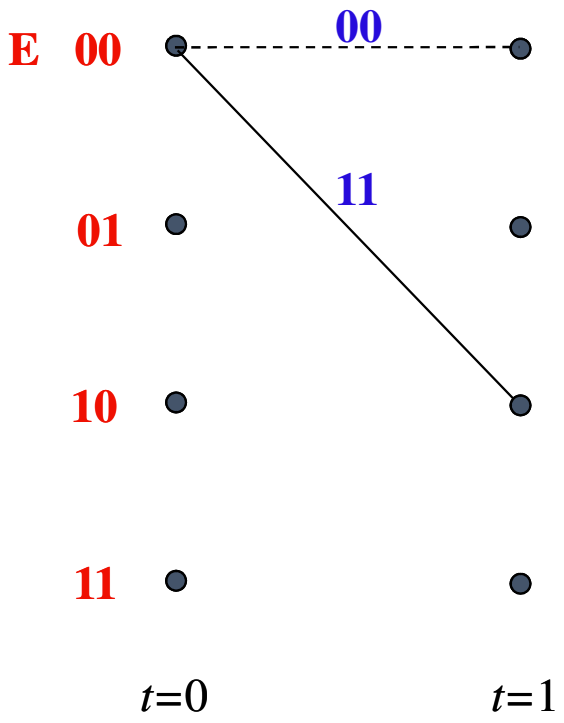
CentraleSupélec

Treillis representation



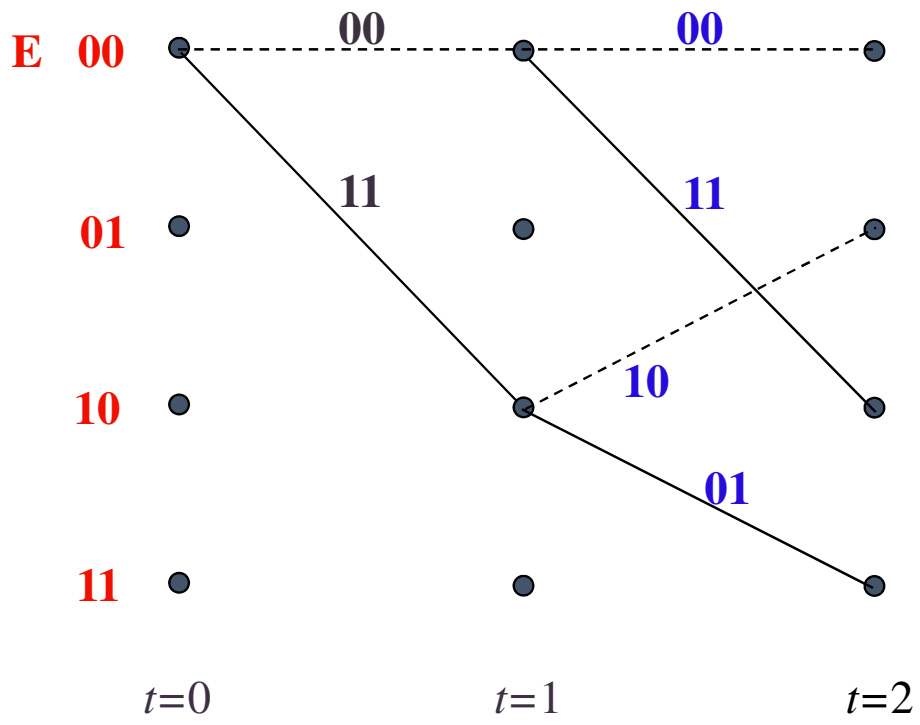
CentraleSupélec

Treillis representation



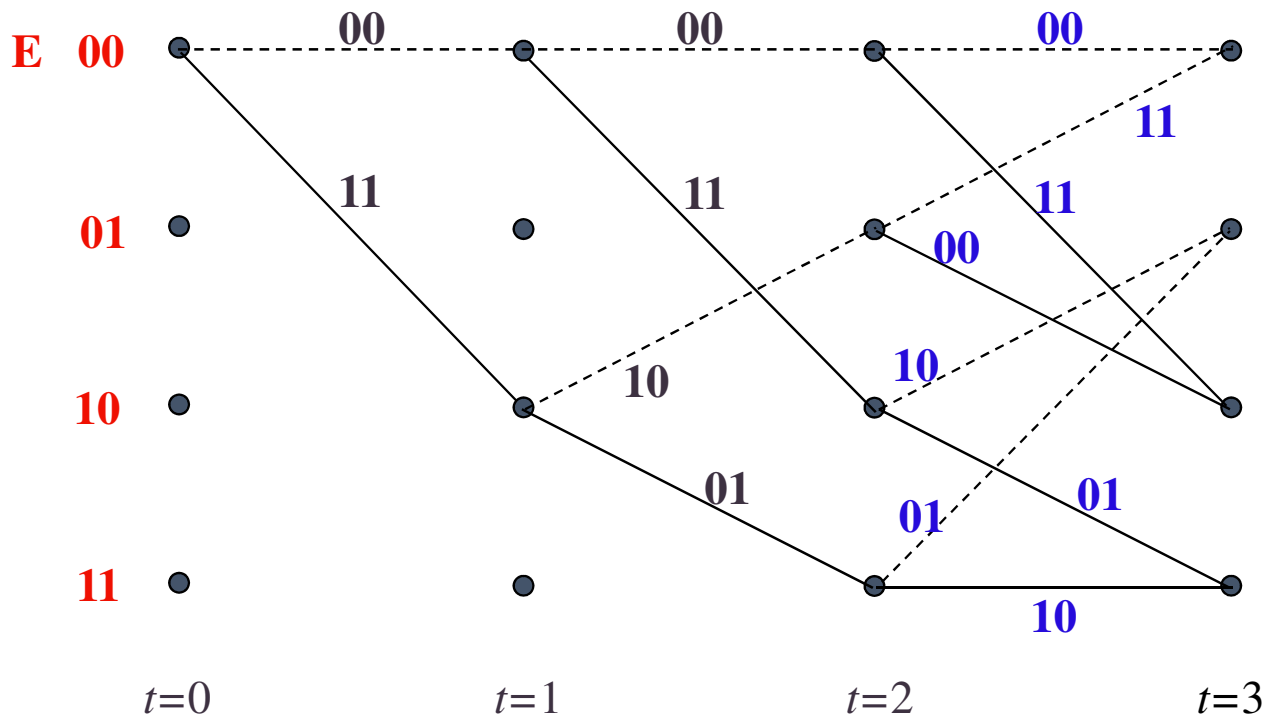
CentraleSupélec

Treillis representation



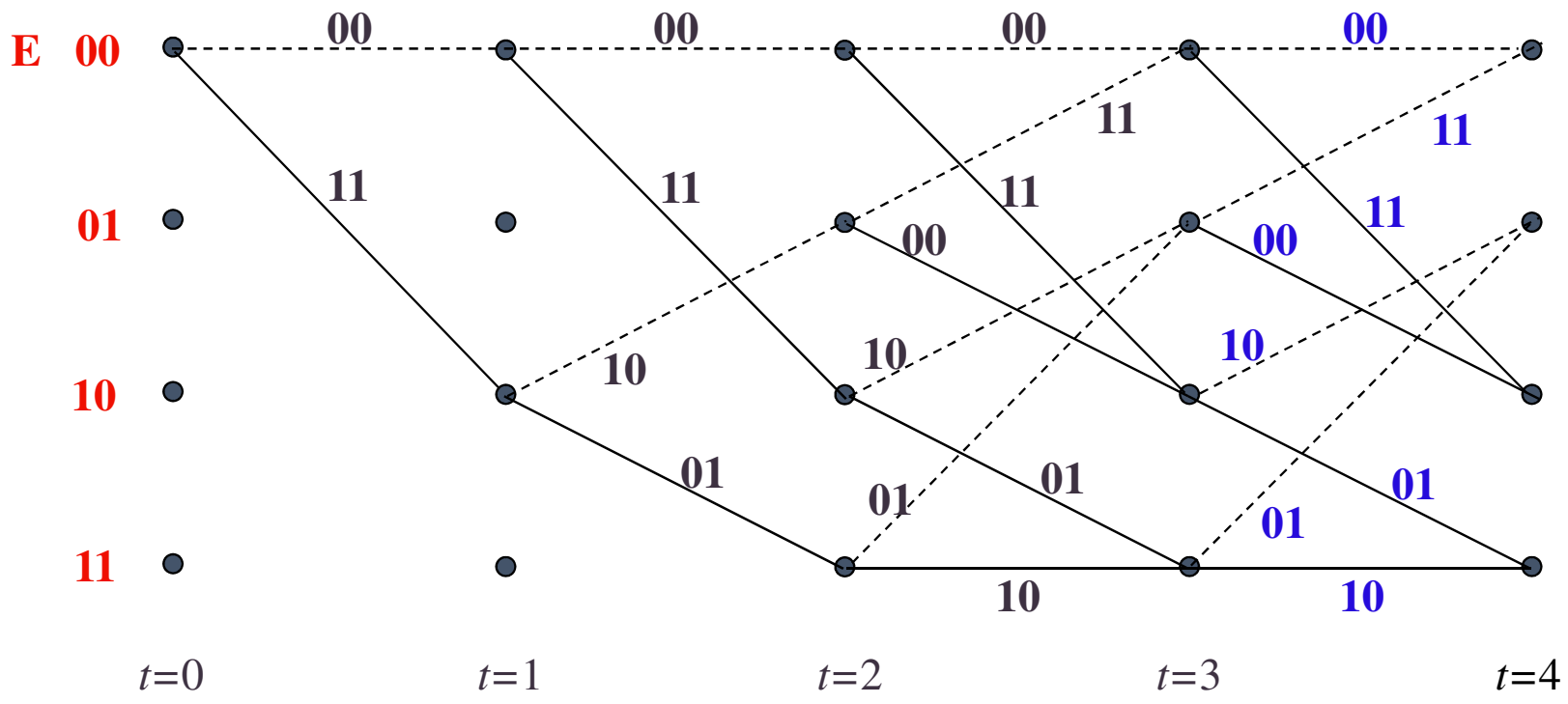
CentraleSupélec

Treillis representation



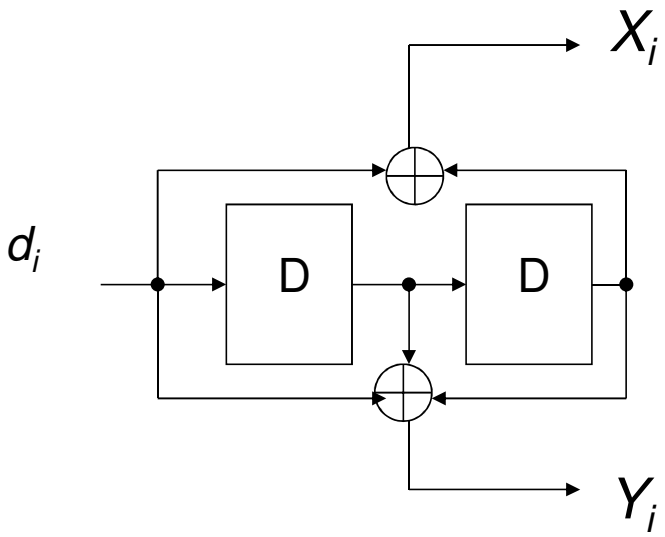
CentraleSupélec

Treillis representation



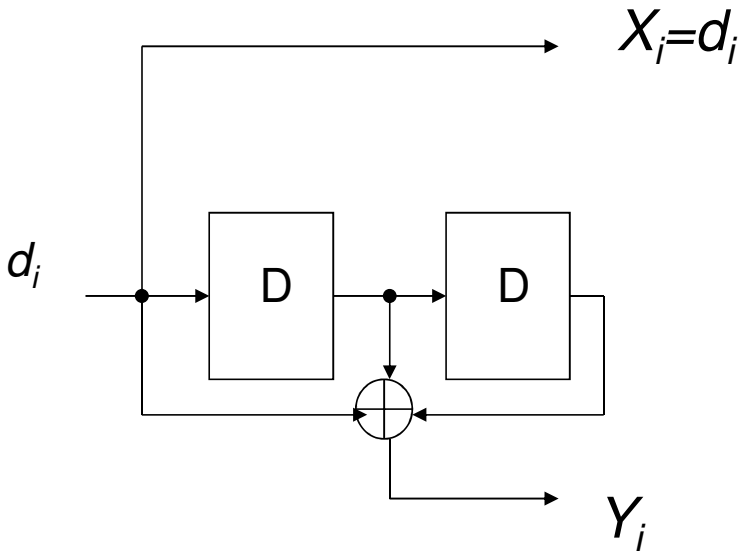
CentraleSupélec

Systematic Convolutional (SC) codes



NSC

(Non Systematic Convolutional)



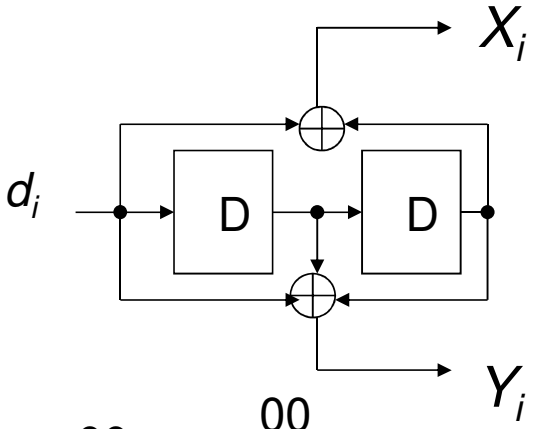
SC

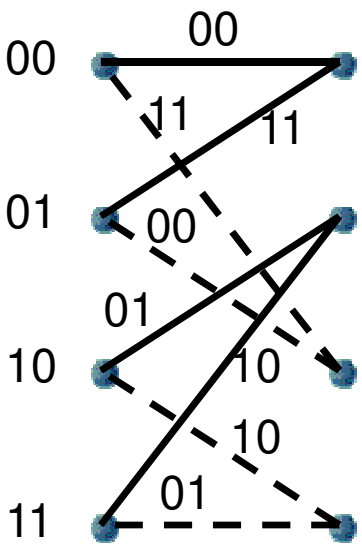
(Systematic Convolutional)

(Elias code)

SC codes: trellis diagram

NSC



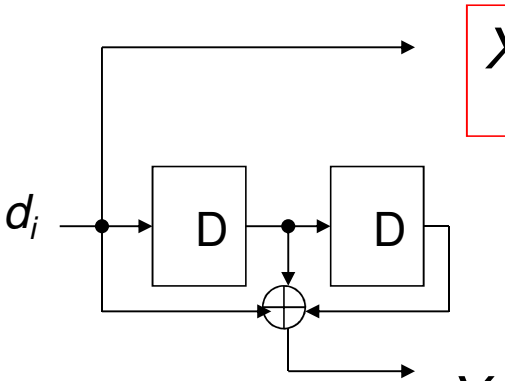


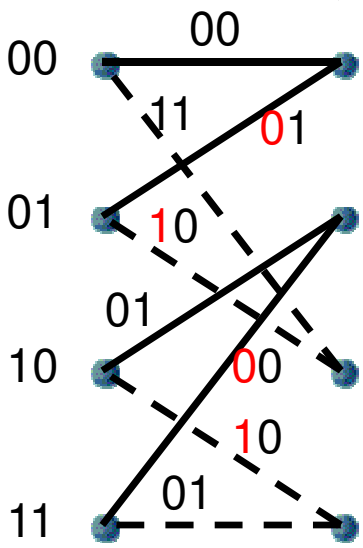
$d_f = 5$

$d_i = 0$ X_i Y_i

$d_i = 1$ X_i Y_i

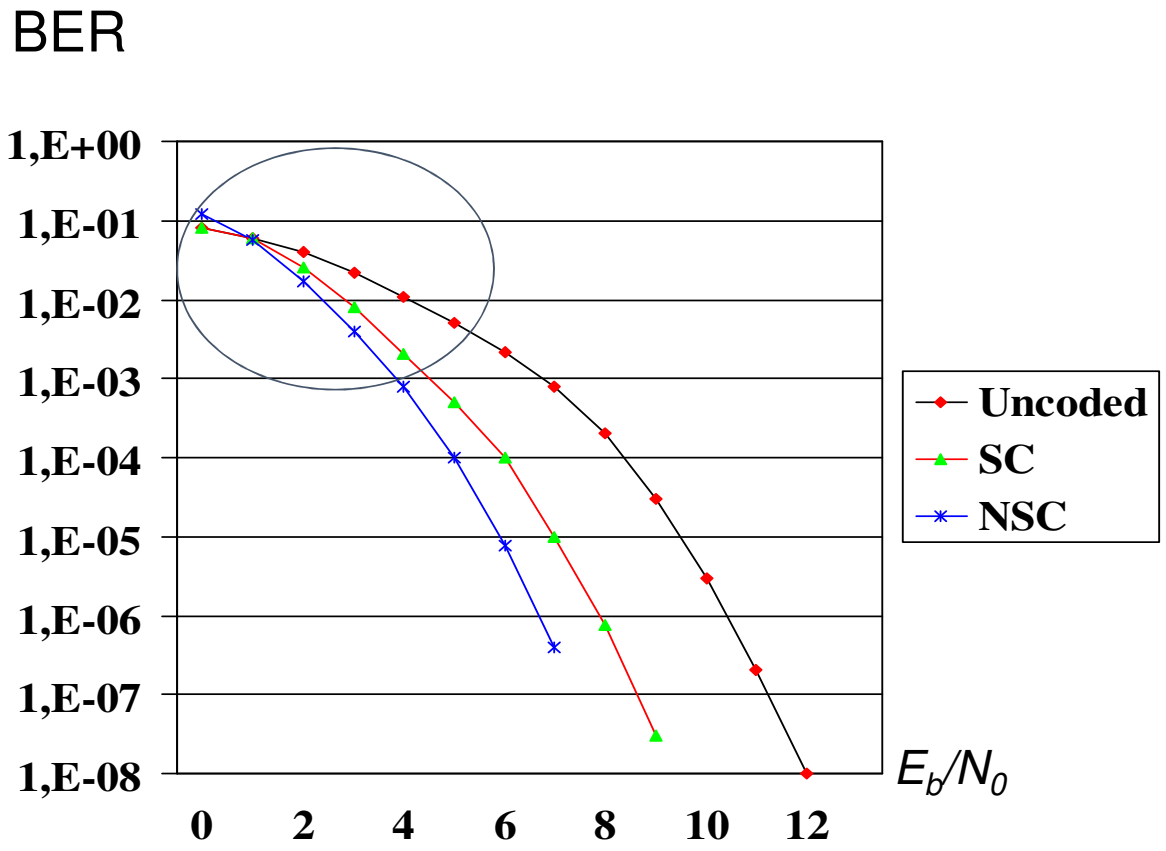
SC





$d_f = 4$

SC codes: Bit Error Rate



CentraleSupélec

SC codes: Weight Distribution

	n(*d*)	*w*(*d*) SC	*n*(*d*)	*w*(*d*) NSC
d=4	2	3	0	0
d=5	0	0	1	1
d=6	5	15	2	4
d=7	0	0	4	12
d=8	13	58	8	32
d=9	0	0	16	80
d=10	34	201	32	192
d=11	0	0	64	448
d=12	89	655	128	1024
d=13	0	0	256	2304
d=14	233	2052	512	5120

n(*d*) = number of cases

w(*d*)= total input weight

Asymptotic performance:

$$BER \le \frac{1}{2k} \sum_{d=d_{\min}}^{\infty} w(d)\,\mathrm{erfc}\left(\sqrt{Rd\frac{E_b}{N_0}}\right)$$ (Union bound)

k = information sequence length
R = coding rate
E_b/N_0 = SNR

For high distances:

$w(d)_{NSC} > w(d)_{SC}$

=> the NSC code shows better performance for high noise levels

CentraleSupélec

CentraleSupélec

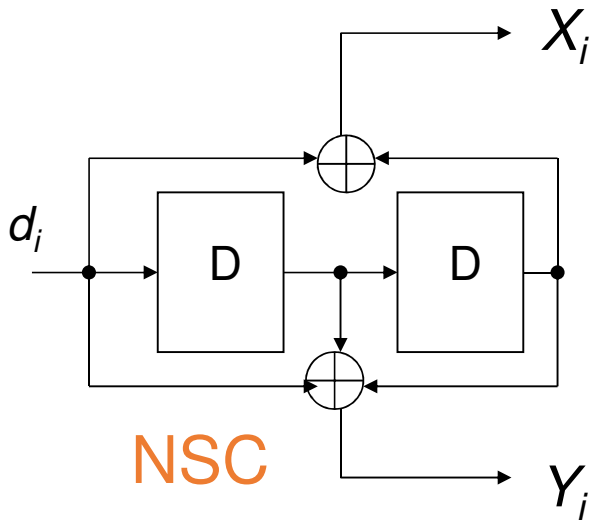
Is there any convolutional code able to combine

- the advantageous d_f of non systematic codes
- the good behaviour at very low SNRs

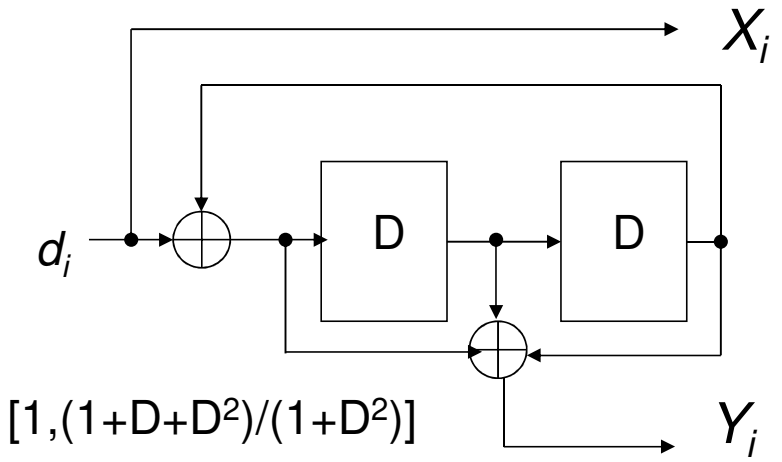
????

