Introduction to information theory and coding

Louis WEHENKEL

Channel coding (data transmission)

1. Intuitive introduction
2. Discrete channel capacity
3. Differential entropy and continuous channel capacity
4. Error detecting and correcting codes
5. Turbo-codes

Continuous channels

- General ideas, motivations and justifications for continuous channel models
- Memoryless discrete time Gaussian channel
- Discussion of Band-limited White Additive Gaussian Noise channel
- Optimal codes for Band-limited channels
- Geometric interpretations

Motivations

Real-world channels are essentially continuous (in time and in signal space) input/output devices.

$\Rightarrow$ we need continuous information theory to study the limitations and the appropriate way to exploit these devices

Physical limitations of real devices :

- Limited Power/Energy of input signals
- Limited Band-width of devices

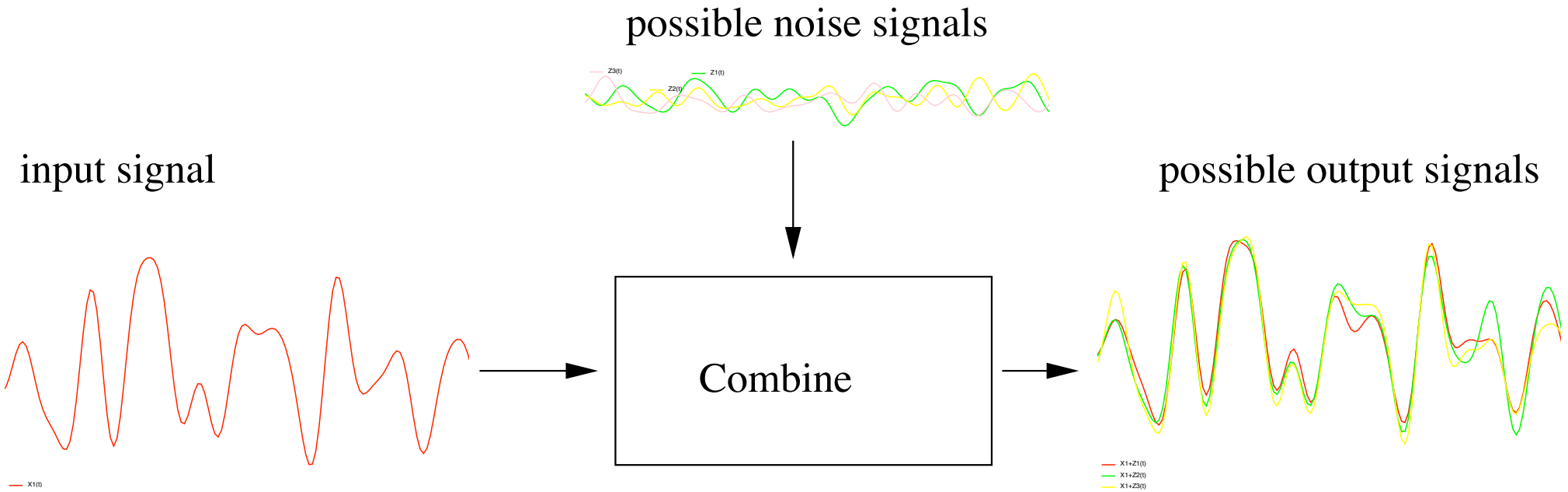
$\Rightarrow$ should be taken into account

Main question :

How to use continuous channels to transmit discrete information in a reliable fashion.

Models

1. Very general model : continuous input signal $\rightarrow$ continuous output signal



From a mathematical point of view :

Input : $\mathcal{X}(t)$ a continuous time stochastic process

Output : $\mathcal{Y}(t)$ another continuous time stochastic process

Channel model : $p(\mathcal{Y}(t)|\mathcal{X}(t))$ ... E.g. $\mathcal{Y}(t) = \mathcal{X}(t) + \mathcal{Z}(t)$

2. Band-limited additive noise channel : general enough in practice

Output signal has no frequencies above f_0.

Output is obtained by $\mathcal{Y}(t) = (\mathcal{X}(t) + \mathcal{Z}(t)) \overset{\text{conv}}{\otimes} h(t)$, where $h(t)$ is band-limited impluse response, and $\mathcal{Z}(t)$ is an *additive* noise process.

$\Rightarrow$ channel is blind to input and noise frequencies above f_0

$\Rightarrow$ input, noise and output become **discrete in time** (cf sampling theorem)

Further assumptions (to make life easier...)

We suppose that $h(f)$ is an ideal low pass filter $\Rightarrow \mathcal{Y}_i = \mathcal{X}_i + \mathcal{Z}_i$

We suppose that $\mathcal{Z}(t)$ is white noise $\Rightarrow \mathcal{Z}_i$ i.i.d.

$\Rightarrow$ Memoryless continuous channel characterized by $p(\mathcal{Z})$ or equivalently by $p(\mathcal{Y}|\mathcal{X})$.

Often, a good model for $p(\mathcal{Z})$ is the Gaussian noise model

$\rightarrow$ this will be our working model

Gaussian channel with average power limitation

Definition (Gaussian channel) :

- Input alphabet $\mathcal{X} = \mathbb{R}$
- Additive white Gaussian noise : $\mathcal{Y}_i = \mathcal{X}_i + \mathcal{Z}_i$ where $\mathcal{Z}_i \sim \mathcal{N}(0, N)$
  (N denotes the noise power)
- Power limitation : $\boxed{\frac{1}{n}\sum_{i=1}^{n} X_i^2 \leq P,}$ for any input message X^n of length n, $\forall n$
  Considering only stationary and ergodic input signals this implies $E\{\mathcal{X}^2\} \leq P$.

Definition (Information capacity of the Gaussian channel) :

$$C \stackrel{\triangle}{=} \max_{\{p(\mathcal{X}) \,:E_p\{\mathcal{X}^2\}\leq P\}} I(\mathcal{X};\mathcal{Y}) \text{ per channel use.}$$

$\rightarrow$ Need to extend definition of mutual information (and entropy) to continuous random variables.

Differential entropy and mutual information of continuous random variables

Direct extension of mutual information (OK) :

$$I(\mathcal{X};\mathcal{Y}) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p_{\mathcal{X},\mathcal{Y}}(x,y)\log\frac{p_{\mathcal{X},\mathcal{Y}}(x,y)}{p_{\mathcal{X}}(x)p_{\mathcal{Y}}(y)}dxdy.$$

Direct (incorrect) extension of entropy function :

$$H_c(p(\cdot)) = -\lim_{\Delta\to 0}\left[\sum_{i=-\infty}^{+\infty}(\Delta p(i\Delta))\log(\Delta p(i\Delta))\right].$$

But the value $\to +\infty$... (NOT OK)

Correct extension (so-called differential entropy) :

$$H_d(p(\cdot)) = -\lim_{\Delta\to 0}\left[\sum_{i=-\infty}^{+\infty}(\Delta p(i\Delta))\log(p(i\Delta))\right] = -\int_{-\infty}^{+\infty}[p(x)\log p(x)]dx.$$

Note that $H_d(f(\mathcal{X}))$ is not necessarily smaller than $H_d(\mathcal{X})$.