The answer is: yes

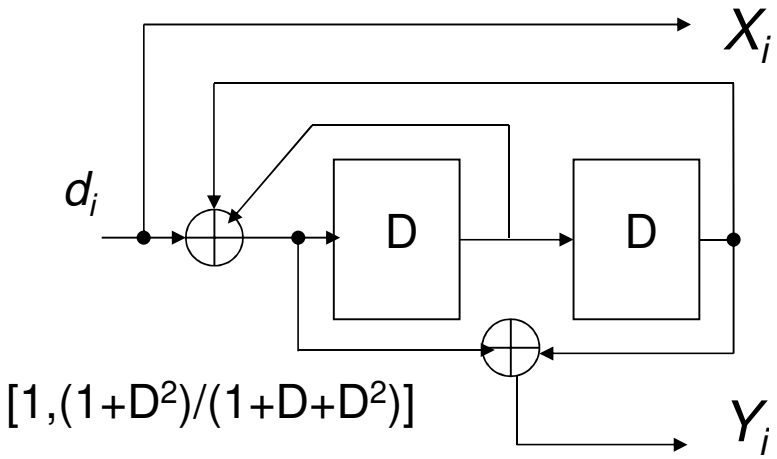
Recursive Systematic Convolutional (RSC) codes



$[(1+D^2),(1+D+D^2)]$



$[1,(1+D+D^2)/(1+D^2)]$

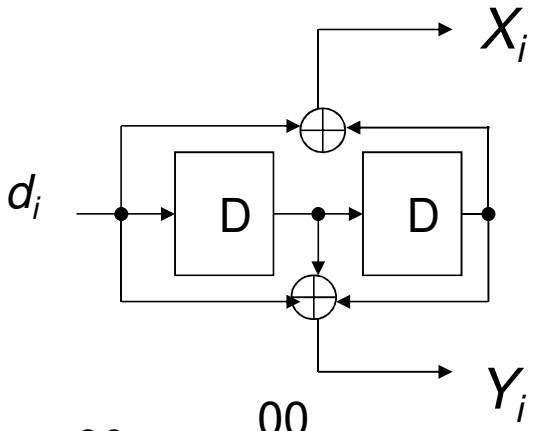


$[1,(1+D^2)/(1+D+D^2)]$

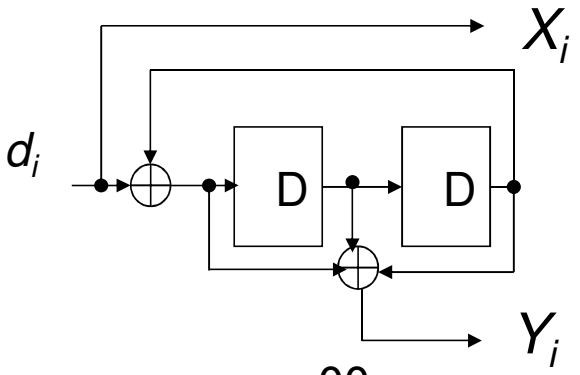
CentraleSupélec

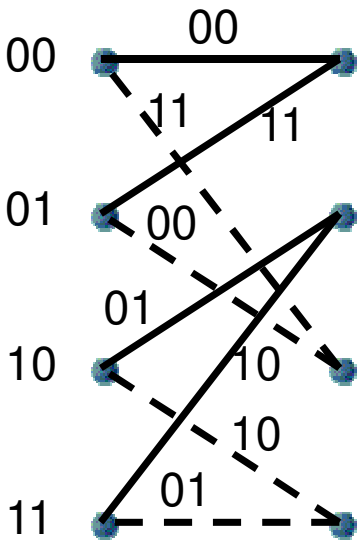
RSC codes: Trellis diagram

NSC



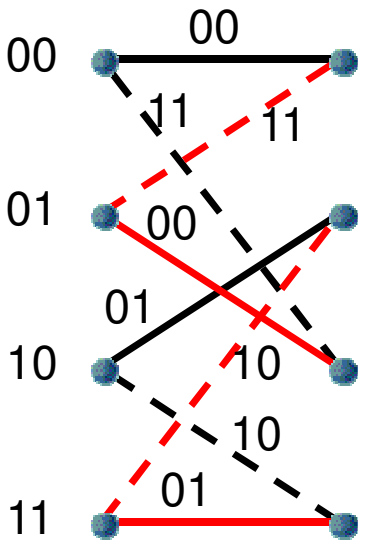
RSC





$d_i = 0$ X_i Y_i

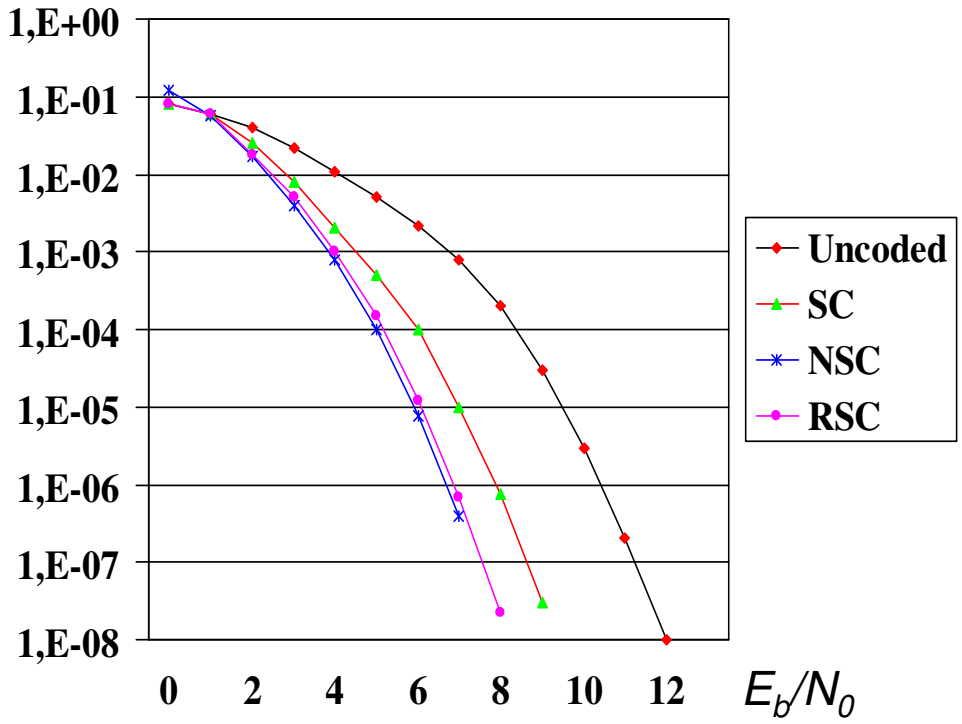
$d_i = 1$ X_i Y_i



CentraleSupélec

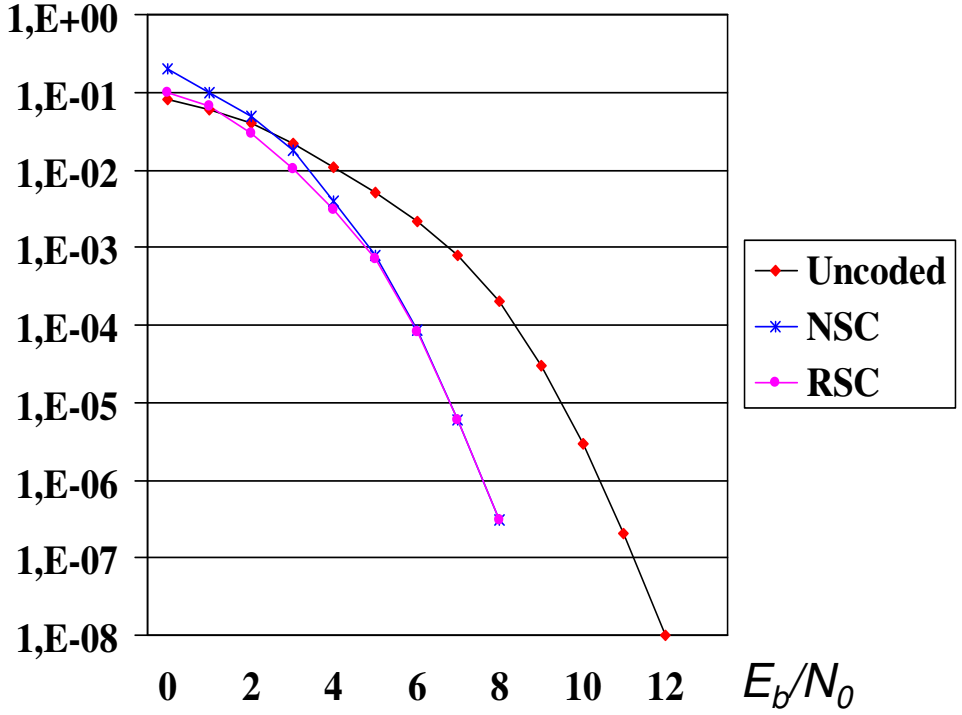
RSC codes: Bit Error Rate

BER



BER

Coding rate =3/4



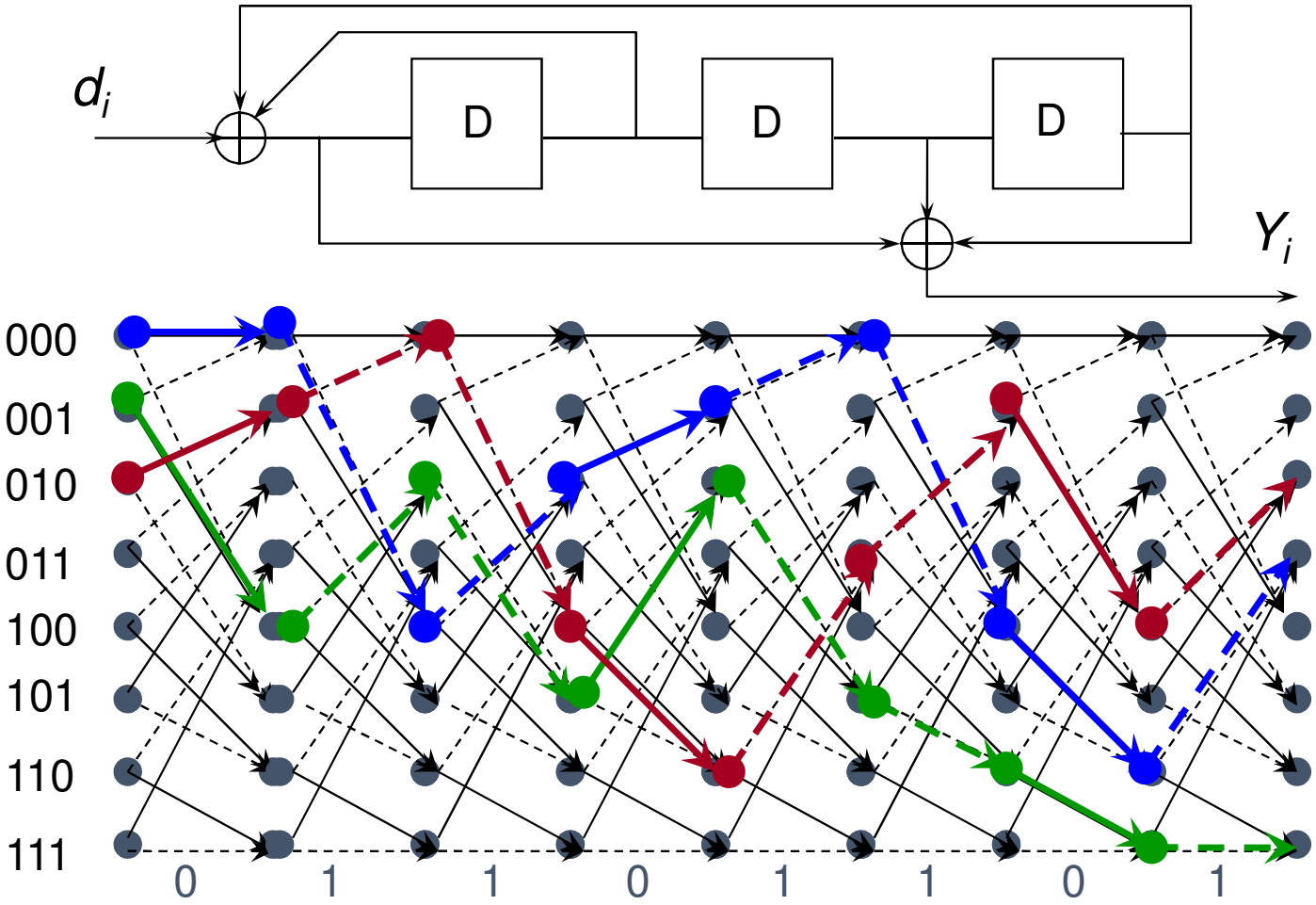
CentraleSupélec

Circular Recursive Systematic Convolutional (CRSC) Codes: Trellis termination

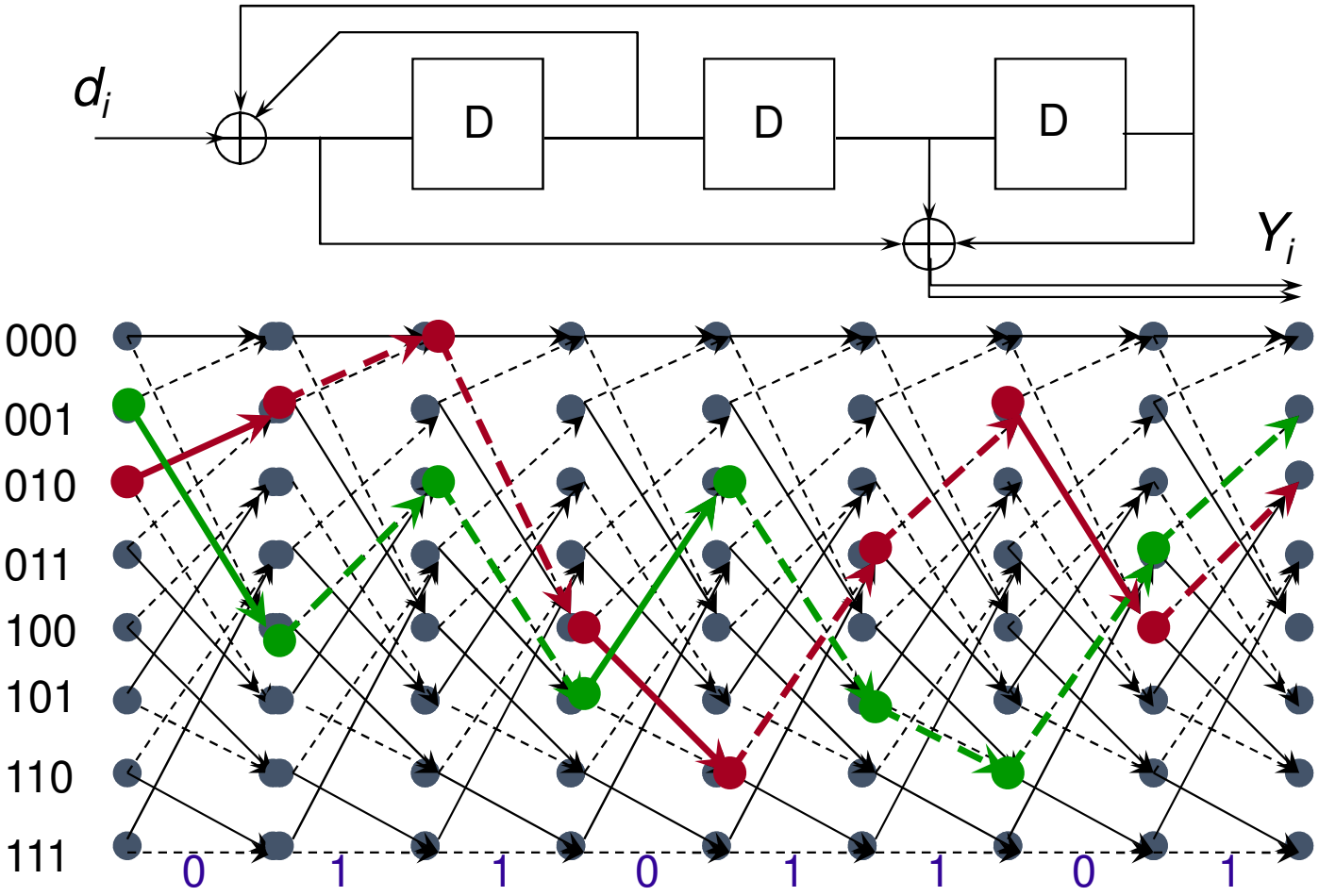
In many applications, block coding is required: How to transform a convolutional code into a block code?

- Inserting tail bits: encoder state returns to 0
  - ☺ easy to implement
  - ☹ the transmission of ν additional bits is needed
  - ☹ initial and final states are singular states
- Adopting circular encoding (= tail-biting)[9][10]
  - ☺ coding rate remains unchanged
  - ☺ the trellis can be regarded as a circle without any singularity
  - ☹ a precoding step is necessary

Convolutional Codes: Tail-biting



Convolutional Codes: Tail-biting



CentraleSupélec

Convolutional Codes: Tail-biting

- **It can be shown that, provided the data block length k is not a multiple of the encoder period P_{enc},**
  - **For a given information block, there is one and only one state S_c (*circulation* state) such as $S_c = S_0 = S_k$**
  - **S_c can be easily computed from**

    $$S_c = f(k \bmod P_{enc}, S_k^0)$$

    S_k^0 is the final encoder state when encoding starts from state 0

CentraleSupélec

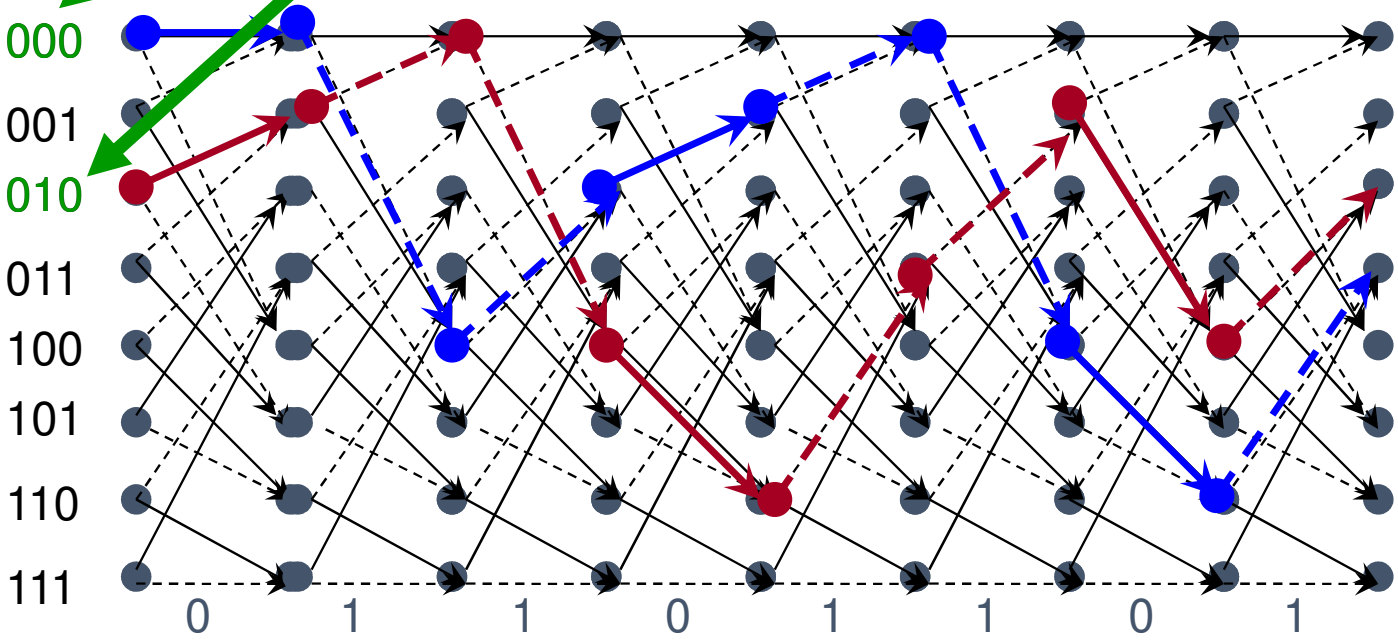
Convolutional Codes: Tail-biting

- Set the encoder to **0** (= 000)
- Encode the information block $S_k = S_k^{\mathbf{0}}$
- Compute S_c

• Pre-coding :

• Set the encoder to S_c

• Encode



CentraleSupélec

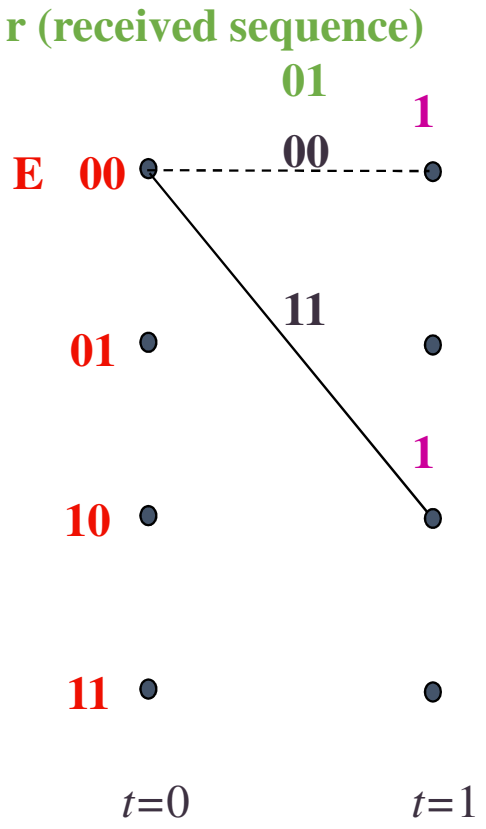
ML decoding of convolutional codes

- The decoding criterion for the convolutional codes is the same as for the codes in blocks: we will search the binary sequence **c** (corresponding to a path in the treillis) closest to the received sequence **r**, in the sense of the Hamming distance .
- This criterion is equivalent to maximum likelihood decoding
- Choose the more likely transmitted sequence

- ***Viterbi algorithm***

  - Principle : search in the treillis diagram the path corresponding to the coded sequence with the minimum differences with received sequence
  - The algorithm operates by computing a metric associated to all candidate paths
  - The metric is simply the Hamming distance between the received sequence and the sequence relative to candidate path
  - We retain as the most likely path the one whose metric is minimum.