Differential entropies : examples and properties

Gausian r.v. $\mathcal{N}(\mu, \sigma^2)$:

$$H_d(\mathcal{X}) = \frac{1}{2}\log_2\left(2\pi e\sigma^2\right)$$

Two, jointly Gaussian random variables $\mathcal{N}(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$:

$$H_d(\mathcal{X}, \mathcal{Y}) = \frac{1}{2}\log\left((2\pi e)^2\sigma_1^2\sigma_2^2(1-\rho^2)\right)$$

Notice that, if $\mathcal{X} \perp \mathcal{Y}$ then $H_d(\mathcal{X}, \mathcal{Y}) = H_d(\mathcal{X}) + H_d(\mathcal{Y})$

Main property of differential entropies

$$\begin{aligned} I(\mathcal{X};\mathcal{Y}) &= H_d(\mathcal{X}) + H_d(\mathcal{Y}) - H_d(\mathcal{X},\mathcal{Y}) \\ &= H_d(\mathcal{X}) - H_d(\mathcal{X}|\mathcal{Y}) = H_d(\mathcal{Y}) - H_d(\mathcal{Y}|\mathcal{X}) \end{aligned}$$

In particular, for two jointly Gaussian r.v. :

$$I(\mathcal{X};\mathcal{Y}) = \frac{1}{2}\log\frac{1}{1-\rho^2}.$$

Information capacity of the Gaussian channel

$I(\mathcal{X};\mathcal{Y}) = H_d(\mathcal{Y}) - H_d(\mathcal{Y}|\mathcal{X})$

But, since $\mathcal{Y} = \mathcal{X} + \mathcal{Z}$ where $\mathcal{Z} \sim \mathcal{N}(0, N)$ and $\mathcal{Z} \perp \mathcal{X}$, we have also

$p(\mathcal{Y}|X) \sim \mathcal{N}(X, N)$ hence $H_d(\mathcal{Y}|X) = \frac{1}{2}\log_2(2\pi eN)$ for every X

Hence
$$H_d(\mathcal{Y}|\mathcal{X}) \stackrel{\triangle}{=} \int H_d(\mathcal{Y}|X)p(X)dX = \frac{1}{2}\log_2(2\pi eN)$$

$\Rightarrow$ need to choose $\{p(\mathcal{X}) \ : E_p\{\mathcal{X}^2\} \leq P\}$ so as to maximize $H_d(\mathcal{Y})$.

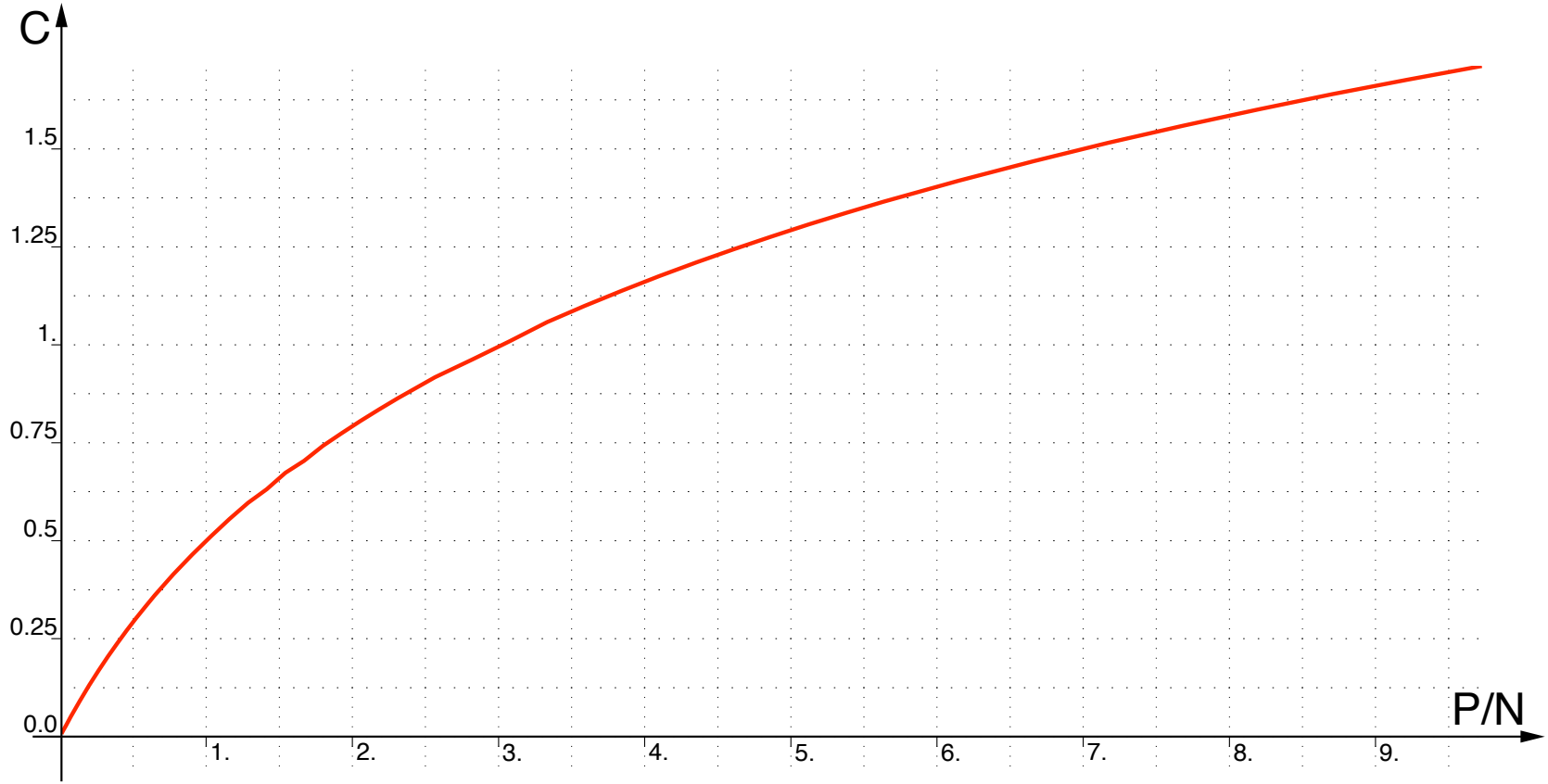
Solution : choose $p(\mathcal{X}) \sim N(0, P)$ (see notes)

Consequence : $p(\mathcal{Y}) \sim N(0, P+N) \Rightarrow H_d(\mathcal{Y}) = \frac{1}{2}\log_2(2\pi e(P+N))$

Conclusion :