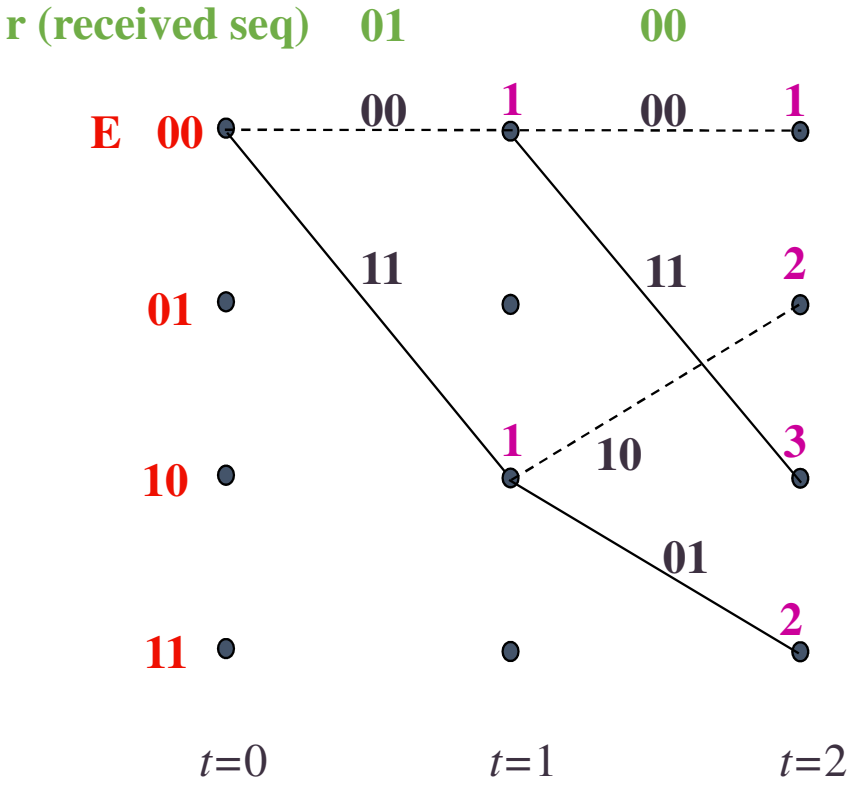
Viterbi decoding



- Example : Code de rendement $R = 1/2$ and constraint length $(k + 1) = 3$.
  - Transmitted sequence: **c** = (0000000000)
  - Received sequence: **r** = (0100010000)