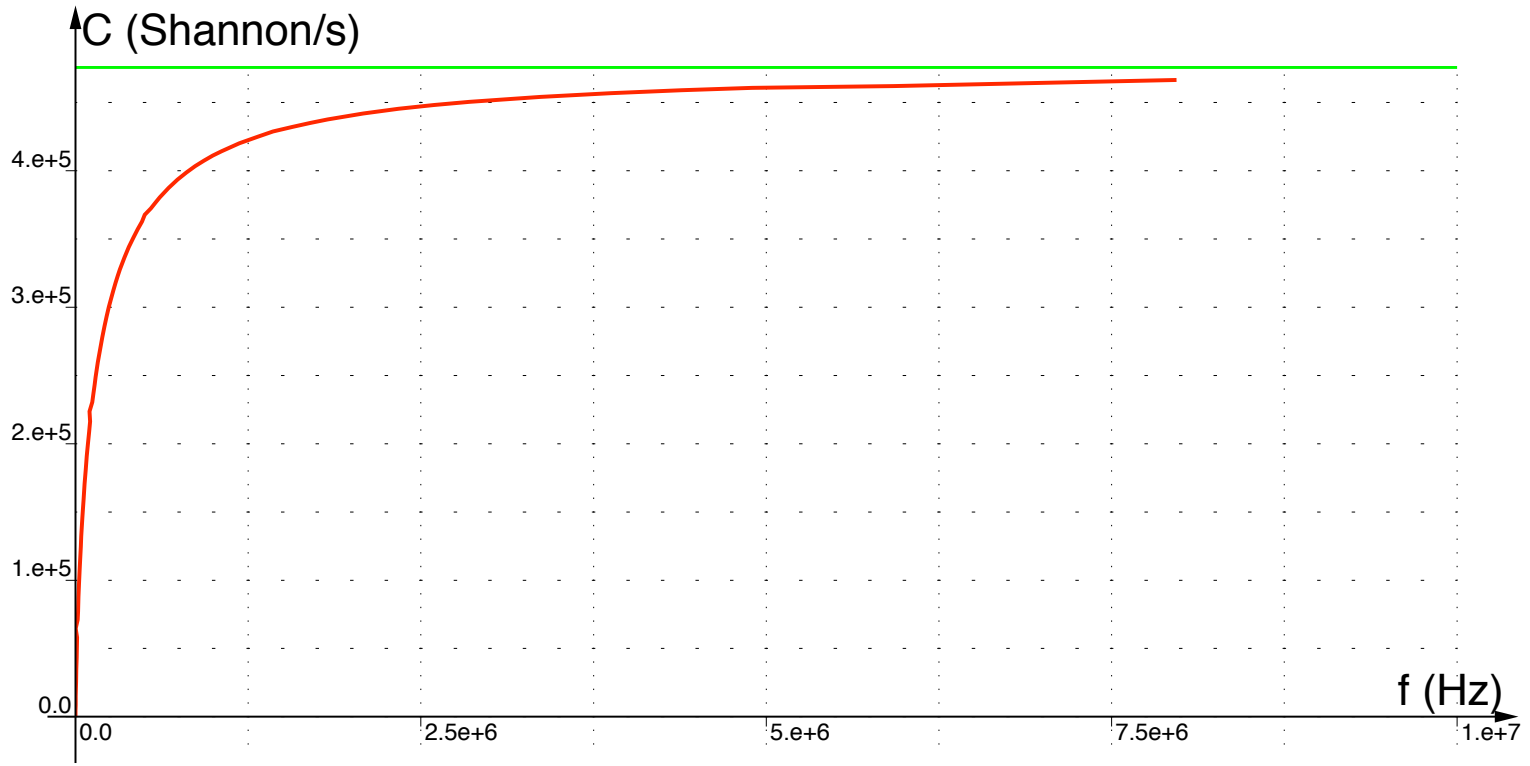
Capacity of Gaussian channel : $C = \frac{1}{2}\log\left(1 + \frac{P}{N}\right)$ Shannon/channel use

Information capacity of the Gaussian Channel



— Channel capacity as a function of P/N

If $P \to \infty$ or $N \to 0$ then $C \to \infty$



— Capacity as a function of bandwidth. P/N0 ratio = 330000

— Asymptotic limit

Capacity of band-limited channel with $\frac{N_0}{2}$ noise power spectral density and signal power P :

$$C = f_0 \log\left(1 + \frac{P}{f_0 N_0}\right) \text{ Shannon/second}$$

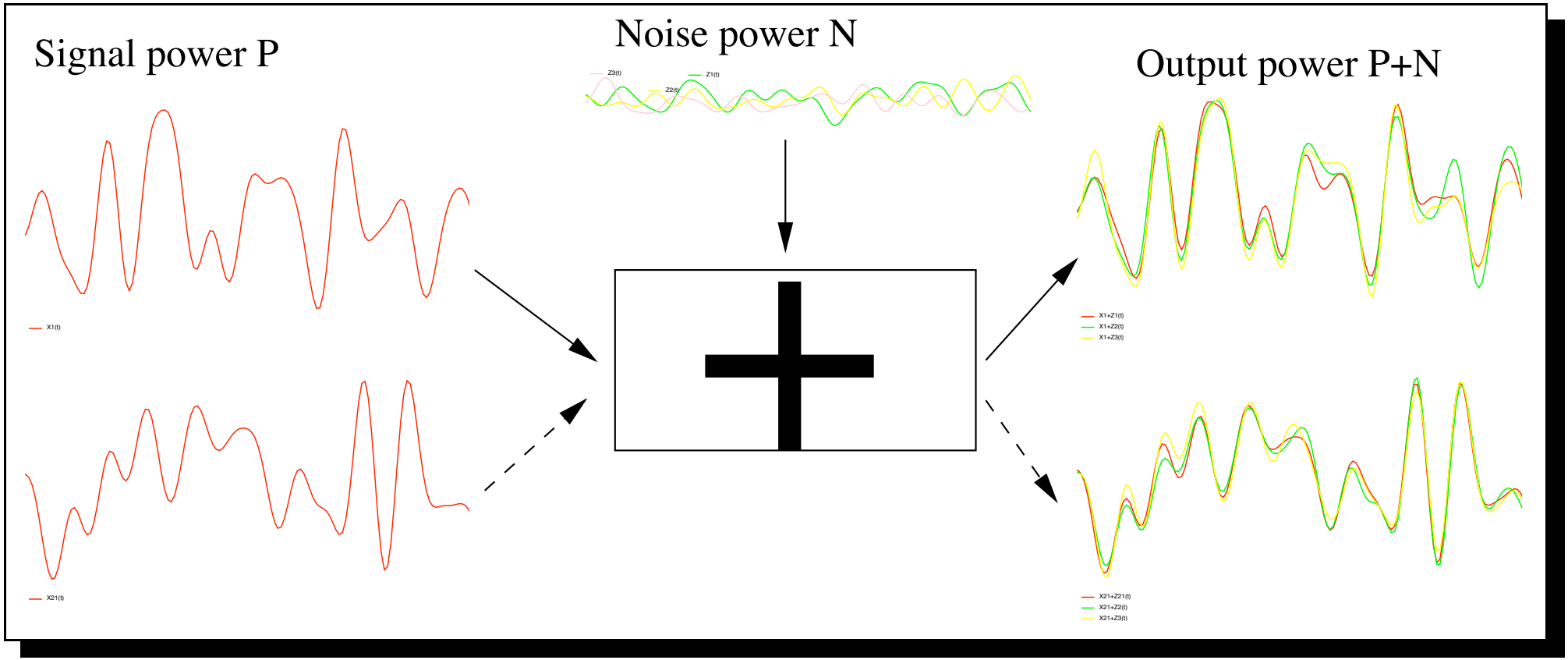
(i.e. $P/N = P/f_0 N_0$ and $2f_0$ channel uses per second)

NB : $\lim_{f_0 \to \infty} C(f_0) = \frac{P}{N_0} \log e$ (Shannon/second).