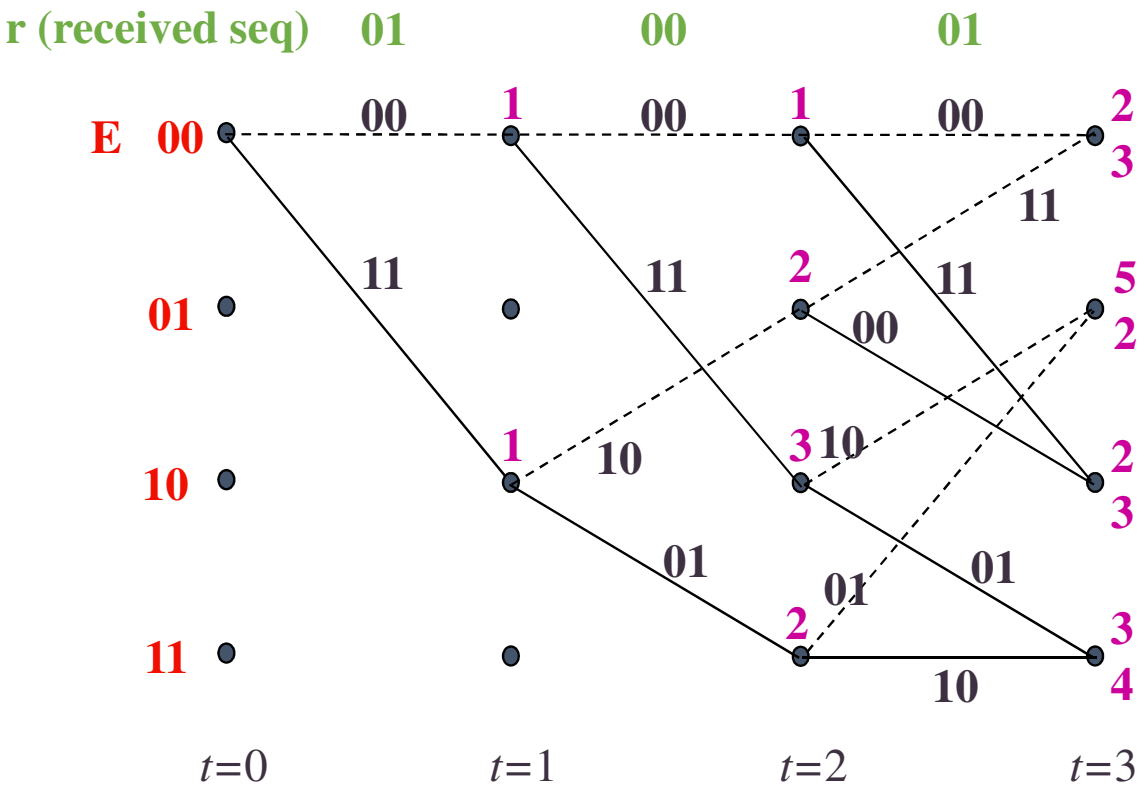
CentraleSupélec

Viterbi decoding

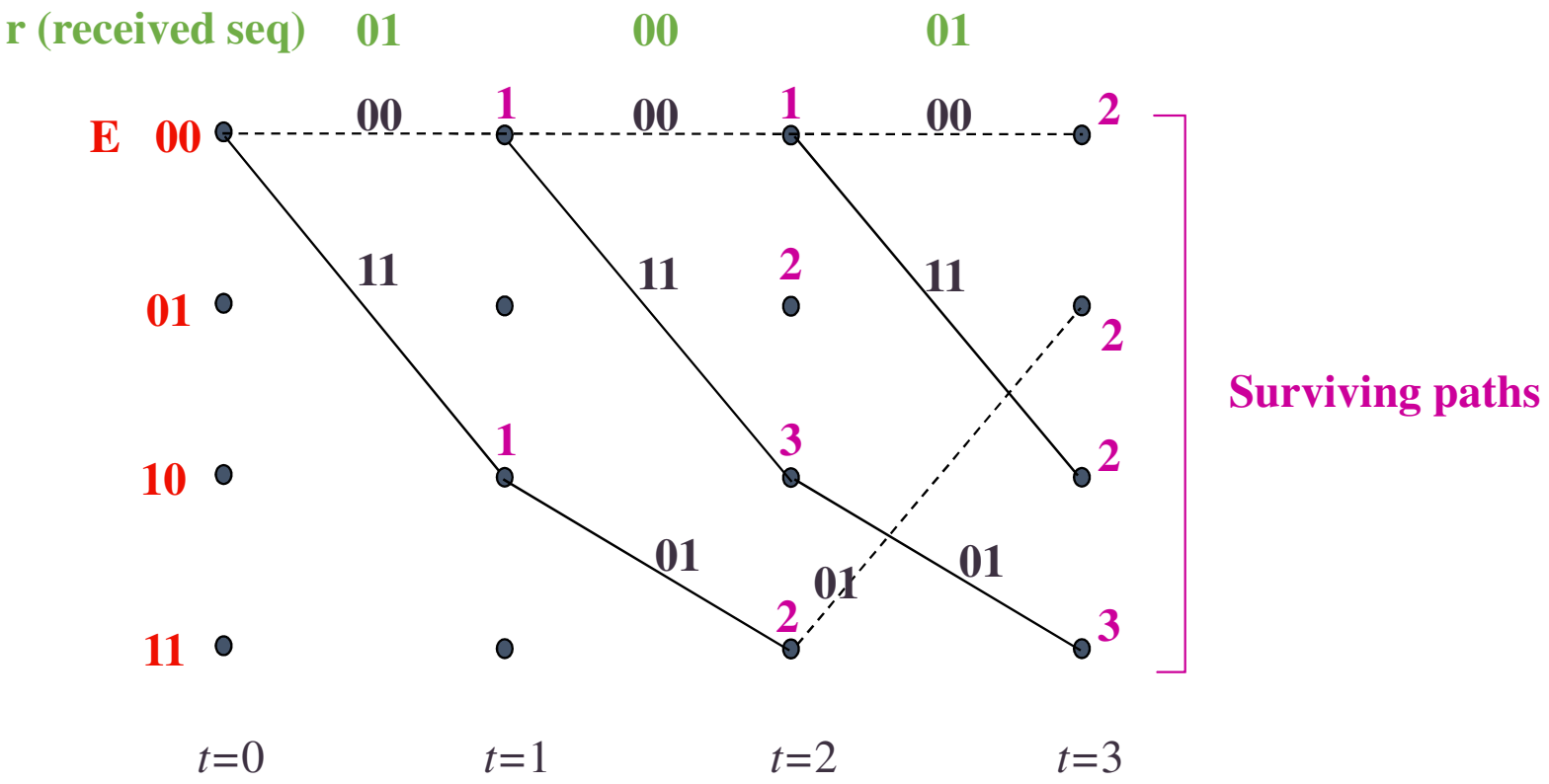


CentraleSupélec

Viterbi decoding

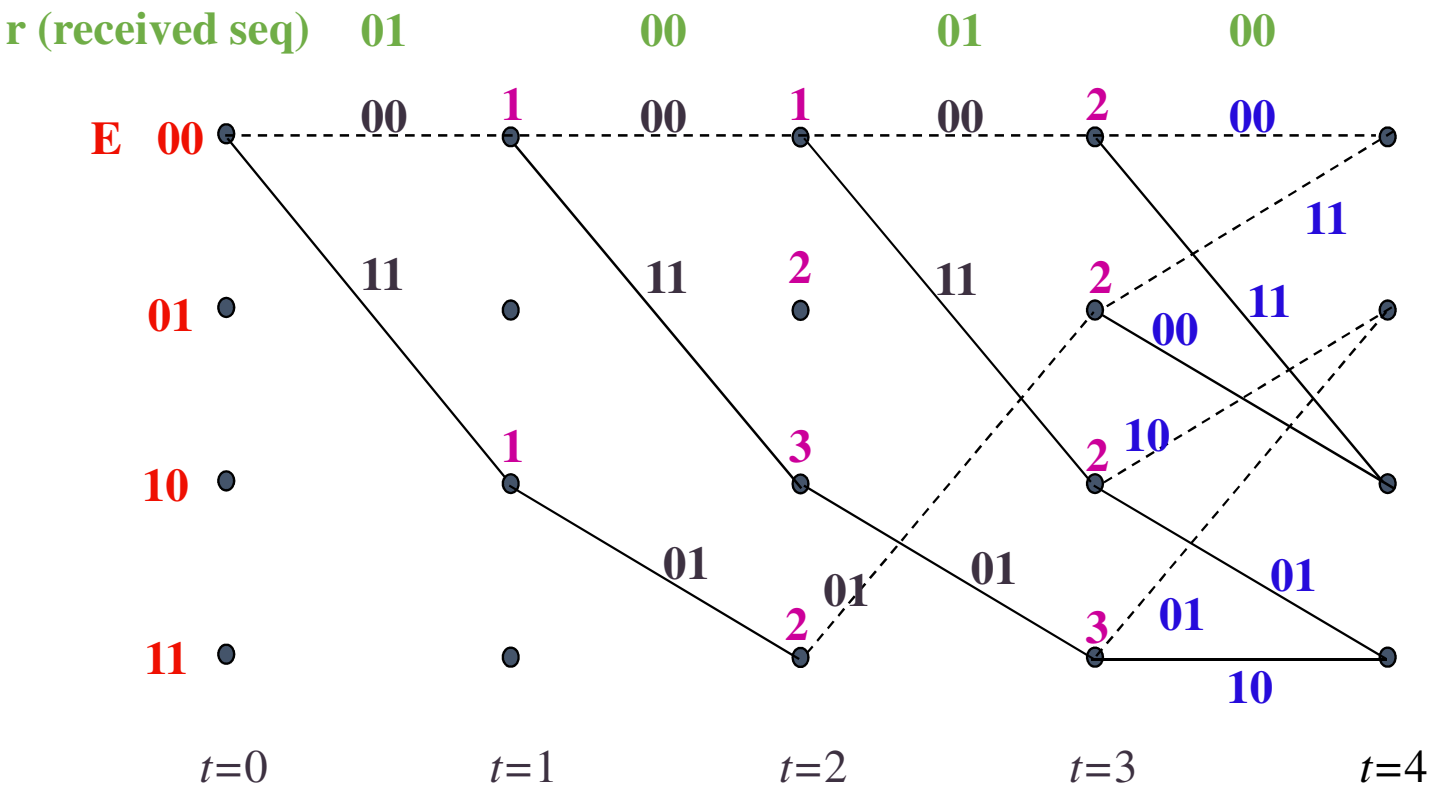


Viterbi decoding

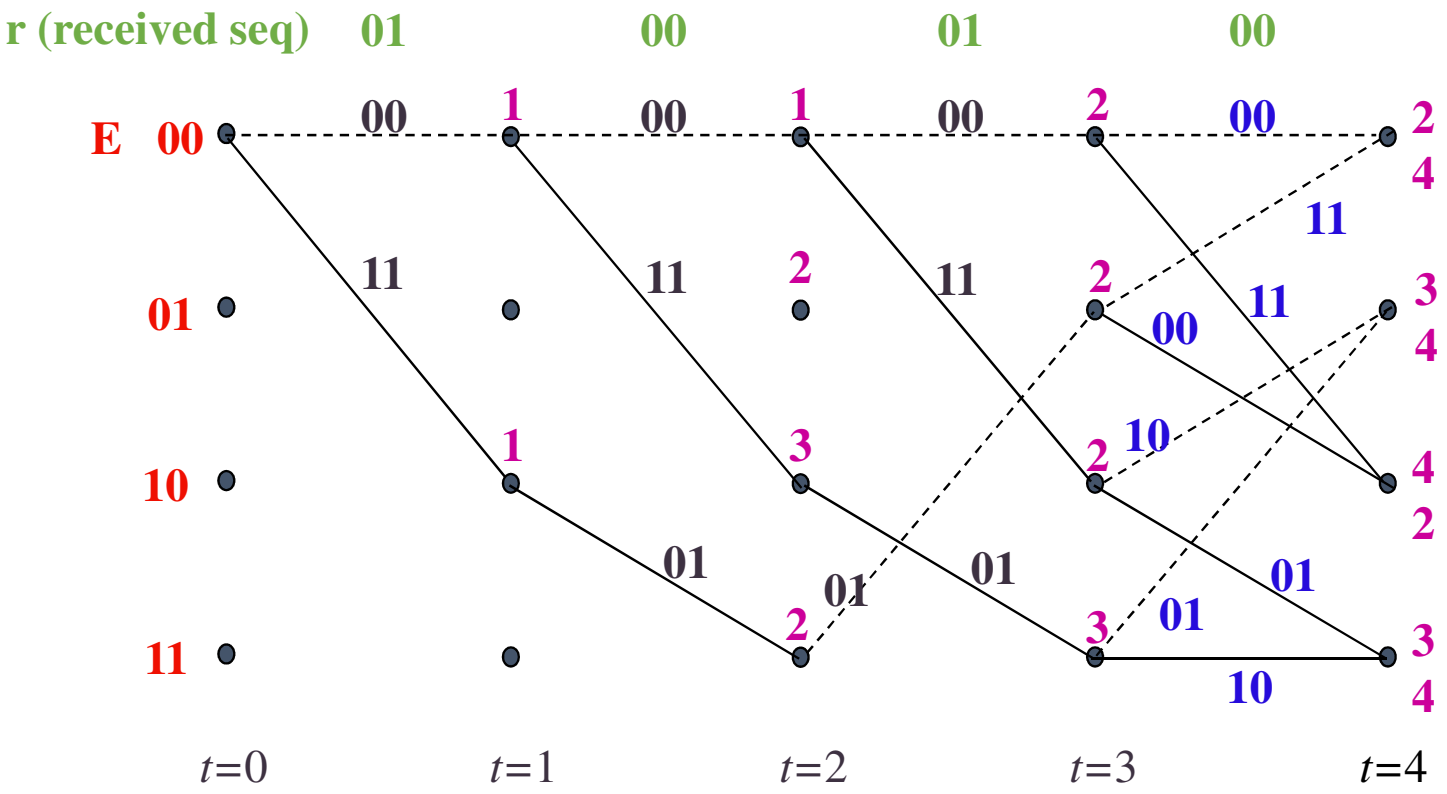


CentraleSupélec

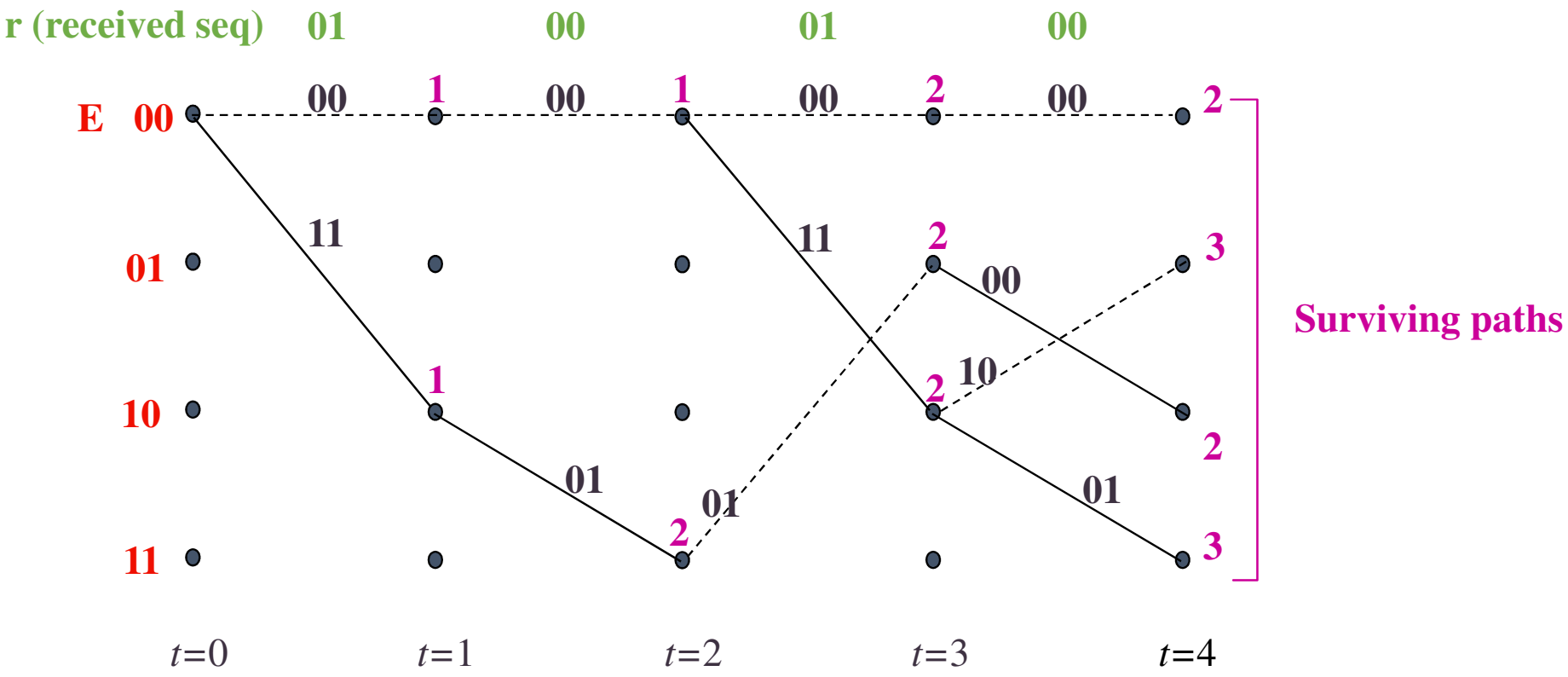
Viterbi decoding



Viterbi decoding

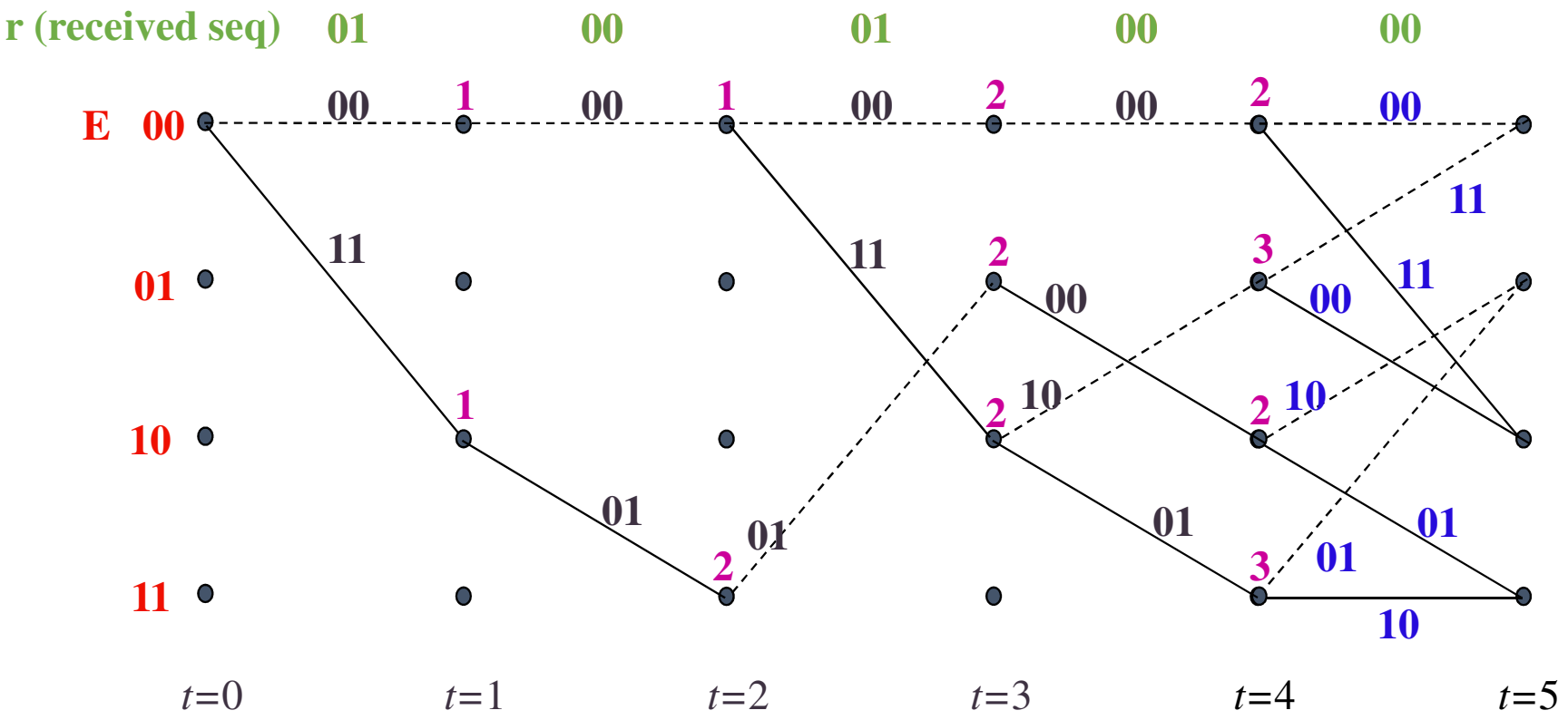


Viterbi decoding

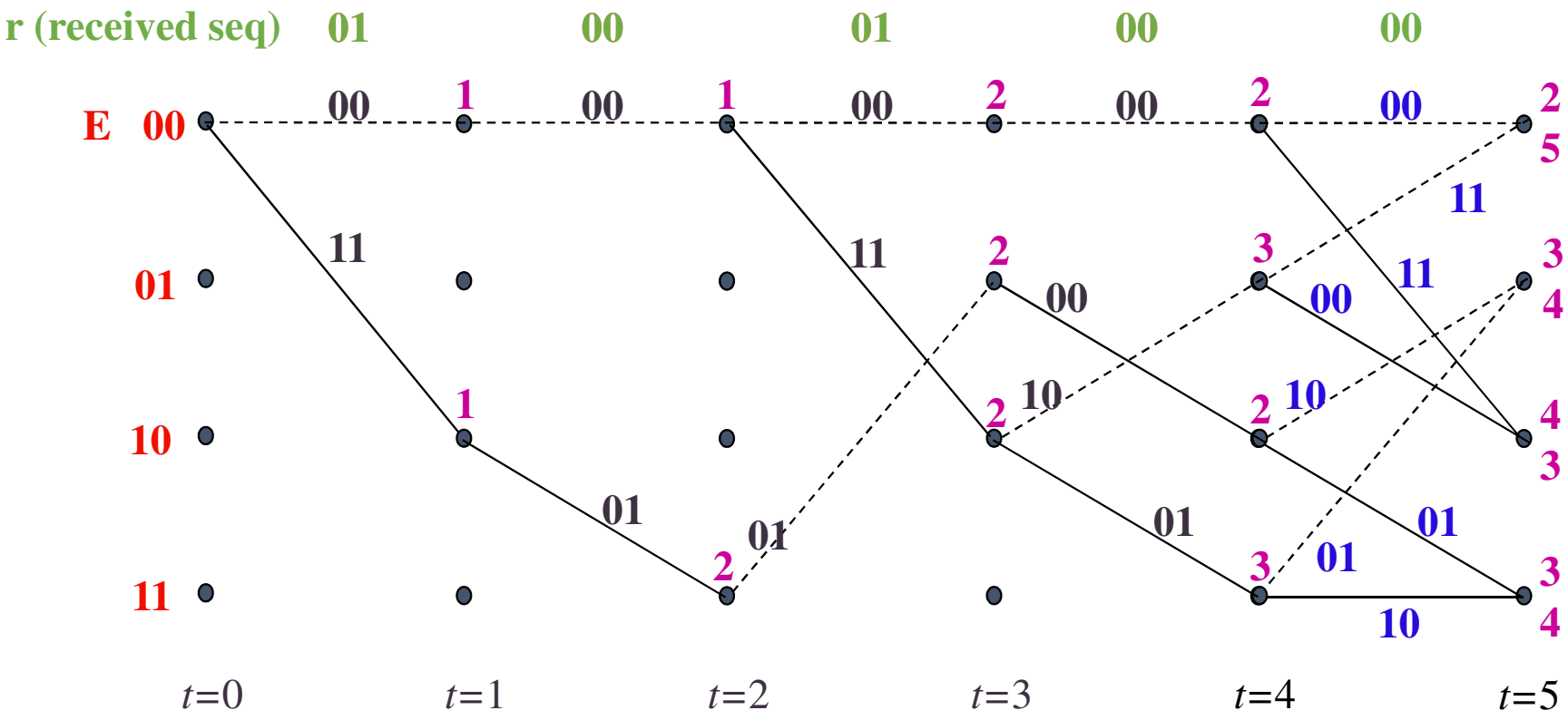


CentraleSupélec

Viterbi decoding

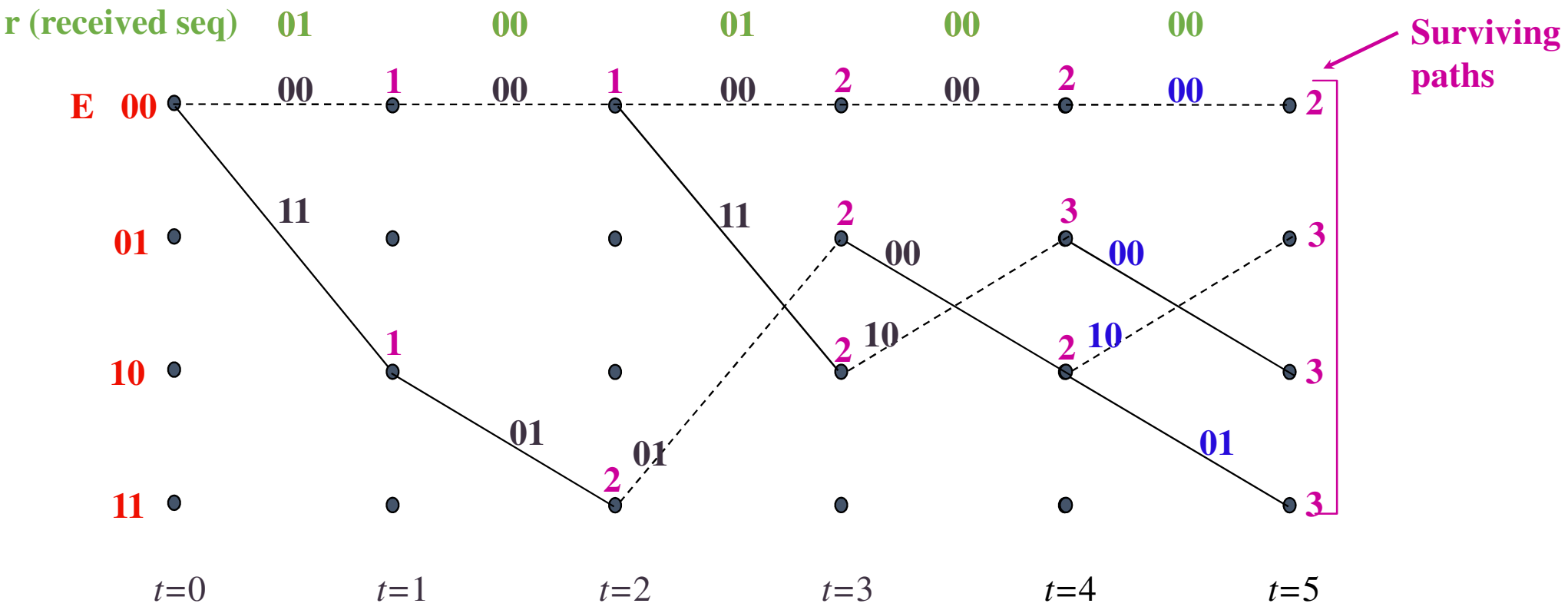


Viterbi decoding

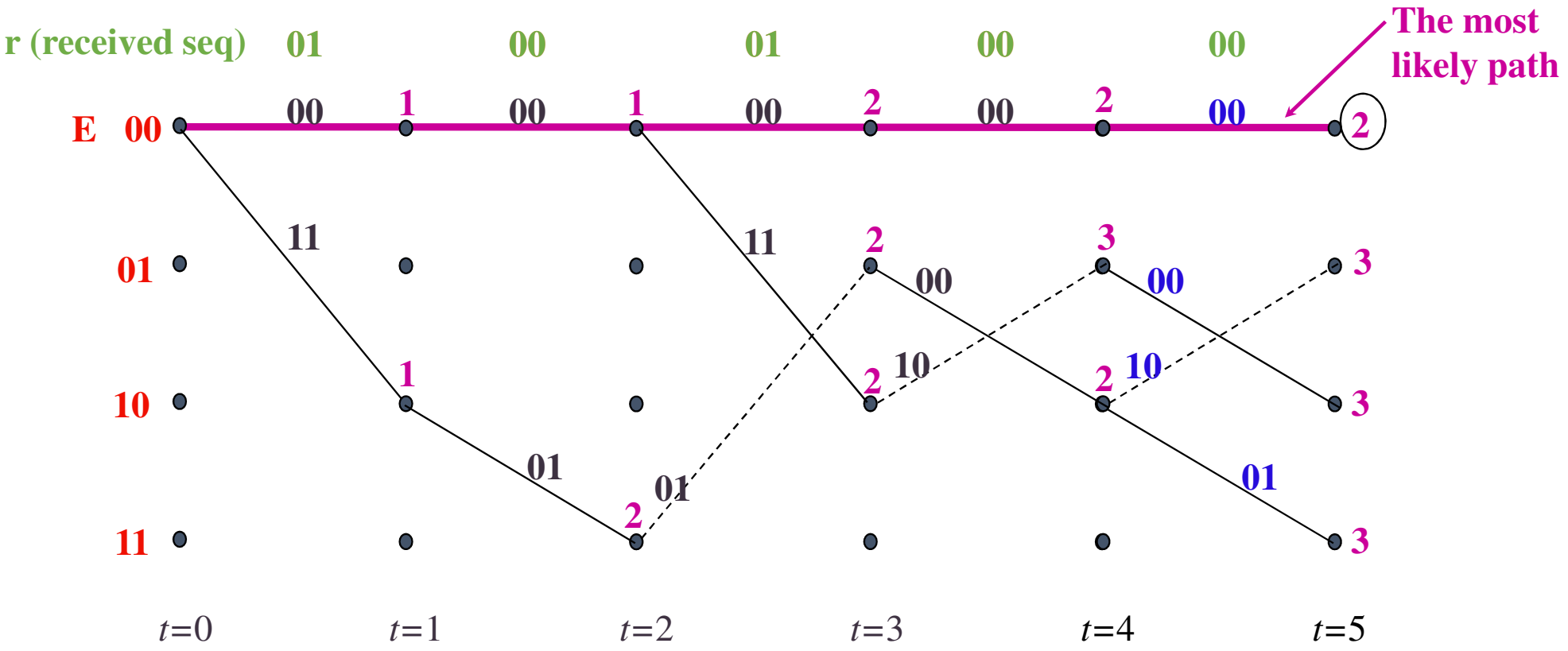


CentraleSupélec

Viterbi decoding



Viterbi decoding



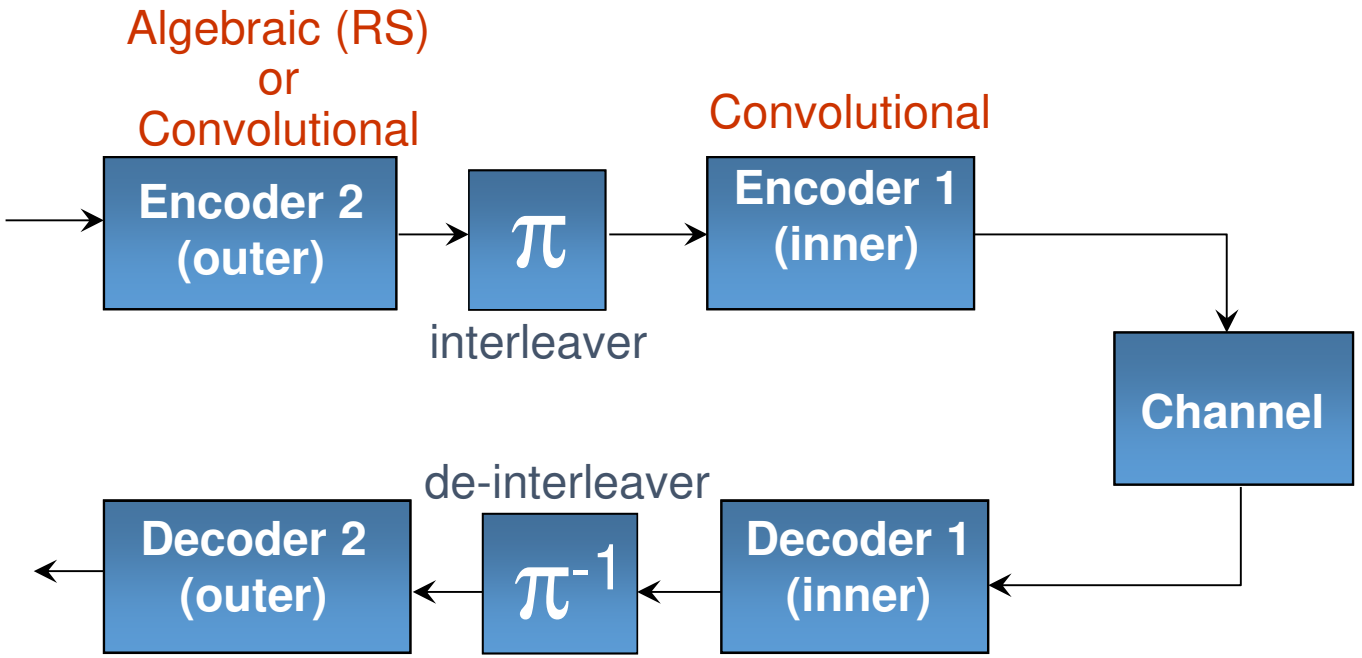
CentraleSupélec

CentraleSupélec

IV- Turbo-Codes

1. *Concatenated codes*
2. *Turbo encoder*
3. *Turbo decoding*
4. *Peformance and standards*

Concatenated Codes : divide and conquer



- First proposed by Forney in 1966 [1]
- Divide complexity between inner and outer decoding
- π and π^{-1} : de-interleaving breaks up error bursts at inner decoder output
- A standard in many communication systems : CCSDS, DVB-S1, DVB-T …

CentraleSupélec

Concatenated Codes : performance improvements

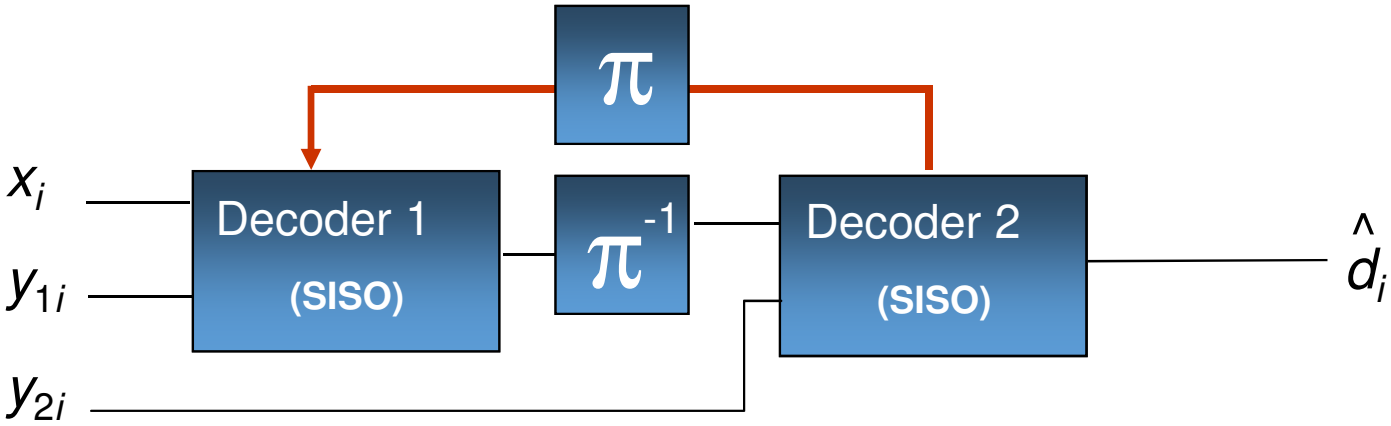
- Performance of concatenated codes suffers because inner decoder only provides “hard” (0 or 1) decisions to outer decoder

- Step 1 Soft decision decoding (~1.5 to 2.0 dB gain)
  - Soft Output Viterbi Algorithm (SOVA): Battail (1987) [2] and Hagenauer (1989) [3]
  - *Maximum a Posteriori* (MAP) or BCJR algorithm (1974) [4]

- Step2 “Turbo” decoding: Soft Iterative Decoding (<1 dB from channel capacity)
  - Turbo Codes (Parallel Concatenation of Convolutional Codes): Berrou and Glavieux (1993)
  - Turbo Product Codes: Pyndiah (1995)
  - Iterative decoding of Serially Concatenated Convolutional Codes: Benedetto *et al*. (1996) …

Iterative (“turbo”) decoding of concatenated codes: principle

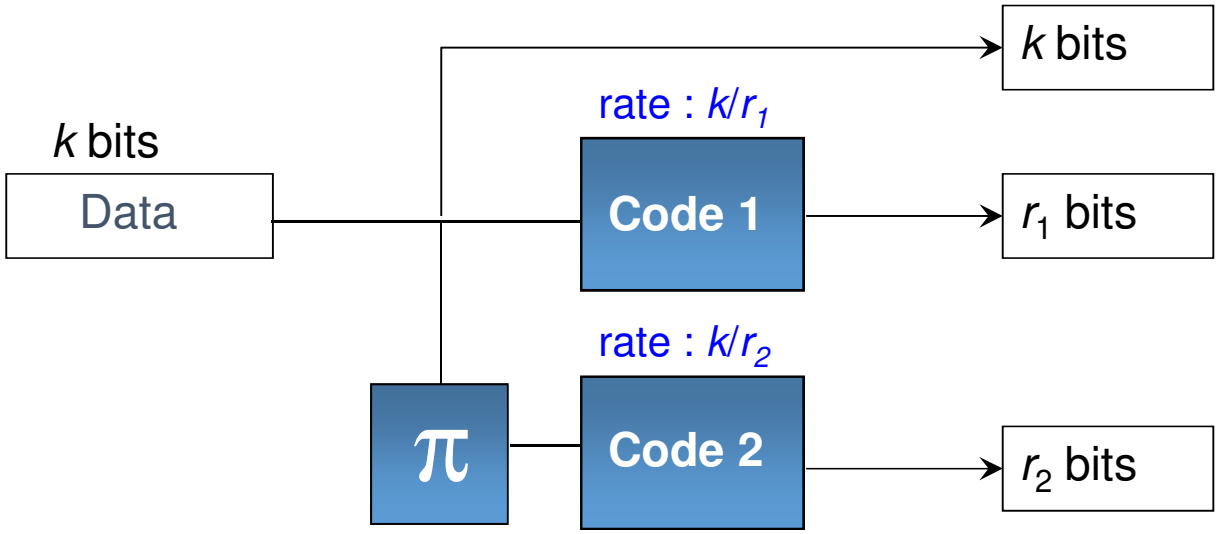
- Based on a Soft Input Soft Output (SISO) elementary decoder
- Iterative decoding process (turbo effect)



$$x_i = (2X_i\text{-}1) + n_i \qquad (X_i = d_i)$$
$$y_{1i} = (2Y_{1i}\text{-}1) + n_{1i}$$
$$y_{2i} = (2Y_{2i}\text{-}1) + n_{2i}$$

CentraleSupélec

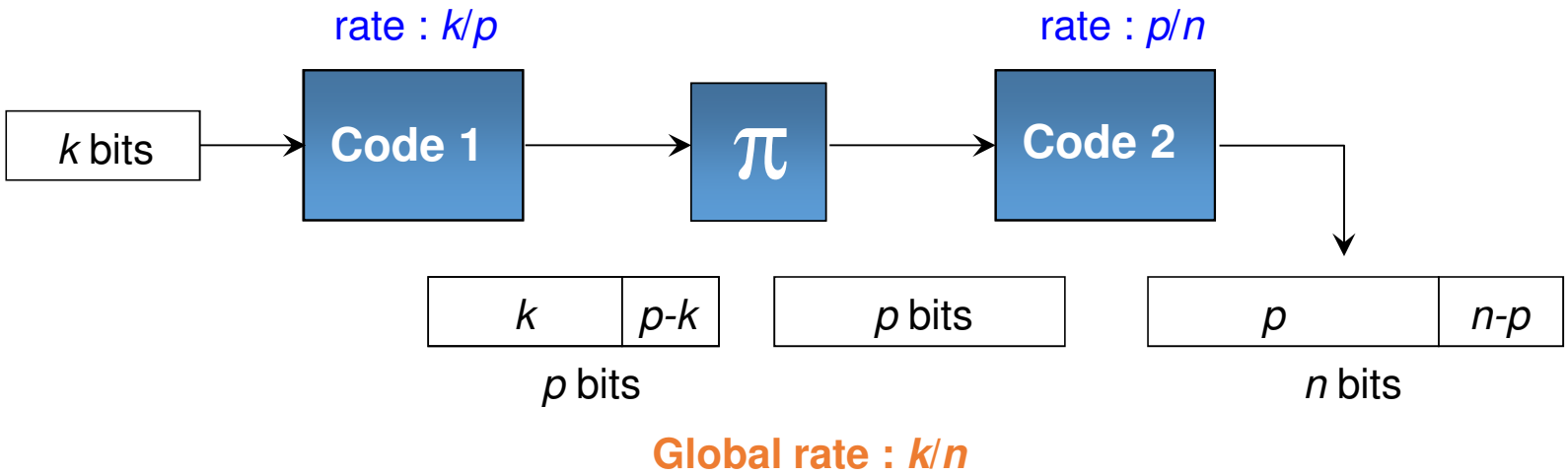
Parallel Concatenated Codes (PCC)



Global code rate:
$k/(k+r_1+r_2)$

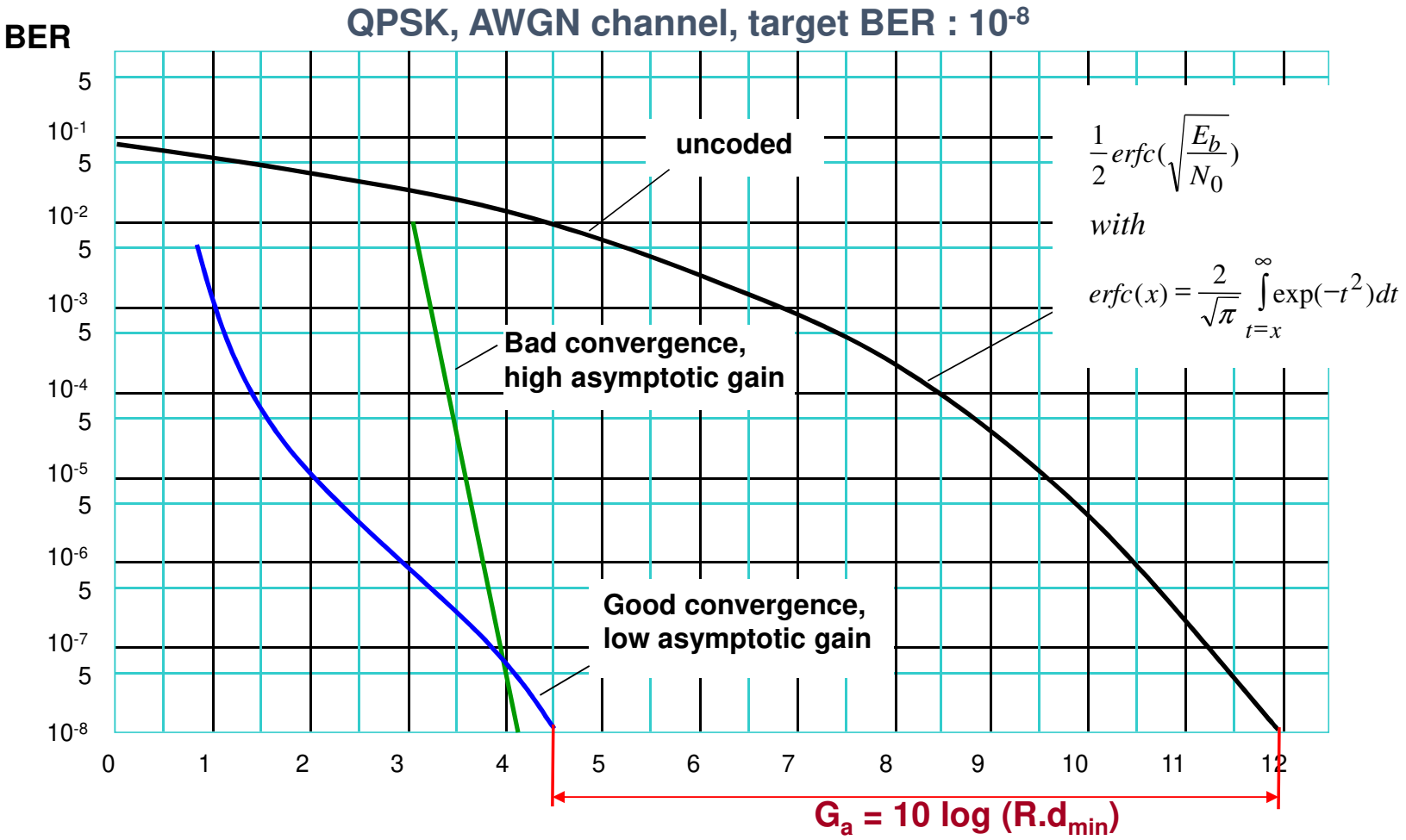
- Parallel concatenation of two (or more) systematic encoders separated by interleavers.
- Component codes can be algebraic or convolutional codes
- Historical Turbo Codes: code 1 and code 2 are two identical Recursive Systematic Convolutional (RSC) codes

Serial Concatenated Codes (SCC)



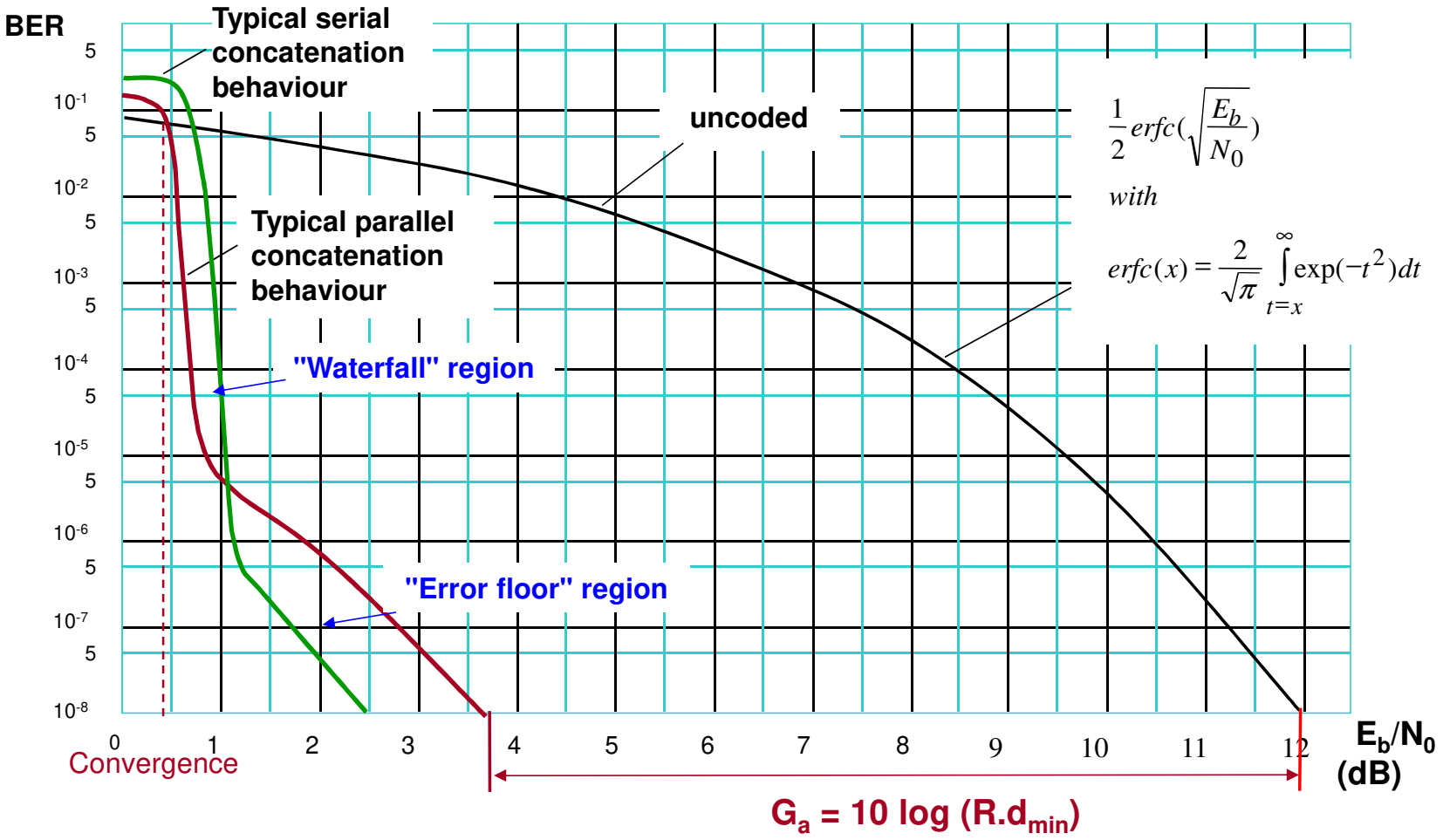
- Component codes can be algebraic or convolutional codes
- SCCC: Serial Concatenation of Convolutional codes (Hagenauer)
- Some hybrid schemes (serial -//) can also be investigated (but have not found any practical application yet)