Second Shannon theorem for the continuous channel

Random coding ideas work in a similar way as for the discrete channel

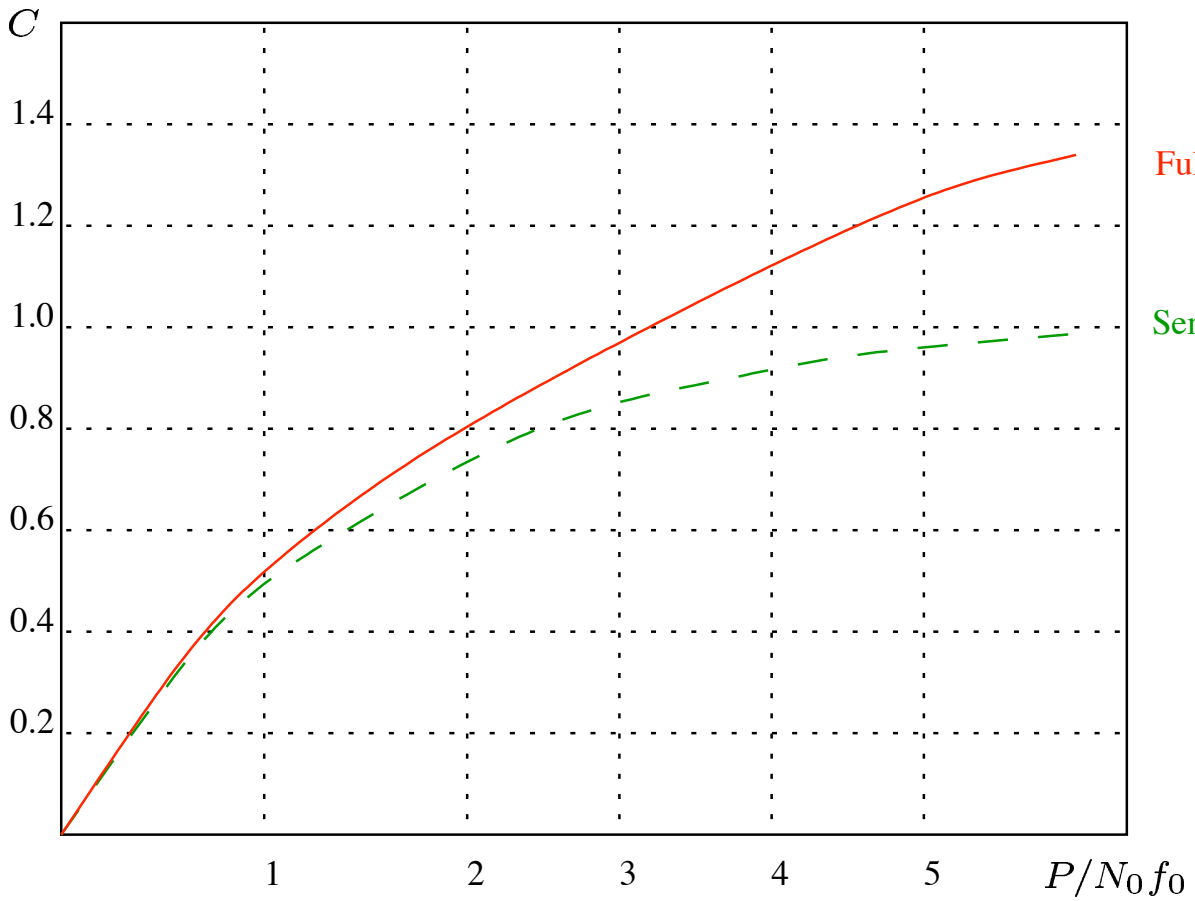
$\Rightarrow$ for long input code-words generated "à la random white noise of power P", a large proportion of random code-books correspond to good codes.



Modulation and detection on the continuous (Gaussian) Channel

Ideal modulation (random Gaussian coding) not feasible (cf decoding difficulties)

Choose discrete set of signals to build-up code-words and then discrete coding theory to generate long code-words. E.g. : binary signalling : $\mathcal{X} = \{-\sqrt{P}, +\sqrt{P}\}$.



Pente à l'origine : $\frac{1}{2 \ln 2}$

Caveat : because of input alphabet restriction we reduce capacity.

But not if SNR is low.

Semi-analogical mode of operation

Input levels are chosen in a discrete catalog.

Code-words are built up as sequences of the discrete input alphabet.

The decoder uses the sequence of real number produced at the output (continuous output alphabet) to guess input word : **soft decoding decisions**

NB.

The number of levels that should be used depends on the SNR. The higher the SNR, the higher the number of levels.

If broad-band communication ($f > P/N0$) : SNR< 1 binary modulation is OK.

Full discrete use of the continuous Channel

Discrete input alphabet, and hard symbol by symbol decisions at the output.

$\Rightarrow$ leads to discrete channel, with transition matrix determined by number and value of discrete signal levels and N_0.

E.g. Binary input and binary output leads to binary symmetric channel.

Caveat : further loss of capacity.

E.g. in low SNR conditions, we loose about 36% of capacity.

(Read section 15.6 in the notes for further explanations about these different alternatives.)

Conclusions

Broad-band (or equivalently high noise) conditions : binary signalling with soft decoding.

Low noise conditions : large input alphabet required

Geometric Interpretations of Channel coding

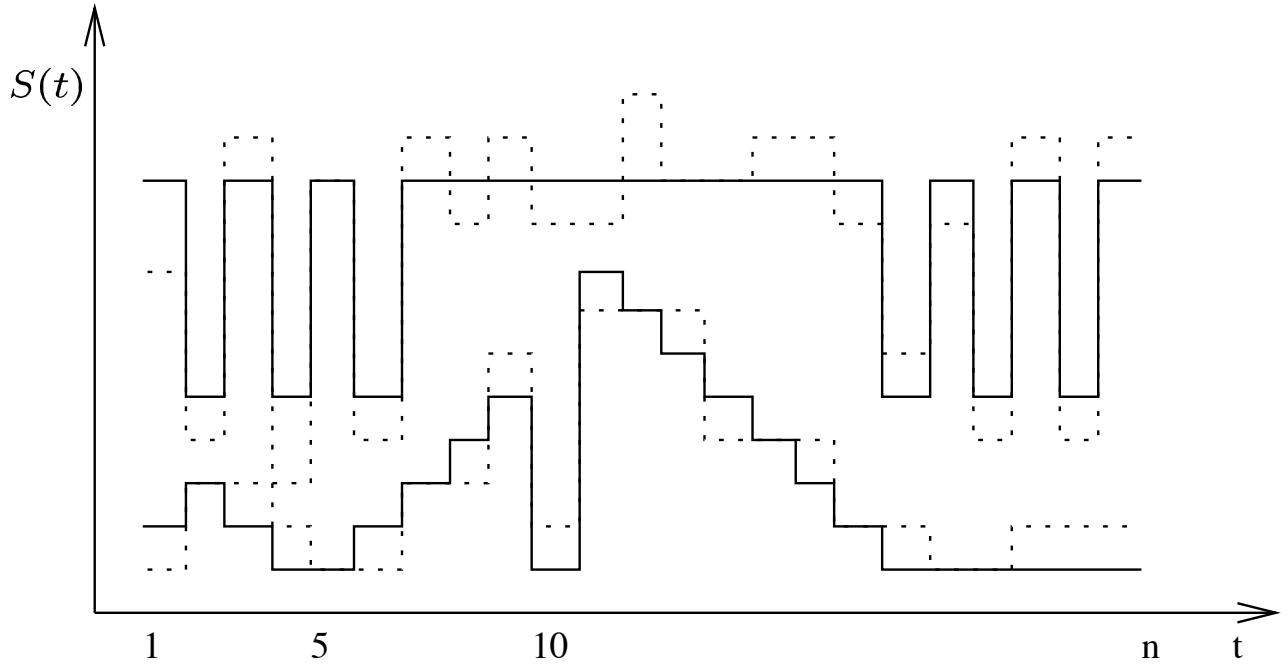
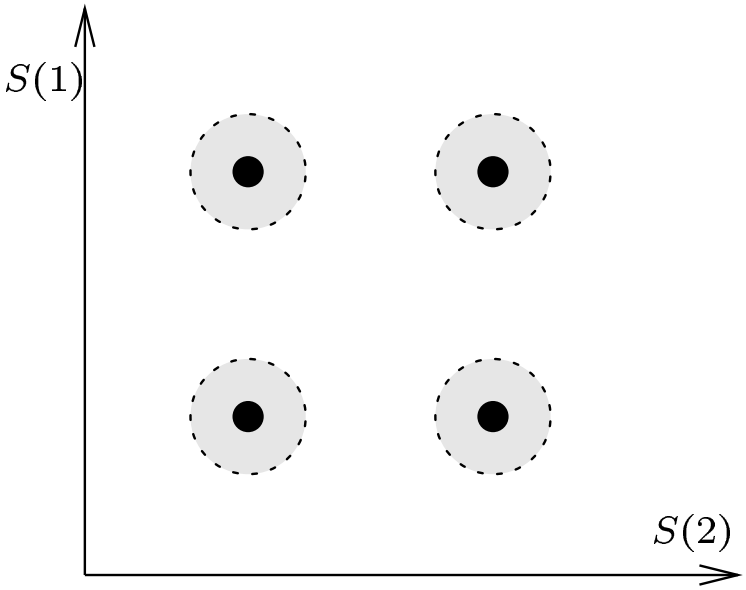
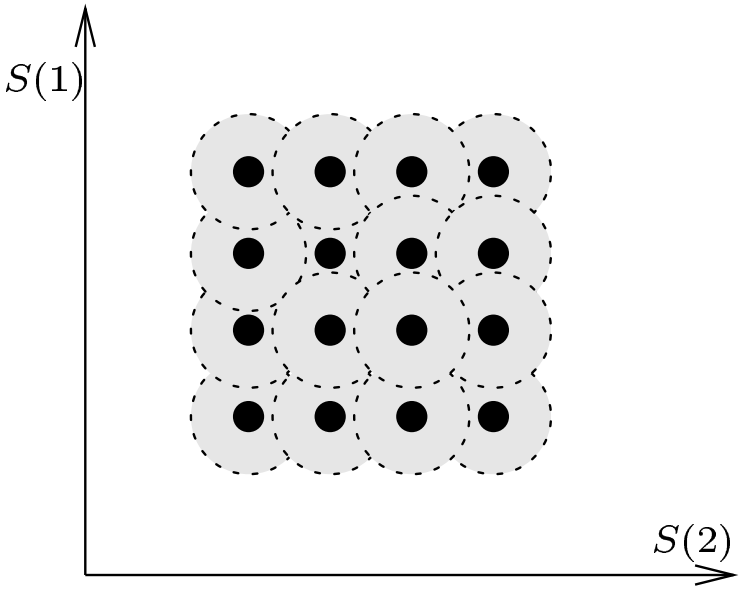


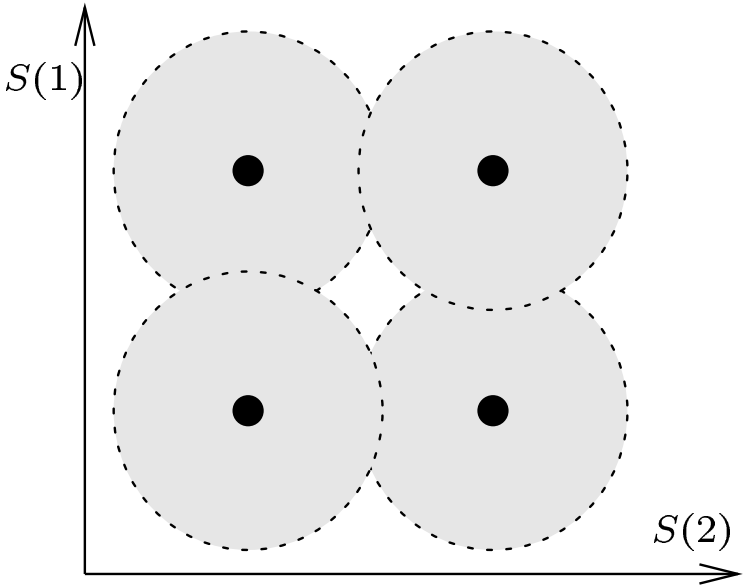
Figure 1 *Representation of messages by (discrete) temporal signals*