Qualitative behavior of concatenated codes under iterative decoding



CentraleSupélec

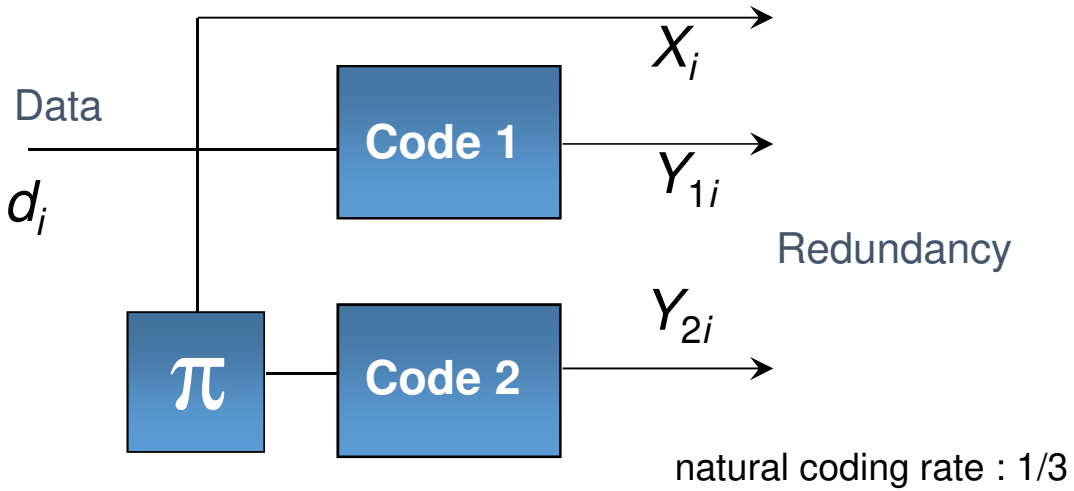
Parallel concatenation vs serial concatenation: performance comparison under iterative decoding



CentraleSupélec

Turbo encoder (1)

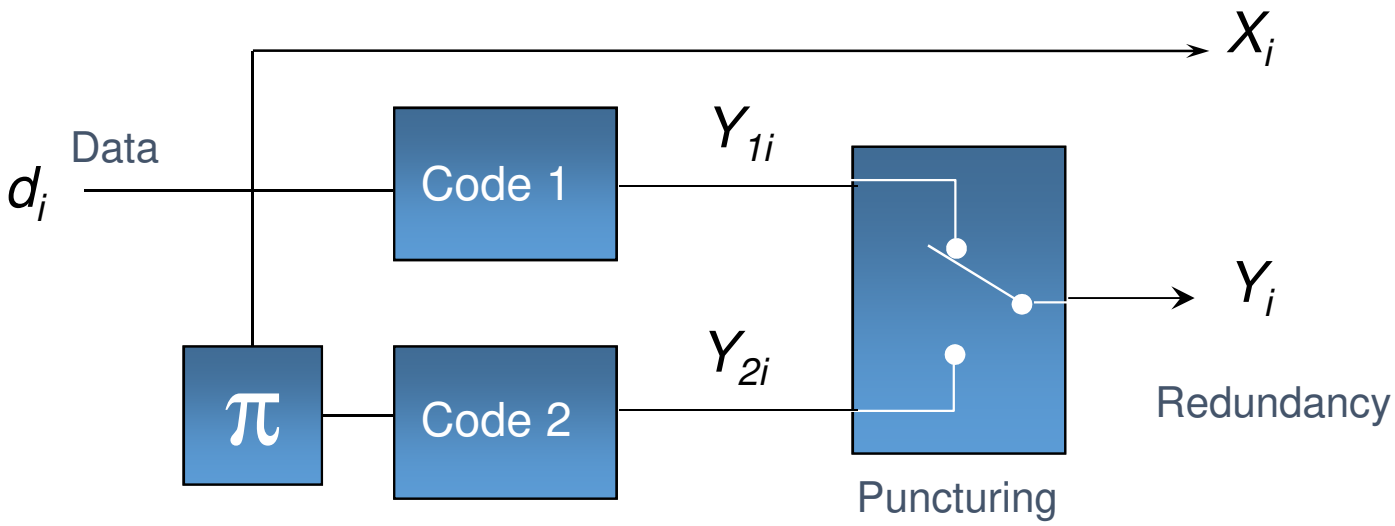
The turbo encoder involves a parallel concatenation of at least two elementary Recursive Systematic Convolutional (RSC) codes separated by an interleaver (π)



Parallel concatenation of two RSC codes
or «convolutional» turbo code

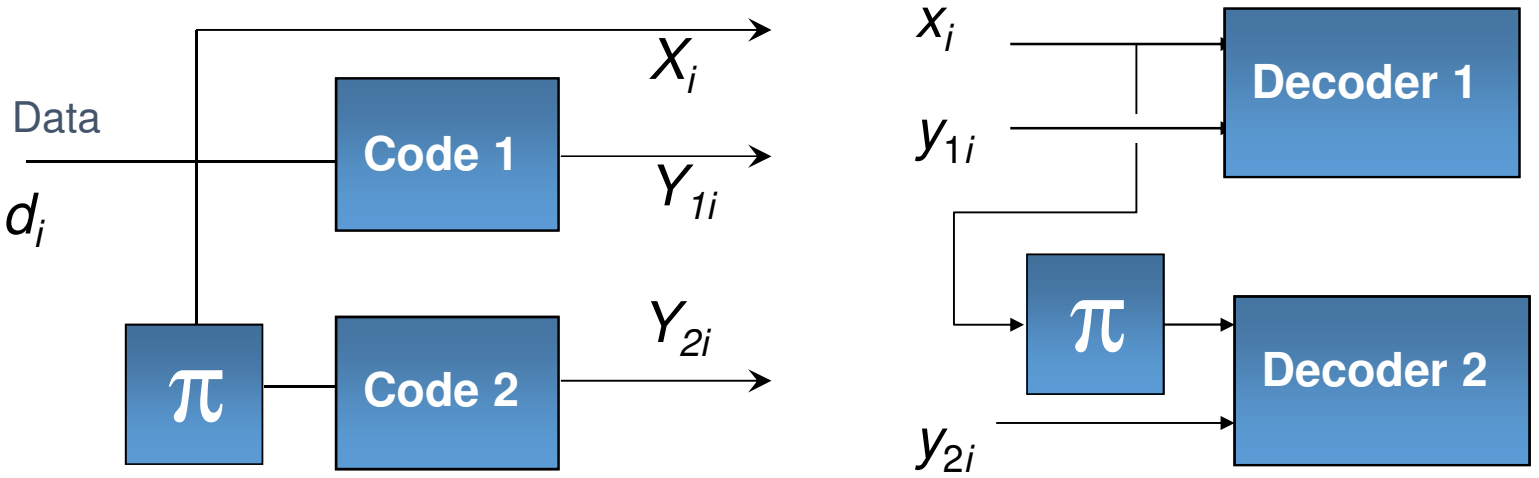
Turbo encoder (2)

Puncturing for higher rates
Example : $R = 1/2$



Turbo codes : Permutation

The basic idea : compensate for the vulnerability of a decoder to errors occuring in bursts



$x_i = (2X_i\text{-}1) + n_i \qquad (X_i=d_i)$

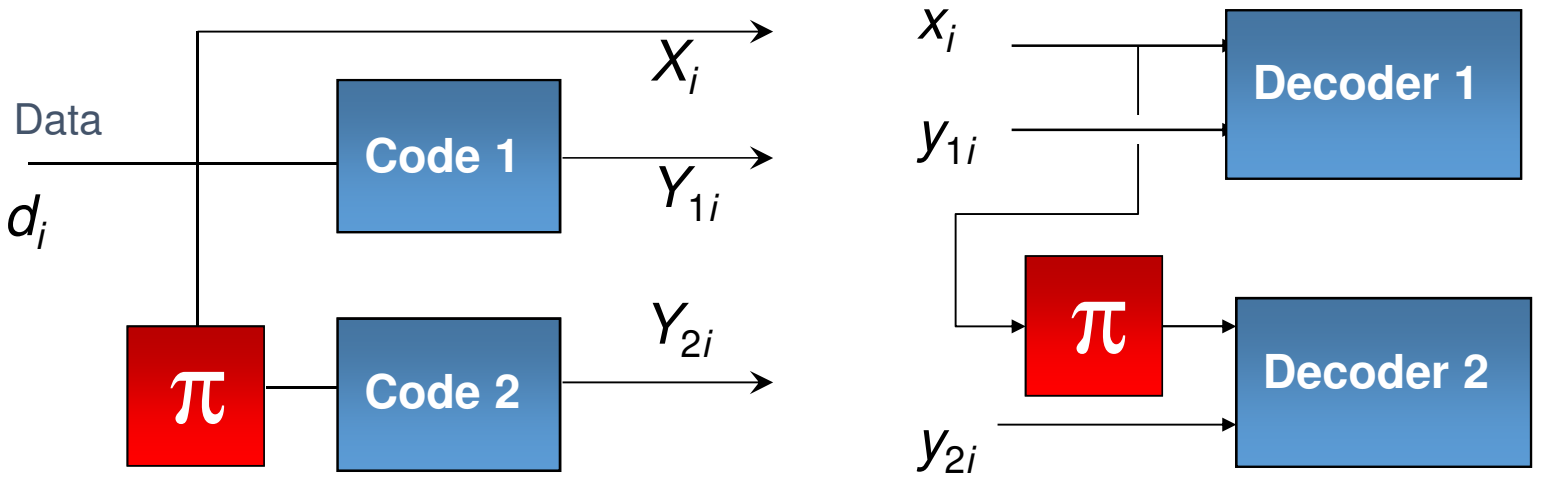
$y_{1i} = (2Y_{1i}\text{-}1) + n_{1i}$

$y_{2i} = (2Y_{2i}\text{-}1) + n_{2i}$

CentraleSupélec

Turbo codes : Permutation

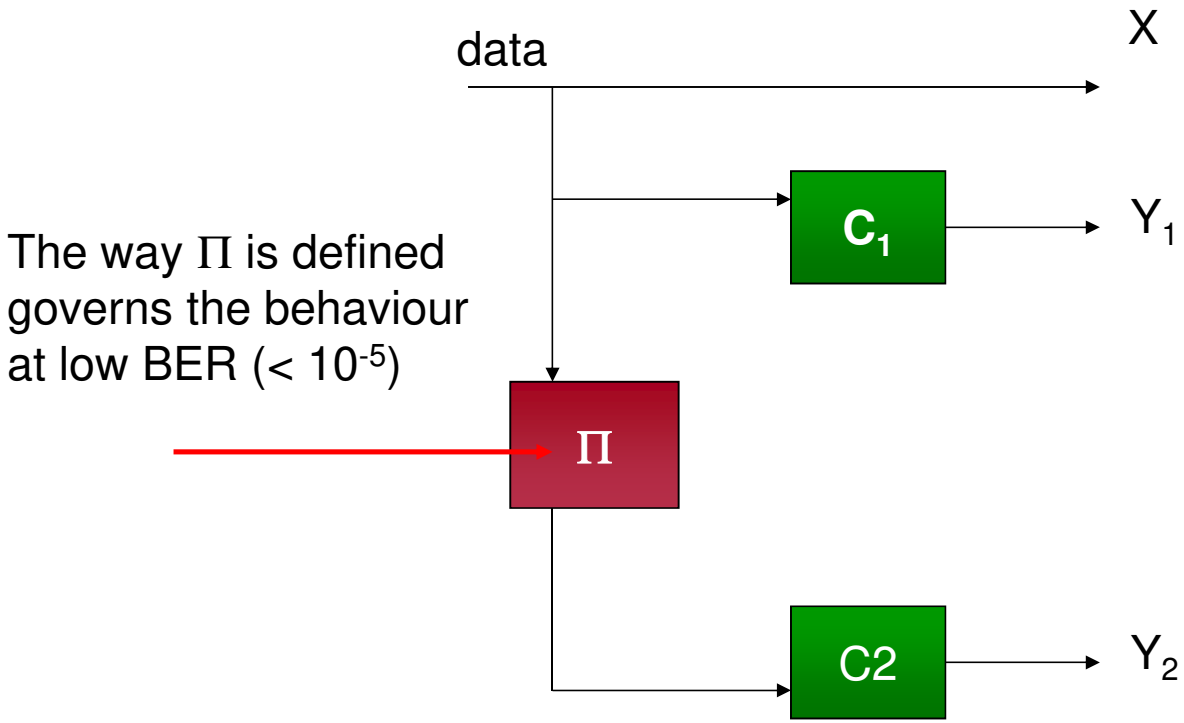
The fundamental role of π :
minimize the probability that both decoders fail (a very tricky problem combining algebra, geometry, ...)
=> Controls the Minimum Hamming Distance (MHD)



The way Π is defined governs the behaviour at low BER (< 10^{-5})

CentraleSupélec

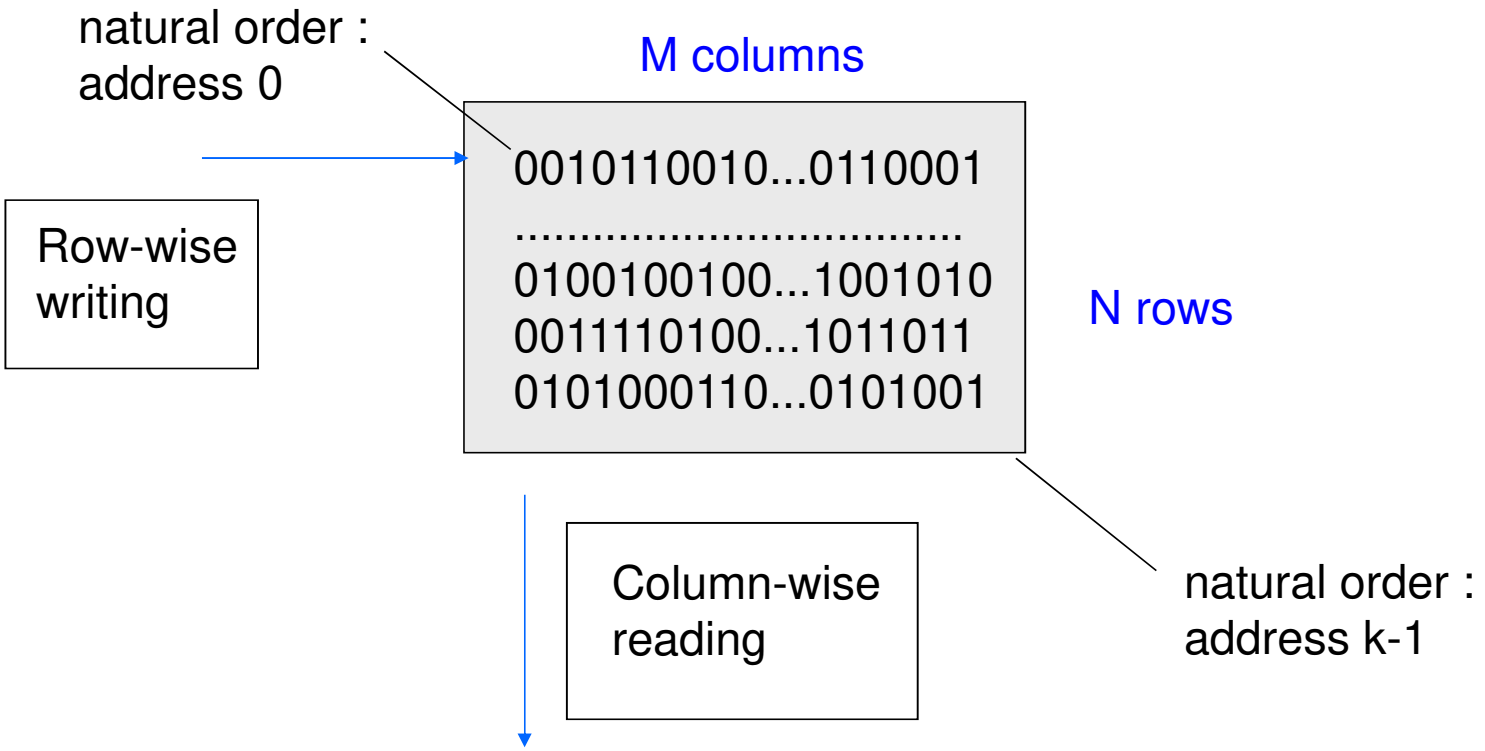
Permutation (permutation ≡ interleaving)



The fundamental role of the permutation : if the direct sequence is RTZ, minimize the probability that the permuted sequence is also RTZ and vice-versa.

CentraleSupélec

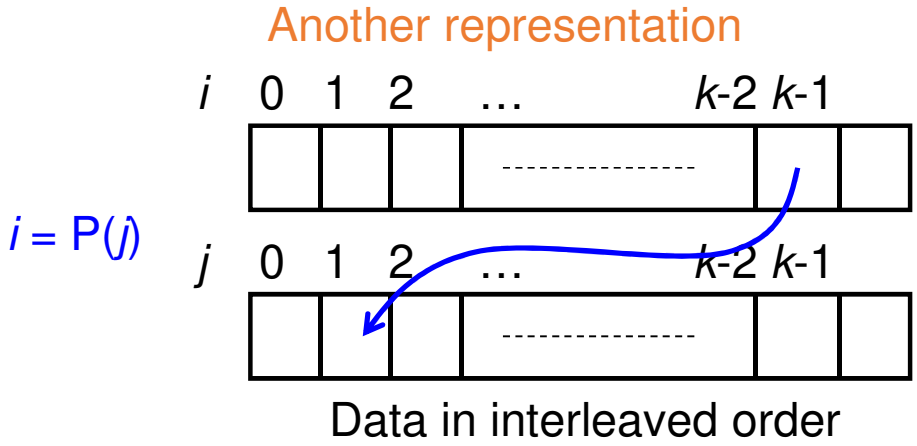
Regular permutation (1)



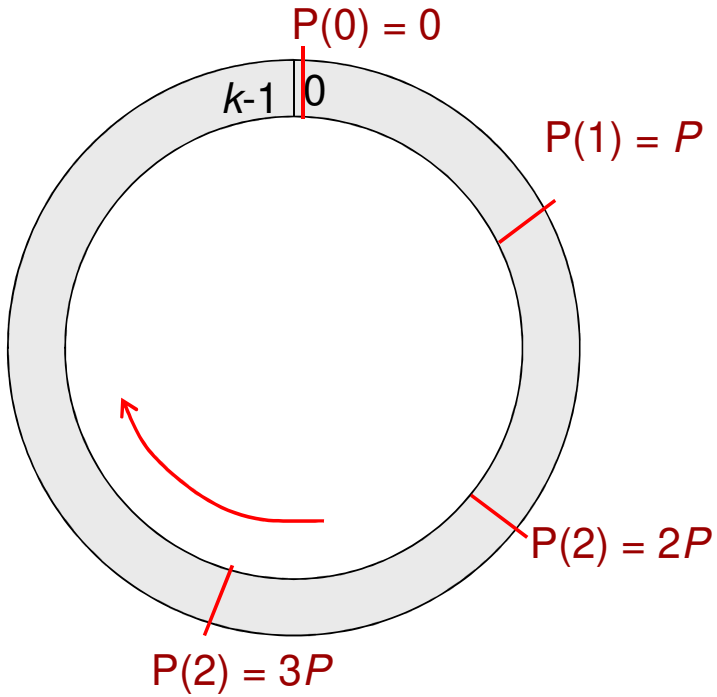
Condition : k = M.N

Regular permutation (2)

Another representation



Data in natural order



$$\text{for } j = 0 \cdots k-1$$

$$\boxed{i = \Pi(j) = Pj \mod k}$$

P and *k* are relatively prime integers

Permutation

So, let us introduce disorder, but not in any manner !!

A good permutation must ensure :
(1) maximum scattering of data
(2) maximum disorder in the permuted sequence

(these two conditions are in conflict)

Permutation: Inserting controlled disorder

Everyone has his own tricks

For instance, the turbo code permutation
DVB-RCS/RCT (ETSI EN 301 790 and 301 958), IEEE 802.16

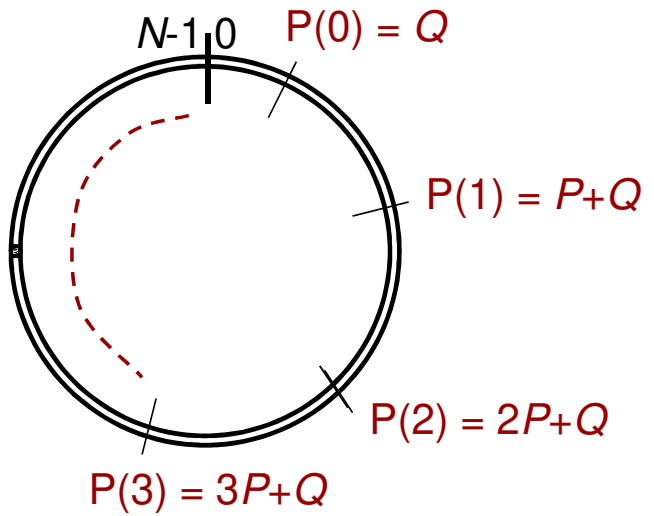
- **Built from a regular permutation**

$$\text{for } j = 0 \cdots N-1$$
$$i = \Pi(j) = Pj + Q \mod N$$

P and *N* are relatively prime integers

- **Introduction of controlled disorder (here with max degree 4)**

$$j \bmod 4 = 0 \Rightarrow Q = P_1$$
$$j \bmod 4 = 1 \Rightarrow Q = P_2$$
$$j \bmod 4 = 2 \Rightarrow Q = P_3$$
$$j \bmod 4 = 3 \Rightarrow Q = P_4$$

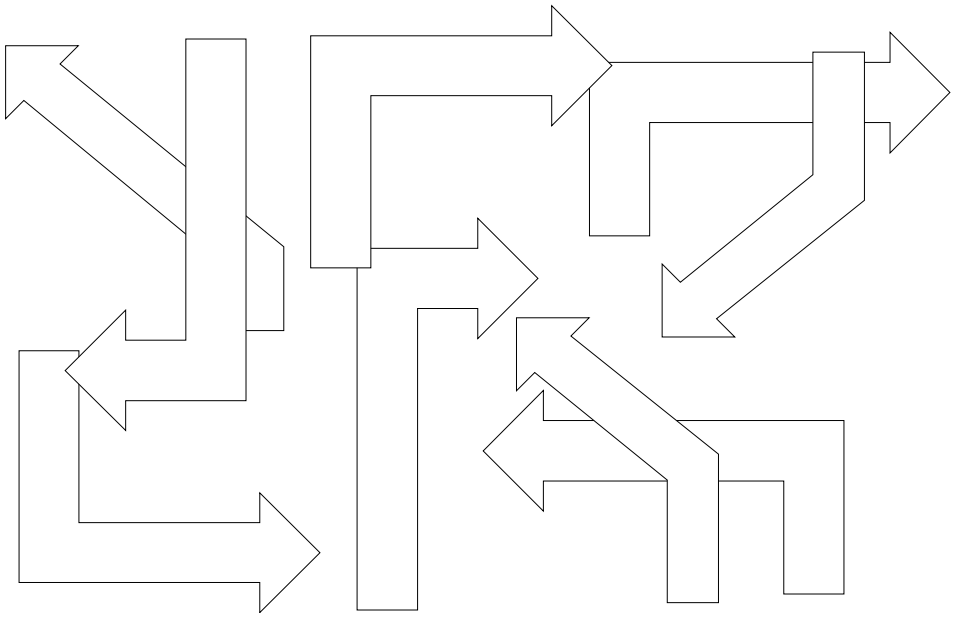


$P_1 \ldots P_4$ are small integers depending on the block size

CentraleSupélec

Permutation

In conclusion :



No ideal permutation for the time being (does it exist ?)

CentraleSupélec

Turbo principle

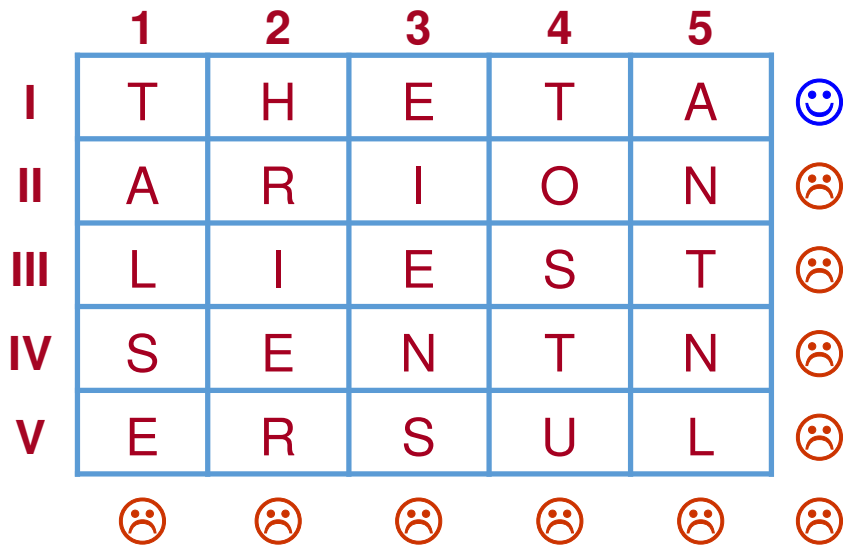
Horizontal

I. Grecque
II. Aéronef
III. Réfutent
IV. Chemin
V. Anneaux

Vertical

1. Ballet
2. Lavabo
3. Attaches
4. Taille
5. Piliers

	1	2	3	4	5
I	T	H	E	T	A
II	A	R	I	O	N
III	L	I	E	S	T
IV	S	E	N	T	N
V	E	R	S	U	L



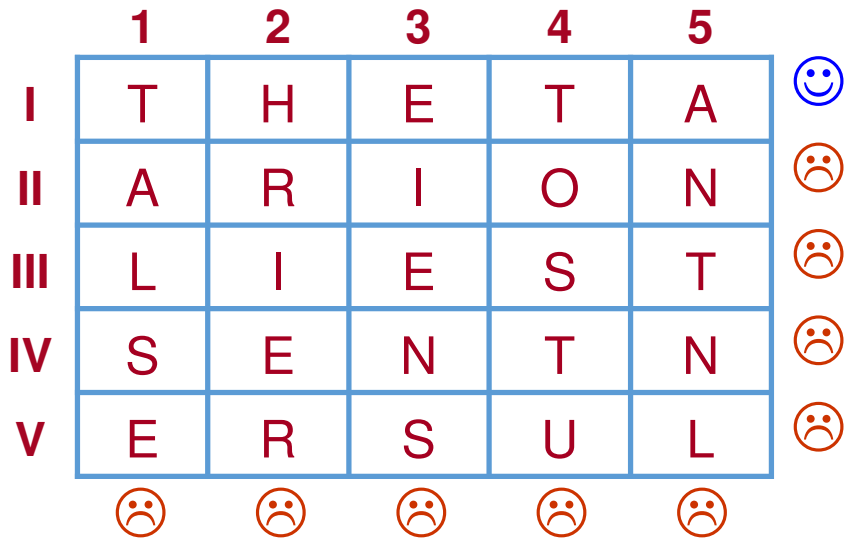
Turbo principle

Horizontal

I. Grecque
II. Aéronef
III. Réfutent
IV. Chemin
V. Anneaux

Vertical

1. Ballet
2. Lavabo
3. Attaches
4. Taille
5. Piliers



Horizontal decoding

Turbo principle

Horizontal

I. Grecque
II. Aéronef
III. Réfutent
IV. Chemin
V. Anneaux

Vertical

1. Ballet
2. Lavabo
3. Attaches
4. Taille
5. Piliers

Vertical decoding

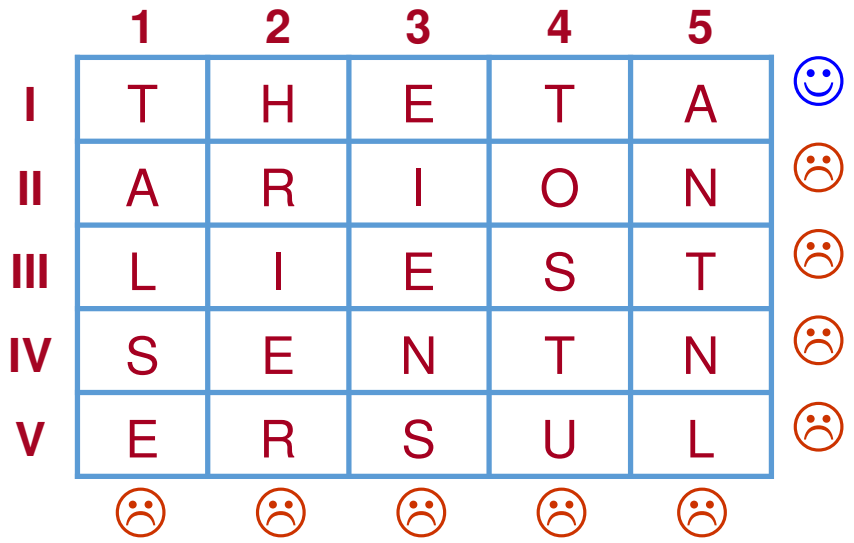
Turbo principle

Horizontal

I. Grecque
II. Aéronef
III. Réfutent
IV. Chemin
V. Anneaux

Vertical

1. Ballet
2. Lavabo
3. Attaches
4. Taille
5. Piliers



Horizontal decoding

Turbo principle

Horizontal

I. Grecque
II. Aéronef
III. Réfutent
IV. Chemin
V. Anneaux

Vertical

1. Ballet
2. Lavabo
3. Attaches
4. Taille
5. Piliers

Vertical decoding

Turbo principle

Horizontal

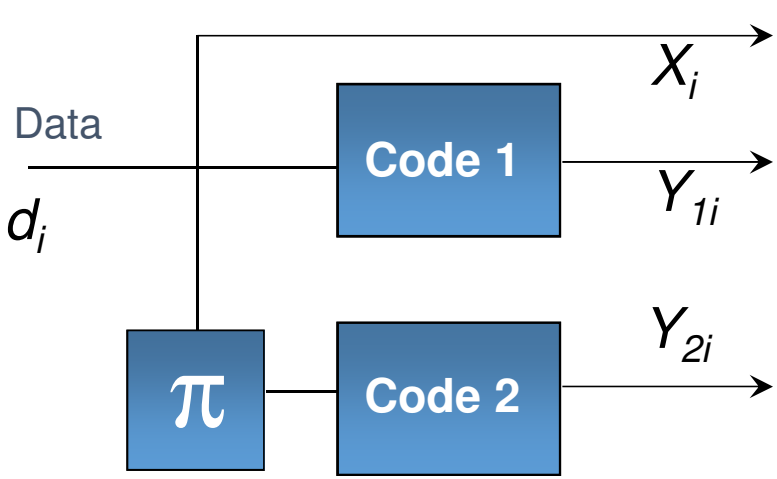
I. Grecque
II. Aéronef
III. Réfutent
IV. Chemin
V. Anneaux

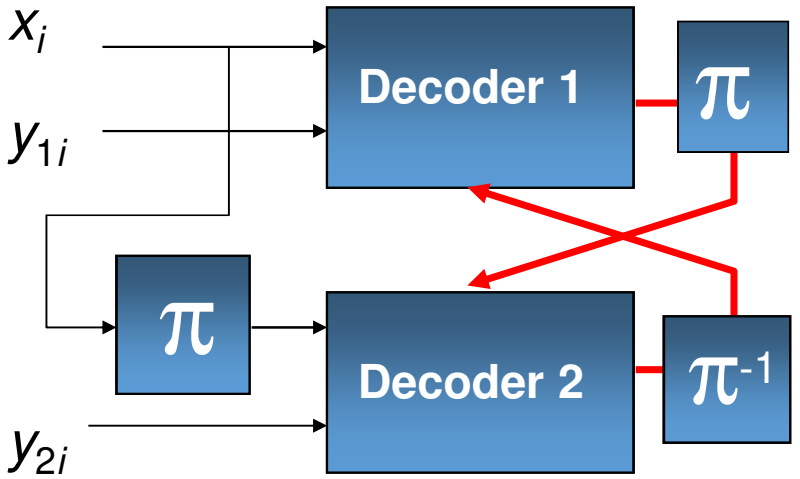
Vertical

1. Ballet
2. Lavabo
3. Attaches
4. Taille
5. Piliers

Turbo decoding

How to make both decoders work jointly,
so that decoder 1 could benefit from Y_2 and
decoder 2 could benefit from Y_1 ?





Message passing 'turbo'

Turbo decoding: SISO algorithms

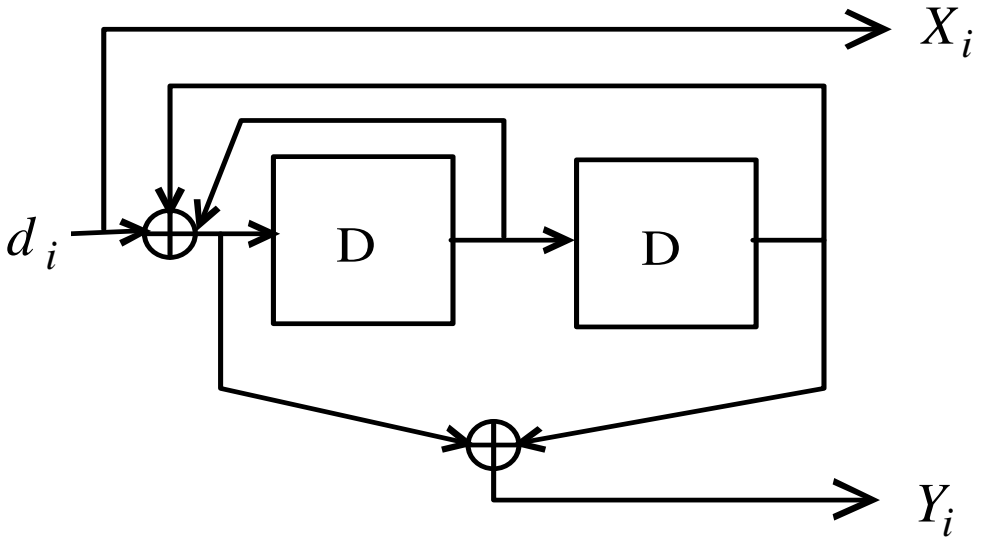
Soft-Input Soft-Output (SISO) decoding algorithms are necessary to turbo decoding.

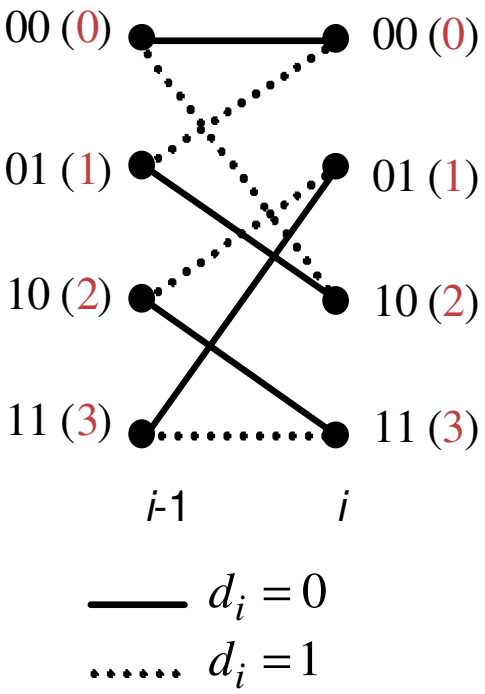
Two families of decoding algorithms:

- The Viterbi based SISO algorithms [2][3]
- The MAP algorithm and its approximations [4]

Turbo decoding: SISO algorithms

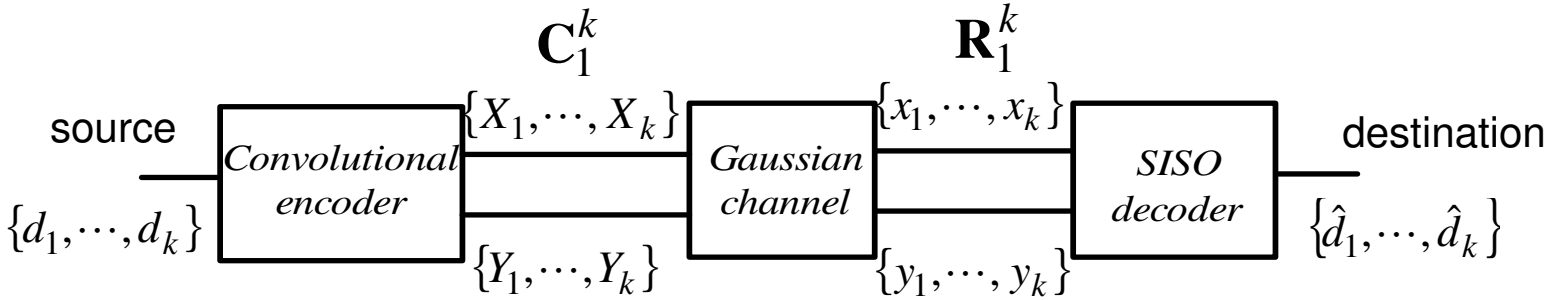
Illustration with the following code:





Turbo decoding: SISO algorithms

Considered transmission chain



The transmitted symbols $X_i, i=1\cdots k$ and $Y_i, i=1\cdots k$ are taken from set {-1,+1}

$\mathbf{C}_1^k$ is the encoded sequence

$\mathbf{C}_1^k = \{C_1,\cdots,C_k\}$ with $C_i = (X_i, Y_i)$

$\mathbf{R}_1^k$ is the noisy received sequence

$\mathbf{R}_1^k = \{R_1,\cdots,R_k\}$ with $R_i = (x_i, y_i)$, $x_i = X_i + n_i^x$, $y_i = Y_i + n_i^y$

Turbo decoding: SISO algorithms

The MAP (*Maximum A Posteriori*) algorithm

- Also known as BCJR (Bahl, Cocke, Jelinek, Raviv), APP (*A Posteriori* Probability), or Backward-Forward algorithm
- Aim of the algorithm: computation of the Logarithm of Likelihood Ratio (LLR) relative to data d_i

$$\Lambda(d_i) = \ln \frac{\Pr\{d_i = 1 \mid \mathbf{R}_1^k\}}{\Pr\{d_i = 0 \mid \mathbf{R}_1^k\}}$$

Sign ⇒ hard decision
Magnitude ⇒ reliability

$\mathbf{R}_1^k$ is the noisy received sequence

$\mathbf{R}_1^k = \{R_1, \cdots, R_k\}$ with $R_i = (x_i, y_i)$

CentraleSupélec

Turbo decoding: MAP algorithm

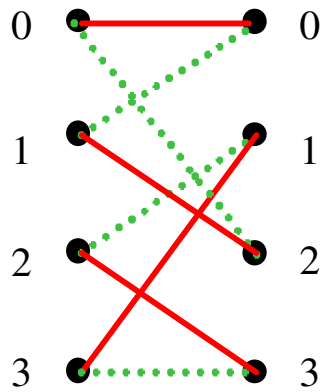
- Each *A Posteriori Probability* (APP) $\Pr\{d_i = j \,|\, \mathbf{R}_1^k\}$ can be written using joint probabilities $\lambda_i^j(\mathbf{m})$:

$$\lambda_i^j(\mathbf{m}) = \Pr\{d_i = j, \mathbf{S}_i = \mathbf{m}, \mathbf{R}_1^k\} \quad j = 0,1$$

$\mathbf{S}_i$: encoder state at time or step *i*

$$\Pr\{d_i = j \,|\, \mathbf{R}_1^k\} = \sum_{\mathbf{m}} \Pr\{d_i = j, \mathbf{S}_i = \mathbf{m} \,|\, \mathbf{R}_1^k\} = \frac{\sum_{\mathbf{m}} \lambda_i^j(\mathbf{m})}{\Pr\{\mathbf{R}_1^k\}}$$

$$\Rightarrow \quad \boxed{\Lambda(d_i) = \ln \frac{\sum_{\mathbf{m}} \lambda_i^1(\mathbf{m})}{\sum_{\mathbf{m}} \lambda_i^0(\mathbf{m})}}$$



- Example:

$$\Lambda(d_i) = \ln \frac{\lambda_i^1(\mathbf{0}) + \lambda_i^1(1) + \lambda_i^1(\mathbf{2}) + \lambda_i^1(\mathbf{3})}{\lambda_i^0(\mathbf{0}) + \lambda_i^0(\mathbf{1}) + \lambda_i^0(\mathbf{2}) + \lambda_i^0(\mathbf{3})}$$

Turbo decoding: MAP algorithm

- The principle of MAP algorithm: processing separately data available between steps 1 to *i* and between steps *i* +1 to *k*.

  => Introduction of *forward state probabilities* $\alpha_i(\mathbf{m}), i = 1 \cdots k$
  and *backward state probabilities* $\beta_i(\mathbf{m}), i = 1 \cdots k$

$$\alpha_i(\mathbf{m}) = \Pr\{\mathbf{S}_i = \mathbf{m}, \mathbf{R}_1^i\}$$

$$\beta_i(\mathbf{m}) = \Pr\{\mathbf{R}_{i+1}^k \mid \mathbf{S}_i = \mathbf{m}\}$$

Turbo decoding: MAP algorithm

- One can show that:

$$\lambda_i^j(\mathbf{m}) = \sum_{\mathbf{m}'} \gamma_j(R_i, \mathbf{m}', \mathbf{m}) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m}), \; j = 0,1$$

where $\gamma_j(R_i, \mathbf{m}', \mathbf{m})$ represents the *branch probability* between states **m'** and **m** when the received symbol is R_i at step *i*.

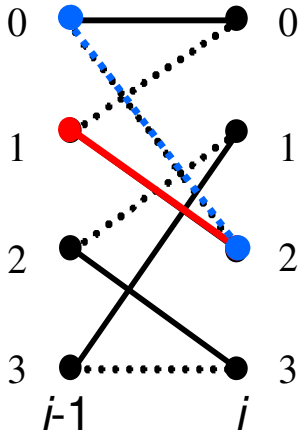
=>

$$\Lambda(d_i) = \ln \frac{\sum\sum_{\mathbf{m}\,\mathbf{m}'} \gamma_1(R_i, \mathbf{m}', \mathbf{m}) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}{\sum\sum_{\mathbf{m}\,\mathbf{m}'} \gamma_0(R_i, \mathbf{m}', \mathbf{m}) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}$$

Turbo decoding: MAP algorithm

Example:

* $\lambda_i^0(\mathbf{2}) = \gamma_0(R_i,\mathbf{1},\mathbf{2})\alpha_{i-1}(\mathbf{1})\beta_i(\mathbf{2})$
* $\lambda_i^1(\mathbf{2}) = \gamma_1(R_i,\mathbf{0},\mathbf{2})\alpha_{i-1}(\mathbf{0})\beta_i(\mathbf{2})$



$$\Lambda(d_i) = \ln \frac{\gamma_1(R_i,\mathbf{1},\mathbf{0})\alpha_{i-1}(\mathbf{1})\beta_i(\mathbf{0}) + \gamma_1(R_i,\mathbf{2},\mathbf{1})\alpha_{i-1}(\mathbf{2})\beta_i(\mathbf{1})}{\gamma_0(R_i,\mathbf{0},\mathbf{0})\alpha_{i-1}(\mathbf{0})\beta_i(\mathbf{0}) + \gamma_0(R_i,\mathbf{3},\mathbf{1})\alpha_{i-1}(\mathbf{3})\beta_i(\mathbf{1})} \cdots$$

$$\cdots \frac{+\gamma_1(R_i,\mathbf{0},\mathbf{2})\alpha_{i-1}(\mathbf{0})\beta_i(\mathbf{2}) + \gamma_1(R_i,\mathbf{3},\mathbf{3})\alpha_{i-1}(\mathbf{3})\beta_i(\mathbf{3})}{+\gamma_0(R_i,\mathbf{1},\mathbf{2})\alpha_{i-1}(\mathbf{1})\beta_i(\mathbf{2}) + \gamma_0(R_i,\mathbf{2},\mathbf{3})\alpha_{i-1}(\mathbf{2})\beta_i(\mathbf{3})}$$

Turbo decoding: MAP algorithm

- **Forward recursion**: computation of $\alpha_i(\mathbf{m})$

$\alpha_i(\mathbf{m})$ can be recursively computed from $\alpha_i(\mathbf{m}) = \sum_{\mathbf{m}'} \sum_j \alpha_{i-1}(\mathbf{m}')\gamma_j(R_i, \mathbf{m}', \mathbf{m})$

$\alpha_i(\mathbf{m})$ are all computed during the forward recursion in the trellis from initial values $\alpha_0(\mathbf{m})$

If $\mathbf{m}_0$ is the initial state of the encoder: $\alpha_0(\mathbf{m}_0) = 1$

$$\alpha_0(\mathbf{m}) = 0 \quad \forall \mathbf{m} \neq \mathbf{m}_0$$

NB: to avoid numerical precision problems, $\alpha_i(\mathbf{m})$ can be normalized at each step of processing ($\Lambda(d_i)$ is a ratio)

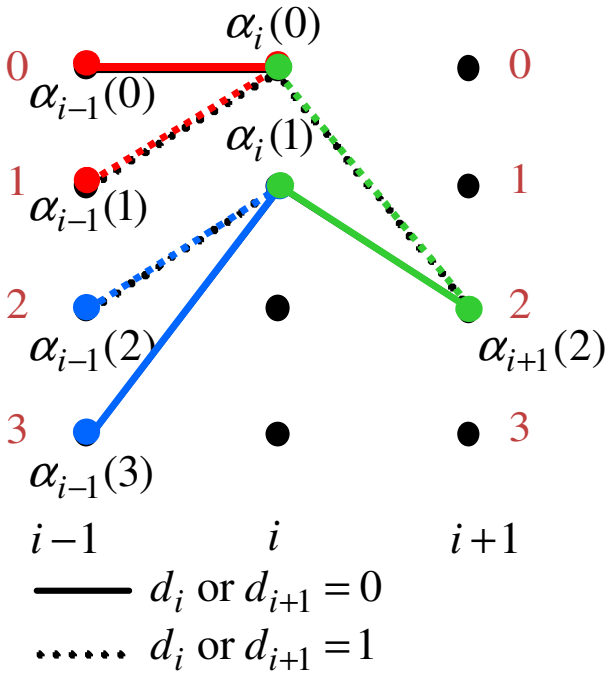
$$\alpha'_i(\mathbf{m}) = \frac{\alpha_i(\mathbf{m})}{\sum_{\mathbf{m}'} \alpha_i(\mathbf{m}')}$$