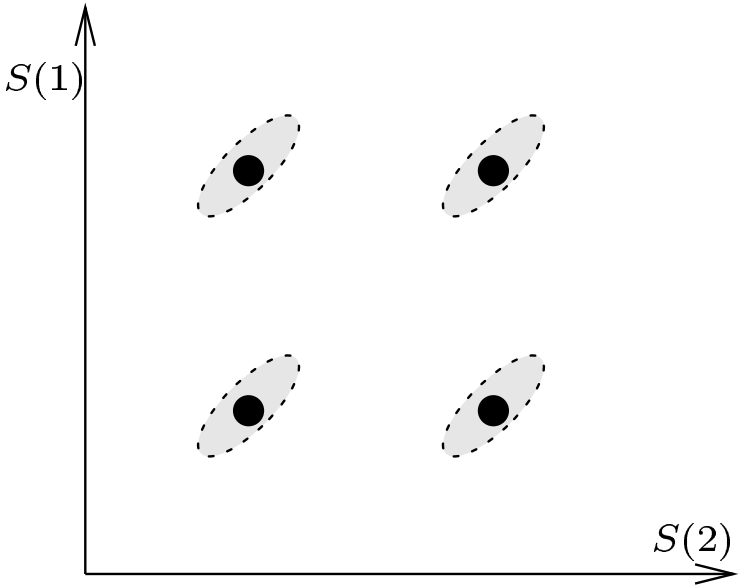
(a) Not enough codewords $R \ll C_1$



(b) Too many codewords $R \gg C_1$



(c) More noise : $R \gg C_2 \ll C_1$



(d) Correlated noise : $C_3 \gg C_1$

Channel coding and error control

Construction of good channel codes.

1. Why not random coding ?

Main problem : we need to use very long words (n very large) to ensure that $\lambda^{(n)}$ is small.

Since M increases exponentially with n, this means that the codebook becomes HUGE, and "nearest neighbor" decoding becomes unfeasible.

$\Rightarrow$ a good channel code is a code which at the same time is good in terms error correction ability and easy to decode.

Coding theory aims at building such codes, since 1950.

$\Rightarrow$ theory is significant (sometimes quite complicated) with rather deceiving results in terms of code-rates.

Recent progress is very significant :

- in 1993, Berrou et al. discover the Turbo-codes (low SNR : wireless, space)

- last 20 years, work on codes in Euclidean spaces (high SNR noise : modem).

Note

Codes for the binary symmetric channel.

Codes for the modulated Gaussian channel.

Codes for other types of channels (fading, burst errors...)

Practical impact ($\Rightarrow$ there is still a lot of work to be done)

Reduce energy consumption : weight, size, autonomy (e.g. GSM, satellites...)

Work under highly disturbed conditions (e.g. magnetic storms...)

Increase bandwidth of existing channels (e.g. phone lines).

Menu : for our brief overview of channel codes...

Linear bloc codes (cf. introduction)

Convolutional codes and trellis representation

Viterbi algorithm

BCJR algorithm (just some comments on it, and connection with Bayesian networks)

(Combination of codes (product, concatenation))

Turbo-codes

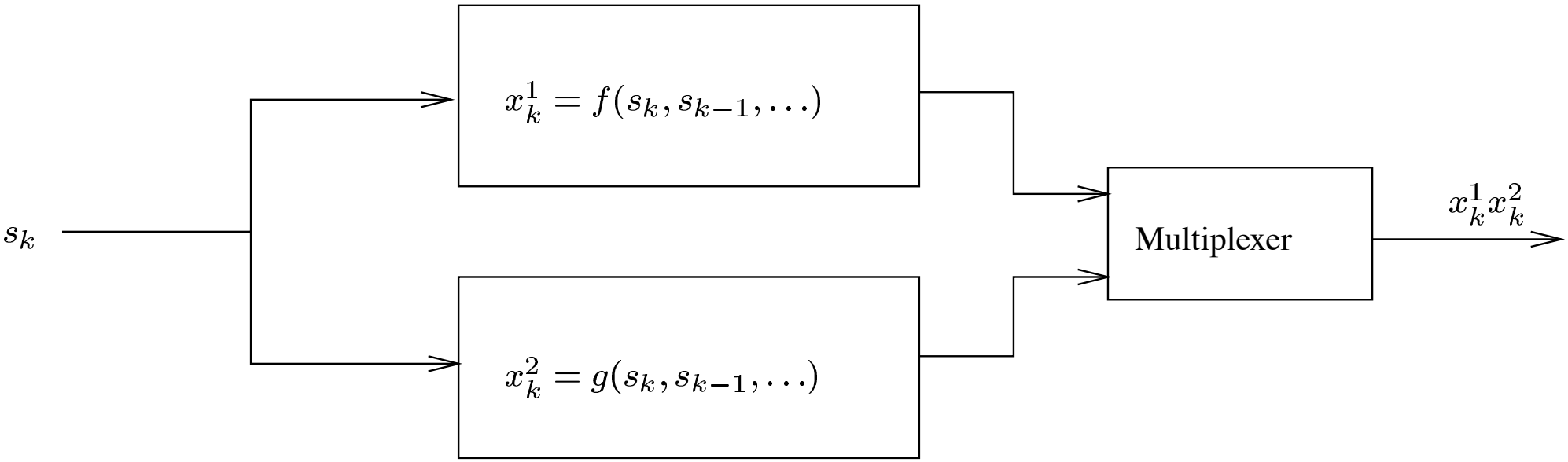
Convolutional Codes

Linear codes (but not bloc codes).

Idea : the signal stream (sequence of source symbols) s^N feeds a linear system (with memory) which generates an output sequence.

System = encoder : initially at rest.

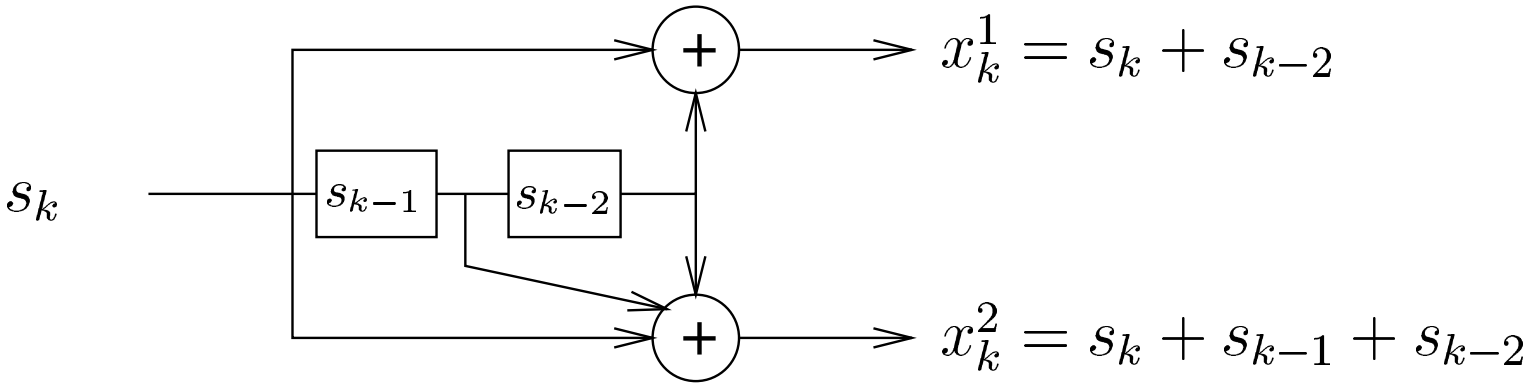
Output sequence is redundant, but partly like a random string