CentraleSupélec

Turbo decoding: MAP algorithm

Example:

(without normalization)



* $\alpha_i(\mathbf{0}) = \alpha_{i-1}(\mathbf{0})\gamma_0(R_i,\mathbf{0},\mathbf{0}) + \alpha_{i-1}(\mathbf{1})\gamma_1(R_i,\mathbf{1},\mathbf{0})$
* $\alpha_i(\mathbf{1}) = \alpha_{i-1}(\mathbf{3})\gamma_0(R_i,\mathbf{3},\mathbf{1}) + \alpha_{i-1}(\mathbf{2})\gamma_1(R_i,\mathbf{2},\mathbf{1})$
* $\alpha_{i+1}(\mathbf{2}) = \alpha_i(\mathbf{1})\gamma_0(R_{i+1},\mathbf{1},\mathbf{2}) + \alpha_i(\mathbf{0})\gamma_1(R_{i+1},\mathbf{0},\mathbf{2})$

Turbo decoding: MAP algorithm

- **Backward recursion**: computation of $\beta_i(\mathbf{m}')$

$\beta_i(\mathbf{m}')$ can be recursively computed from $\beta_i(\mathbf{m}') = \sum_{\mathbf{m}'} \sum_j \beta_{i+1}(\mathbf{m}) \gamma_j(R_{i+1}, \mathbf{m}', \mathbf{m})$

$\beta_i(\mathbf{m}')$ are all computed during the backward recursion into the trellis from initial values $\beta_k(\mathbf{m})$

If the final state of the encoder is known ($\mathbf{m}_k$):

$$\beta_k(\mathbf{m}_k) = 1, \quad \beta_k(\mathbf{m}) = 0 \quad \forall \mathbf{m} \neq \mathbf{m}_k$$

If the final state of the encoder is unknown:

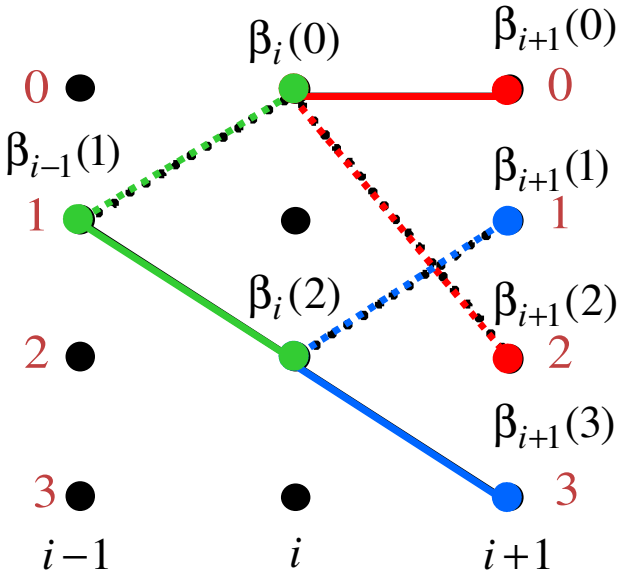
$$\beta_k(\mathbf{m}) = \frac{1}{2^\nu} \quad \forall \mathbf{m} \qquad \text{(n = code memory)}$$

Normalization: $$\beta'_i(\mathbf{m}) = \frac{\beta_i(\mathbf{m})}{\sum_{\mathbf{m}'} \beta_i(\mathbf{m}')}$$

Turbo decoding: MAP algorithm

Example:

(without normalization)



d_i or $d_{i+1} = 0$

d_i or $d_{i+1} = 1$

* $\beta_i(\mathbf{0}) = \beta_{i+1}(\mathbf{0})\gamma_0(R_{i+1},\mathbf{0},\mathbf{0}) + \beta_{i+1}(\mathbf{2})\gamma_1(R_{i+1},\mathbf{0},\mathbf{2})$
* $\beta_i(\mathbf{2}) = \beta_{i+1}(\mathbf{3})\gamma_0(R_{i+1},\mathbf{2},\mathbf{3}) + \beta_{i+1}(\mathbf{1})\gamma_1(R_{i+1},\mathbf{2},\mathbf{1})$
* $\beta_{i-1}(\mathbf{1}) = \beta_i(\mathbf{2})\gamma_0(R_i,\mathbf{1},\mathbf{2}) + \beta_i(\mathbf{0})\gamma_1(R_i,\mathbf{1},\mathbf{0})$

CentraleSupélec

Turbo decoding: MAP algorithm

- **Branch probabilities**: computation of $\gamma_j(R_i, \mathbf{m}', \mathbf{m})$

If there is no transition between **m'** and **m** in the trellis or if the transition is not labeled with $d_i = j$.

$$\Rightarrow \gamma_j(R_i, \mathbf{m}', \mathbf{m}) = 0$$

else, for a transmission over a Gaussian channel

$$\gamma_j(R_i, \mathbf{m}', \mathbf{m}) = \Pr\{d_i = \mathrm{j}\} \times \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|R_i - C_i\|^2}{2\sigma^2} \right)$$

Where C_i is the expected symbol along the branch from state **m'** to state **m** and R_i is the received symbol .

CentraleSupélec

Turbo decoding: MAP algorithm

- **Branch probabilities**: computation of $\gamma_j(R_i, \mathbf{m}', \mathbf{m})$

If $C_i = (X_i, Y_i)$, $R_i = (x_i, y_i)$ and $\Pr\{d_i = 0\} = \Pr\{d_i = 1\} = \frac{1}{2}$

$$\gamma_j(R_i, \mathbf{m}', \mathbf{m}) = \frac{1}{4\pi\sigma^2} \exp\left(-\frac{(x_i - X_i)^2}{2\sigma^2}\right) \exp\left(-\frac{(y_i - Y_i)^2}{2\sigma^2}\right)$$

$$\gamma_1(R_i, \mathbf{m}', \mathbf{m}) = A \exp\left(\frac{x_i}{\sigma^2}\right) \exp\left(\pm\frac{y_i}{\sigma^2}\right) \qquad (X_i = +1)$$

$$\gamma_0(R_i, \mathbf{m}', \mathbf{m}) = A \exp\left(-\frac{x_i}{\sigma^2}\right) \exp\left(\mp\frac{y_i}{\sigma^2}\right) \qquad (X_i = -1)$$

Derivation of extrinsic information for iterative decoding

$$\Lambda(d_i) = \ln \frac{\sum\limits_{\mathbf{m}} \sum\limits_{\mathbf{m}'} \gamma_1(R_i, \mathbf{m}', \mathbf{m}) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}{\sum\limits_{\mathbf{m}} \sum\limits_{\mathbf{m}'} \gamma_0(R_i, \mathbf{m}', \mathbf{m}) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}$$

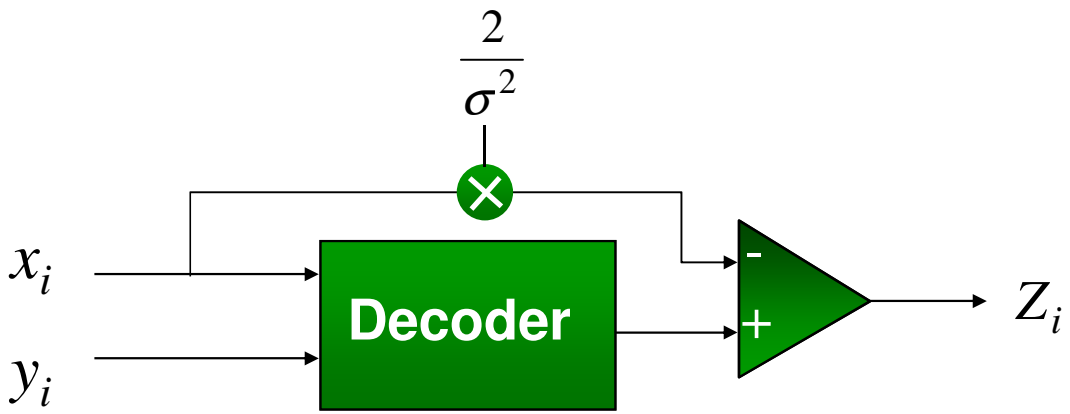
$$\Lambda(d_i) = \ln \frac{\sum\limits_{\mathbf{m}} \sum\limits_{\mathbf{m}'} \exp\left(\frac{x_i}{\sigma^2}\right) \exp\left(\pm \frac{y_i}{\sigma^2}\right) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}{\sum\limits_{\mathbf{m}} \sum\limits_{\mathbf{m}'} \exp\left(-\frac{x_i}{\sigma^2}\right) \exp\left(\mp \frac{y_i}{\sigma^2}\right) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}$$

$$\Lambda(d_i) = \frac{2}{\sigma^2} x_i + \underbrace{\ln \frac{\sum\limits_{\mathbf{m}} \sum\limits_{\mathbf{m}'} \exp\left(\pm \frac{y_i}{\sigma^2}\right) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}{\sum\limits_{\mathbf{m}} \sum\limits_{\mathbf{m}'} \exp\left(\mp \frac{y_i}{\sigma^2}\right) \alpha_{i-1}(\mathbf{m}') \beta_i(\mathbf{m})}}_{z_i}$$

CentraleSupélec

Derivation of extrinsic information for iterative decoding

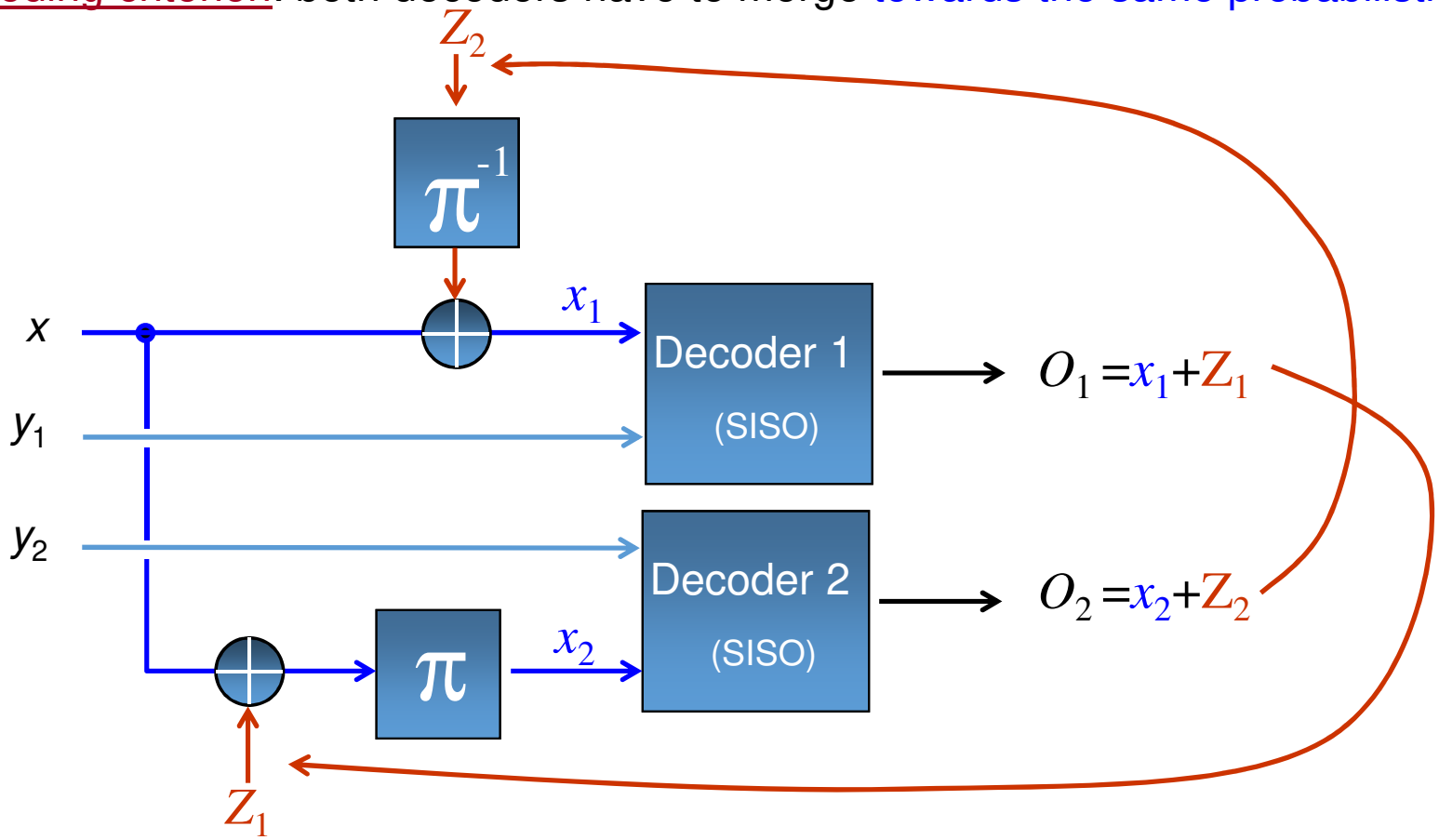
$\Lambda'(d_i)$ can be written as $\Lambda'(d_i) = \frac{2}{\sigma^2} x_i + Z_i$



Z_i is the *extrinsic information* provided by the decoder

TC: Extrinsic Information

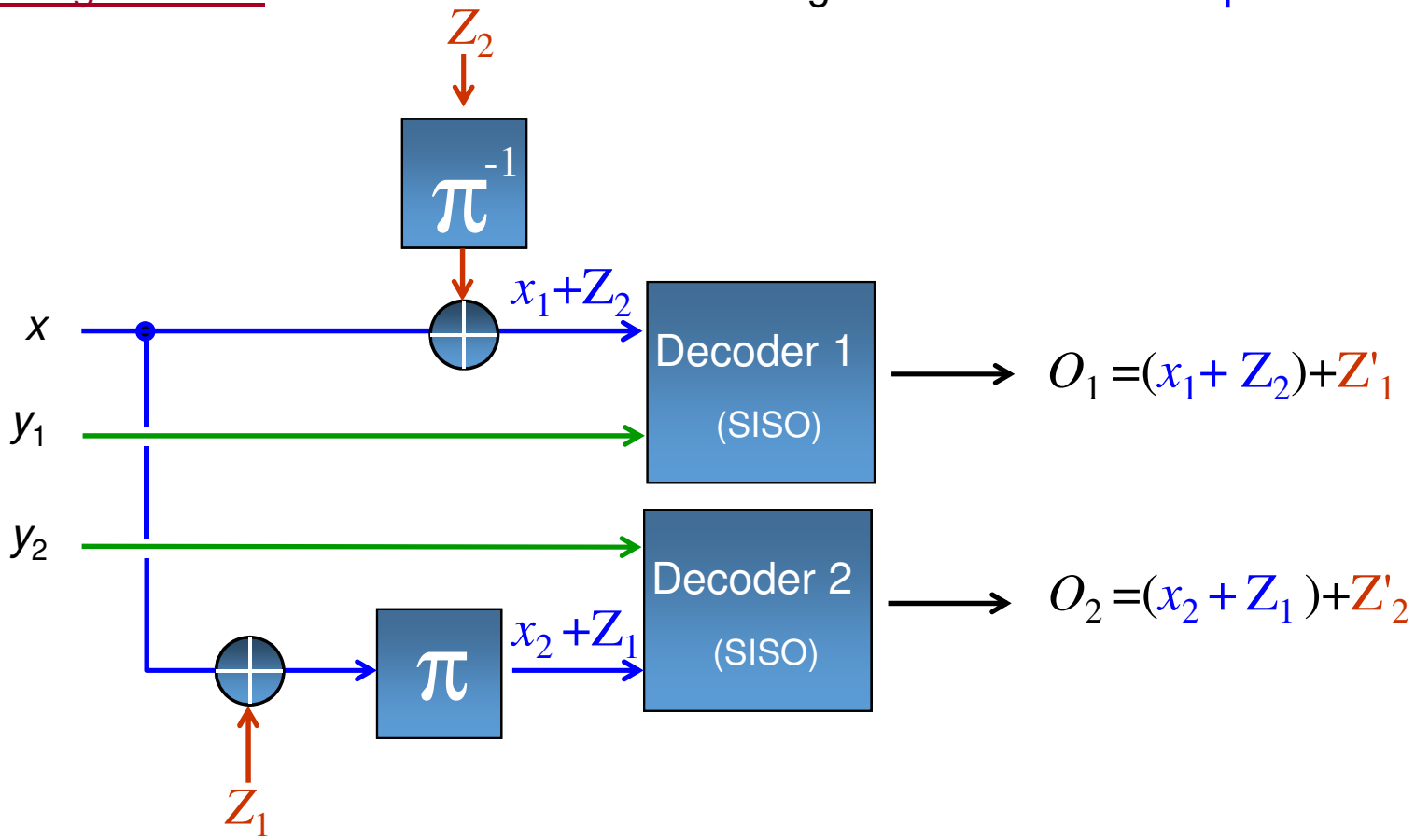
- Decoding criterion: both decoders have to merge towards the same probabilistic decision



CentraleSupélec

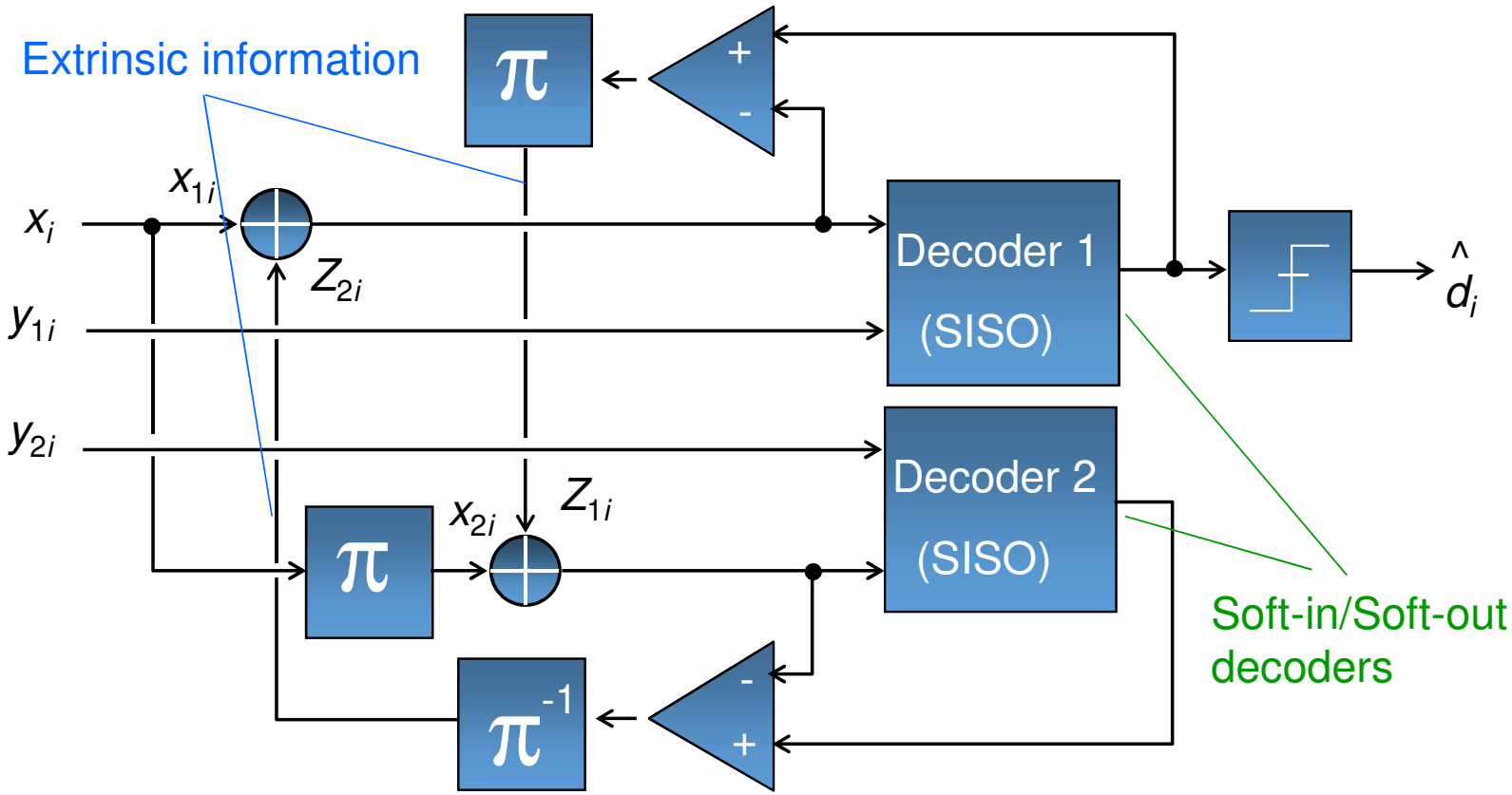
TC: Extrinsic Information

- Decoding criterion: both decoders have to merge towards the same probabilistic decision



CentraleSupélec

The Turbo decoder structure

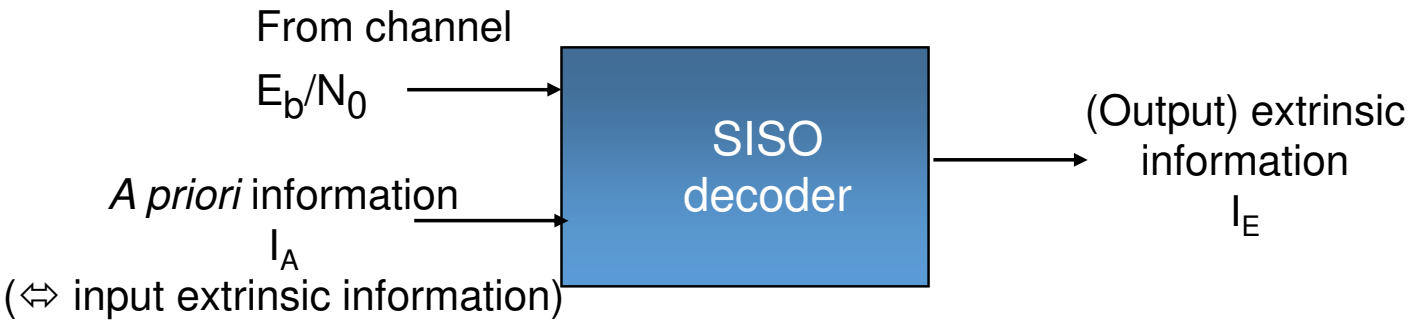


Through extrinsic information, each decoder uses both redundancies.
Fundamental principle: a decoder must not reuse a piece of information provided by itself .