Decoder

Example (simple) : rate 0.5 convolutional code

$s_k \in \{0, 1\}$ and modulo-2 arithmetic



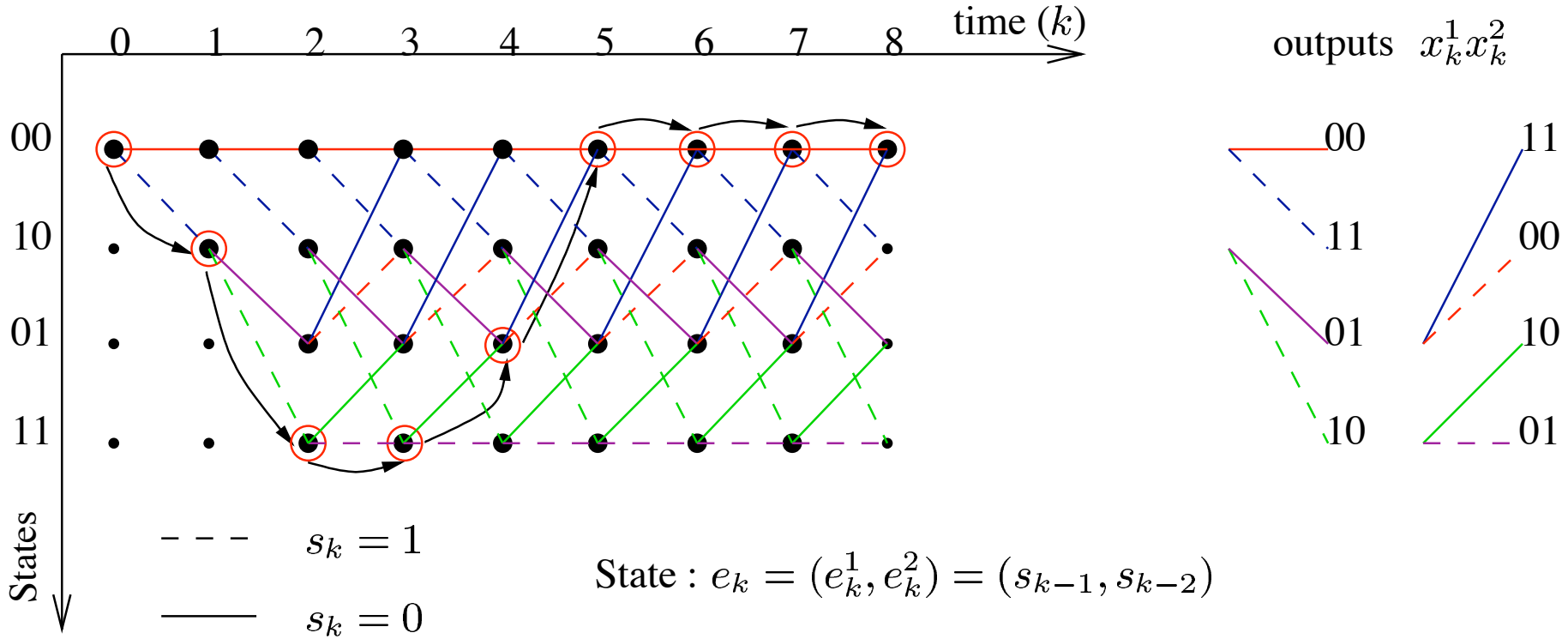
Memory : state of two registers $\{s_{k-1}, s_{k-2}\}$: $2^2 = 4$ possible states.

Initial state (e.g.) : 00

In the absence of noise, one can recover s_k in the following way : $s_0 = x_0^1$, and $s_1 = x_1^1$, $s_k = x_k^1 + s_{k-2}$

Trellis diagram

Simulation of the operation of the encoder by using a graph : represents all possible sequences of states of the encoder, together with the corresponding outputs.



State of encoder : nb. of possible states q^m (m denotes number of memory bits)

From each state there are exactly two starting transitions.

(after s_3, two transitions converge towards each state.)

Trellis decoding

1. It is clear that to each input word corresponds one path through the trellis.

2. Code is uniquely decodable : to $\neq$ input sequences $\rightarrow \neq$ paths.

3. Message sent (codeword) : X^N (alphabet $q = 4$); received message : Y^N ($q = 4$).

4. Find $\hat{X}^N$ such that $P(X^N \neq \hat{X}^N)$ is minimal.

5. $\Rightarrow$ choose $\hat{X}^N$ such that $P(\hat{X}^N|Y^N) \geq P(X^N|Y^N), \forall X^N$.

6. Let us suppose that all X^N are a priori equiprobable : maximize $P(Y^N|X^N)$.

7. Channel without memory : $P(Y^N|X^N) = \prod_{i=1}^{N} P(Y_i|X_i)$

8. Minimize : $-\log P(Y^N|X^N) = \sum_{i=1}^{N} -\log P(Y_i|X_i)$

9. $\Rightarrow -\log P(Y_i|X_i)$ mesure the "costs" of the trellis arcs (branches)

10. $\Rightarrow$ find the least costly path of length N through the trellis.

NB. Solution by enumeration : 2^N possible paths... (all possible s^N)

Viterbi algorithm

Based on the following property (path = sequence of states = sequence of branches) :

> **If e^{K+1} is an optimal path towards state e_{K+1},**
> **then e^{K} (prefix) is an optimal path towards e_K**

indeed, otherwise...

Thus : if we know the q^m optimal paths of length K leading towards each one of the q^m states (nodes in position K) and know also the costs of the transitions at stage K, we can easily find the optimal paths of length $K+1$ leading towards each one of the q^m states (nodes in position $K+1$).

Principle of the algorithm :

- one builds all optimal paths of length $0, 1, \ldots, N$

- we keep the least cost path of length N, and backtrack to find individual transitions, codeword and source bits

$\Rightarrow Nq^{m+1}$ operations (OK if q and m are not to big).

Discussion

Viterbi algorithm is applicable in more general situations :

- continuous output alphabet...