The Turbo principle

Iterations and convergence of a turbo decoder

- Mutual information transfer characteristic
  - Describes the flow of extrinsic information through the SISO decoder

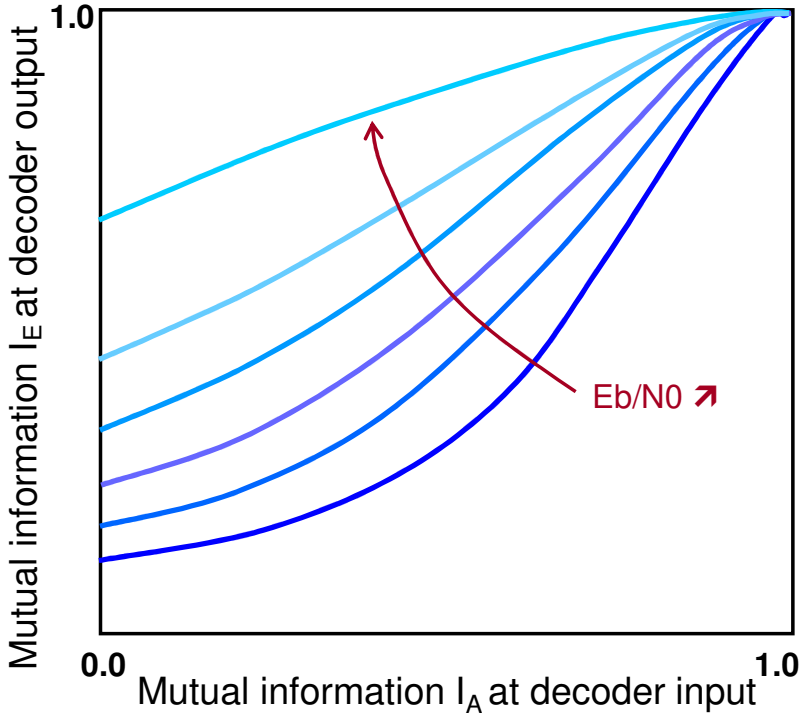


$$I_E = f(I_A, E_b/N_0)$$

The Turbo principle

Iterations and convergence of a turbo decoder

- Mutual information transfer characteristic
  - Describes the flow of extrinsic information through the SISO decoder

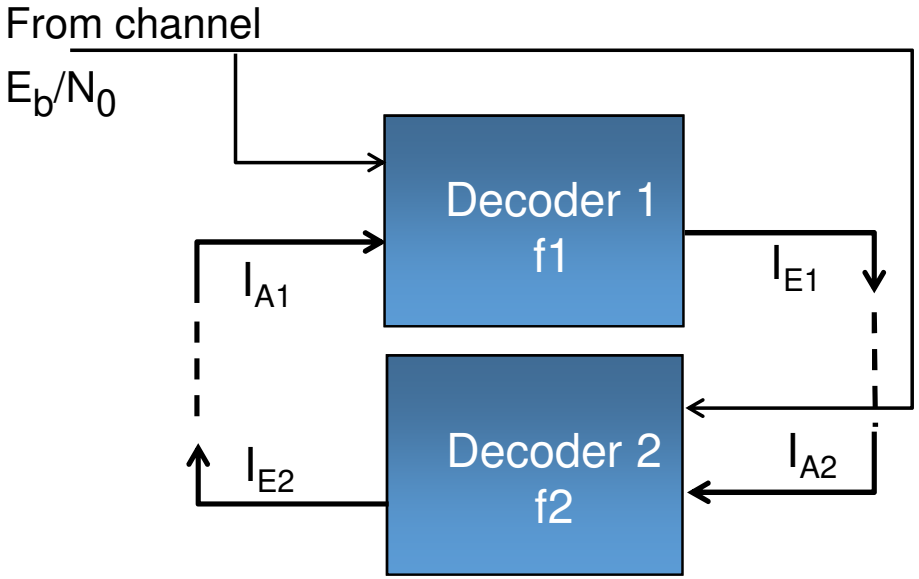


CentraleSupélec

The Turbo principle

Iterations and convergence of a turbo decoder

- Mutual information transfer characteristic
  - Describes the exchange of extrinsic information between the SISO decoders

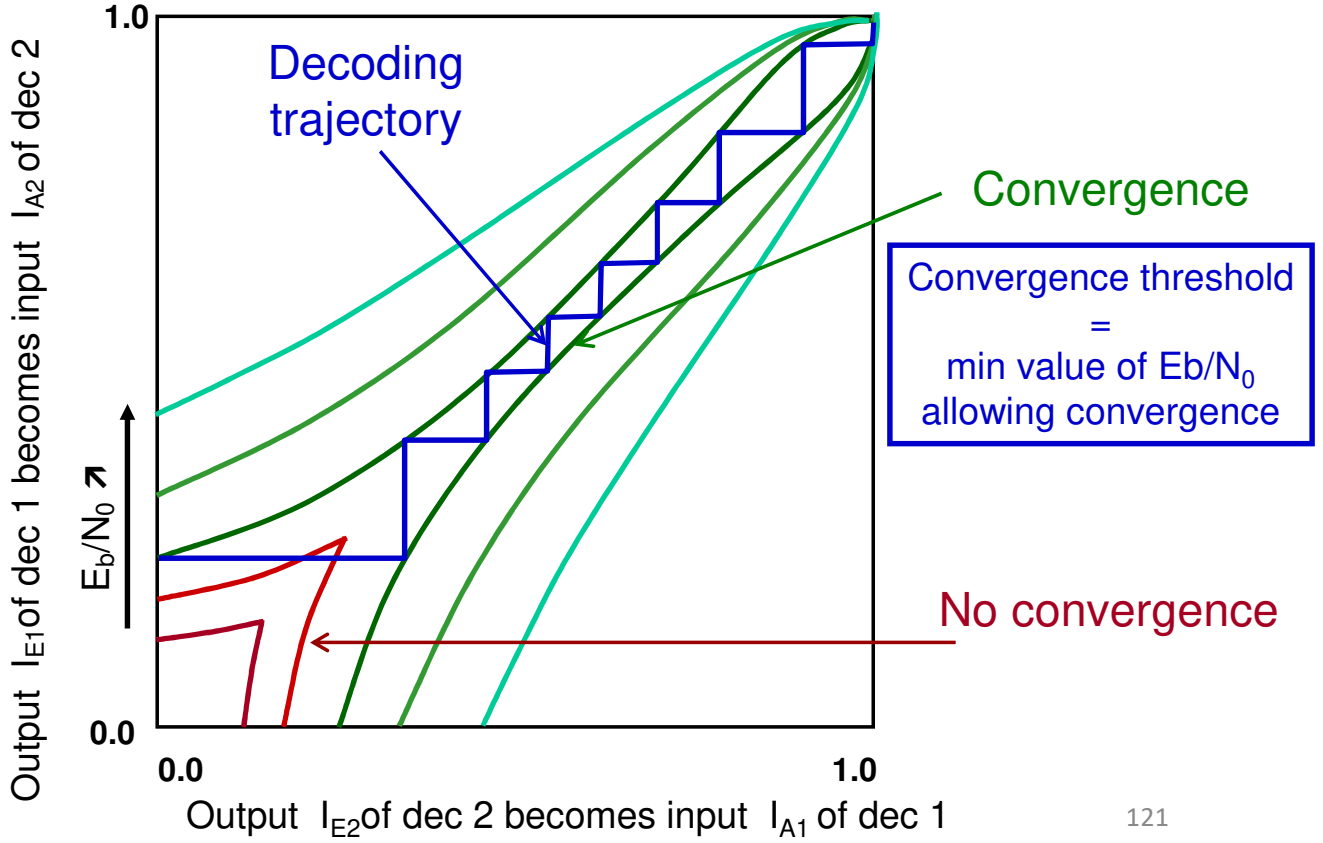


  - Determination of the convergence threshold

The Turbo principle

Iterations and convergence of a turbo decoder

- Mutual information transfer characteristic
  - Describes the exchange of extrinsic information between the SISO decoders



CentraleSupélec

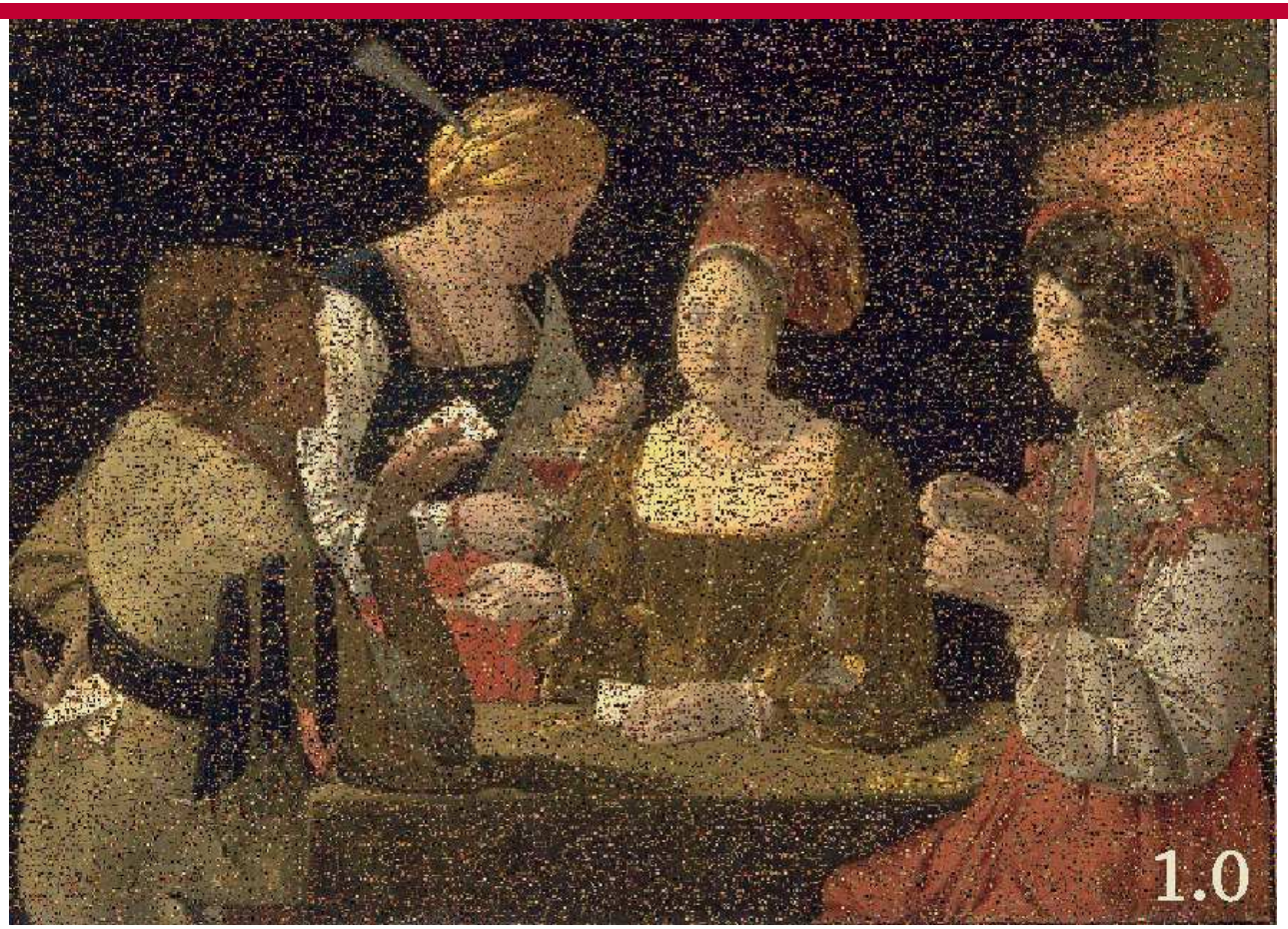
TC: Examples of performance

Received image

[Joseph Boutros, IMT Atlantique Brest, ELEC Dept]

TC: Examples of performance

After 1 decoding iteration



[Joseph Boutros, IMT Atlantique Brest, ELEC Dept]

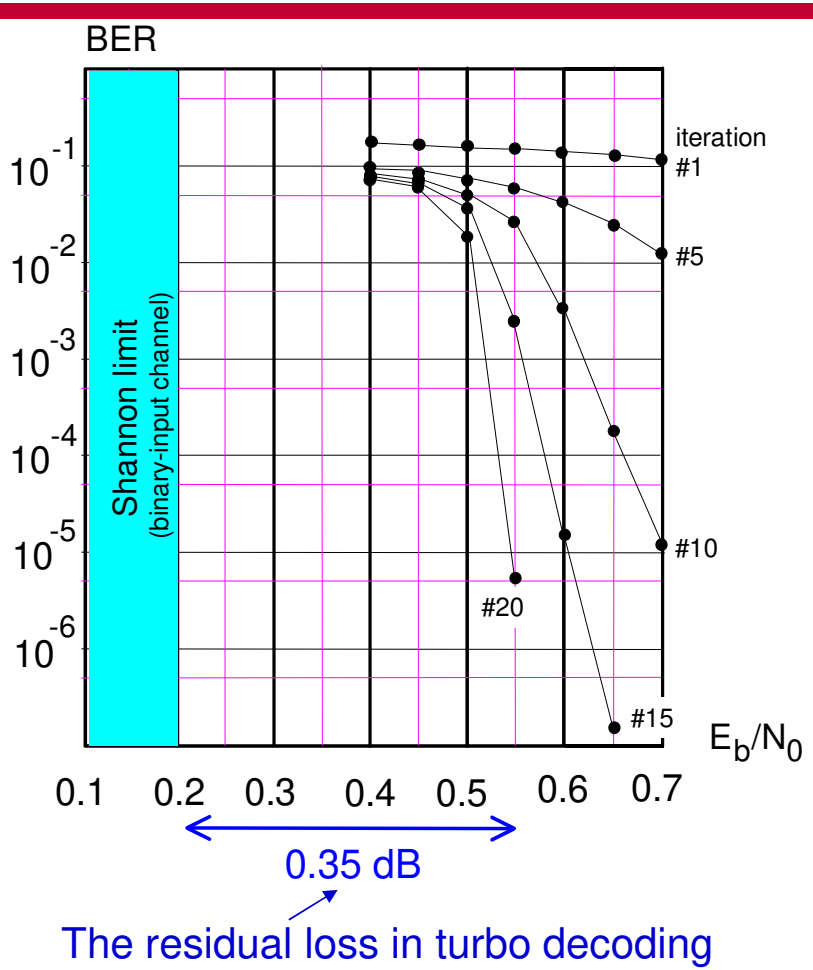
CentraleSupélec

TC: Examples of performance

After 8 decoding iteration

[Joseph Boutros, IMT Atlantique Brest, ELEC Dept]

Convolutional Turbo Codes

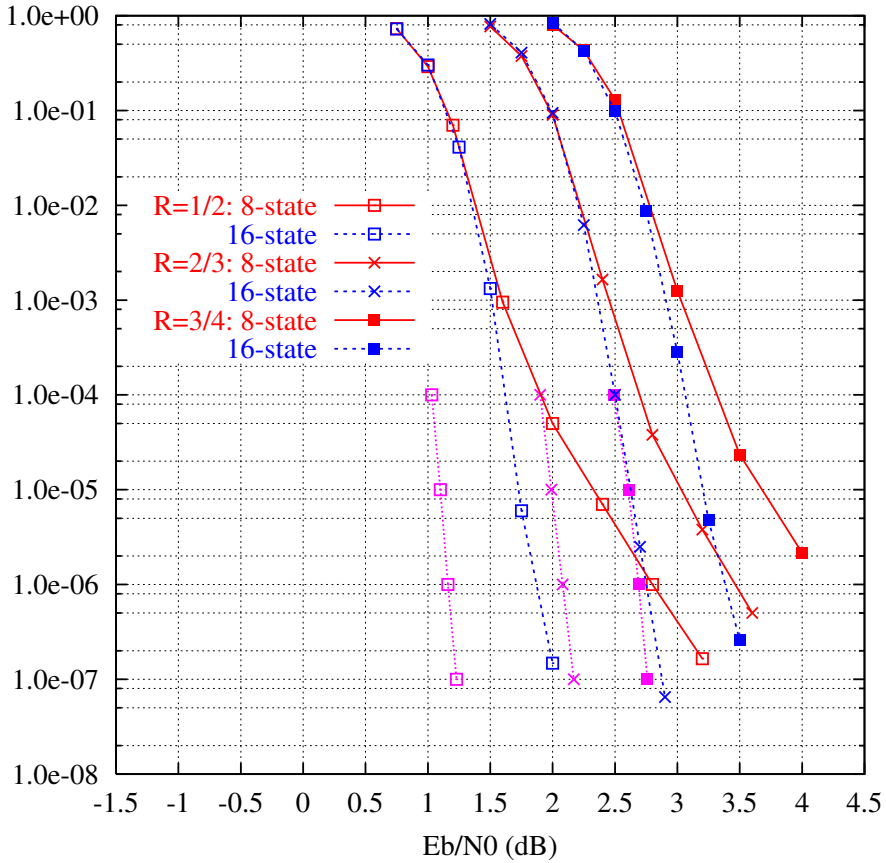


Example of performance

- QPSK modulation
- R=1/2
- AWGN channel
- Double-binary 8-state turbo code
- Large block size
- MAP algorithm
- No quantization

CentraleSupélec

Convolutional Turbo Codes



Example of performance

- QPSK modulation
- AWGN channel
- Double-binary turbo codes
- 1504-bit data blocks
- Max-log-MAP algorithm
- 4-bit quantization
- 8 iterations

CentraleSupélec

Turbo Codes: applications

Turbo Codes: standards (1/2)

Application	Turbo code	termination	polynomials	rates
CCSDS (deep space)	binary, 16-state	tail bits	23,33,25,37	1/6,1/4,1/3,1/2
3GPP (UMTS)	binary, 8-state	tail bits	13,15,17	1/4,1/3,1/2
3GPP2 (CDMA2000)	binary, 8-state	tail bits	13,15,17	1/4,1/3,1/2
3GPP LTE (Long Term Evolution)	binary, 8-state	tail bits	13,15,17	1/4,1/3,1/2
DVB-RCS (Return Channel over Satellite)	double-binary, 8-state	circular (tail-biting)	15,13	1/3 up to 6/7
DVB-RCT (Return Channel over Terrestrial)	double-binary, 8-state	circular (tail-biting)	15,13	1/2, 3/4
DVB-SSP (satellite service to portable)	binary, 8-state	tail bits	15,13	

Turbo Codes: standards (2/2)

Application	Turbo code	termination	polynomials	rates
IEEE 802.16 (WiMAX) + IEEE 802.16e (Mobile WiMAX)	double-binary, 8-state +Turbo product code	circular (tail-biting)	15,13	
Inmarsat (Aero-H)	binary, 16-state	No	23, 35	
Eutelsat (Skyplex)	double-binary, 8-state	circular (tail-biting)	15,13	4/5,6/7
Broadcom (Echostar)				
Qualcomm (MediaFLO)				
Homeplug	Turbo product code			

V- Error Detection and retransmission protocols

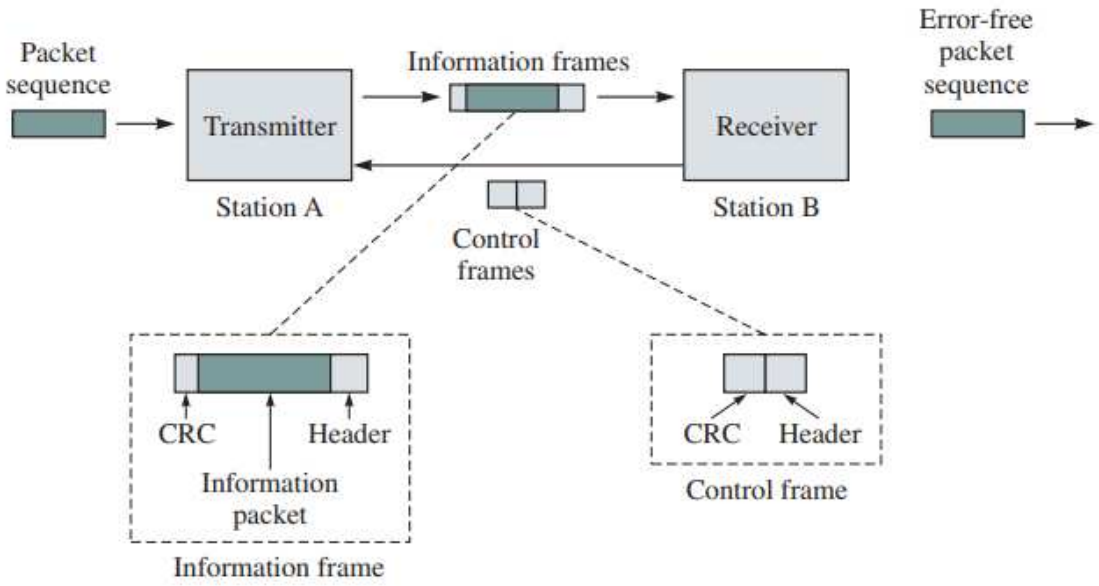
1. *Cyclic Redundancy Check Codes*
2. *HARQ protocols*
3. *IR-HARQ protocols*

CRC (Cyclic Redundancy Check Codes)

- Ils peuvent détecter :
  - toutes les salves d'erreur de longueur *r* ou moins.
  - une partie des salves d'erreurs de longueur *r + 1*
  - une partie des salves d'erreurs de longueur > *r* + 1
  - toutes les combinaisons d'erreurs de poids d_{min}-1
  - toutes les combinaisons d'erreurs de poids impair si *g*(*z*) correspond à un mot-code de poids pair.
- Les CRC possèdent tous *(1+z)* comme polynôme diviseur.

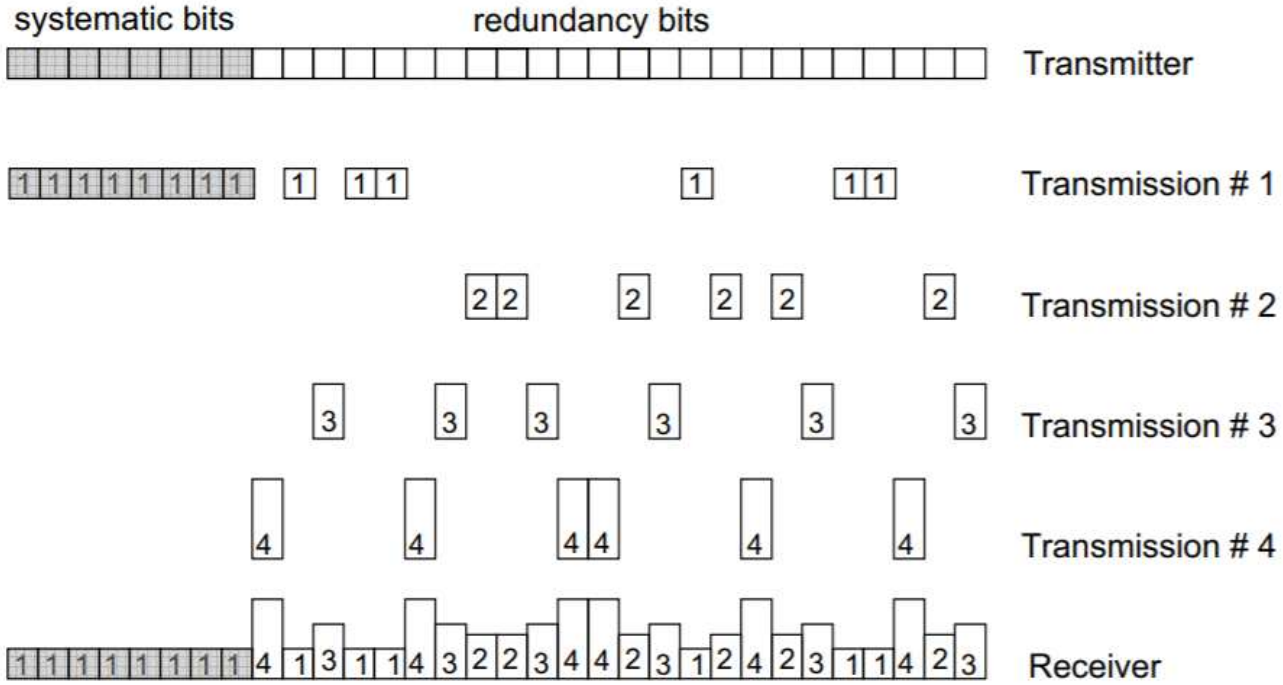
Retransmission protocols (1/2)

- ARQ (Automatic Repeat reQuest) protocols



Retransmission protocols (2/2)

- Hybrid FEC/ARQ protocols
  - Type I HARQ (same channel coding + chase combining)
  - Type II HARQ (Incremental redundancy)



Thank you for you attention!

Haïfa Farès

haifa.fares@centralesupelec.fr

Course materials are highly inspired from «Digital Communications » of Proakis