- non equiprobable source symbols

- works also for linear bloc codes

Viterbi not necessarily very efficient (depends on the structure of the trellis)

Simplified versions :

- hill-climbing method (on-line algorithm maintaining only one single path)

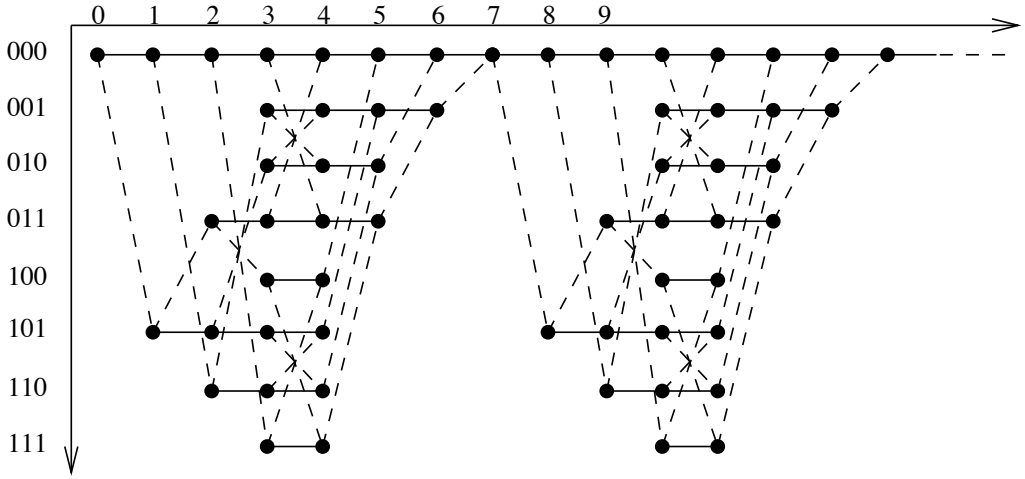
- beam-search...

Main drawback : need to wait for the end of message before we can decode

In practice : we can apply it to consecutive blocs of fixed length of source symbols, we can decode with a delay of the order of de $5 \times m$.

Trellis associated to a linear bloc code

E.g. for the Hamming $(7, 4)$ code



Trellis vs Bayesian network

Trellis provides a feedforward structure for the dependence relations among successive source symbols and code symbols $\Rightarrow$ can be translated directly into a Bayesian network.

We need to add to the Bayesian network the model of the channel : if causal and no feedback $\Rightarrow$ OK to use Bayesian network.

In Bayesian networks we use a generalized version of Viterbi algorithm to find the most probable explanation of a certain observation.

Per source bit decoding vs per source word decoding

Viterbi finds the most probable source (and code) word, given the observed channel outputs $\Rightarrow$ minimizes word error rate.

Suppose we want to minimize bit error rate : we need to compute for each source symbol $P(s_i = 0|Y^N)$, and choose $s_i = 0$ if this number is larger than 0.5 (otherwise we choose $s_i = 1$).

It turns out that there is also an efficient algorithm (called forward-backward algorithm, or BCJR algorithm) which computes these numbers in linear time for all source bits.

A generalized version of this algorithm is used in Bayesian belief network when you query the posterior probability distribution of an unknown variable given the observed variables.

A posteriori probability distribution of source words

Suppose we observe sequence Y^N at the output of the channel, and know prior probabilities of source words and know also the code.

How should we summarize this information, if we want to communicate it to somebody else, who doesn't know anything about decoding ?

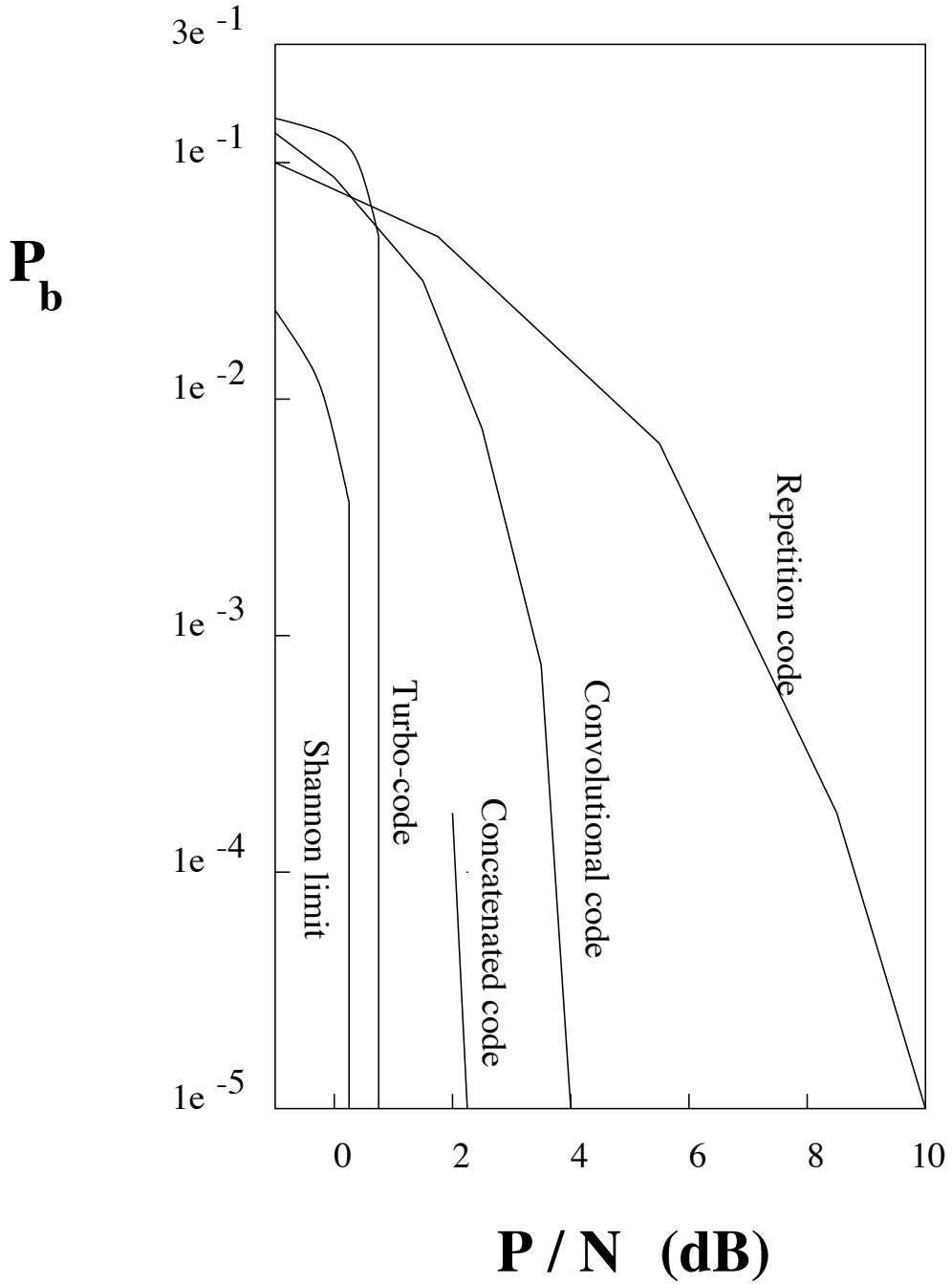
$\Rightarrow$ we need to provide $P(s^N|Y^N)$ (2^N numbers).

We can approximate this by : $P(s_1|Y^N)P(s_2|Y^N)\cdots P(s_N|Y^N) \Rightarrow$ only N numbers (computed by the BCJR algorithm in linear time).

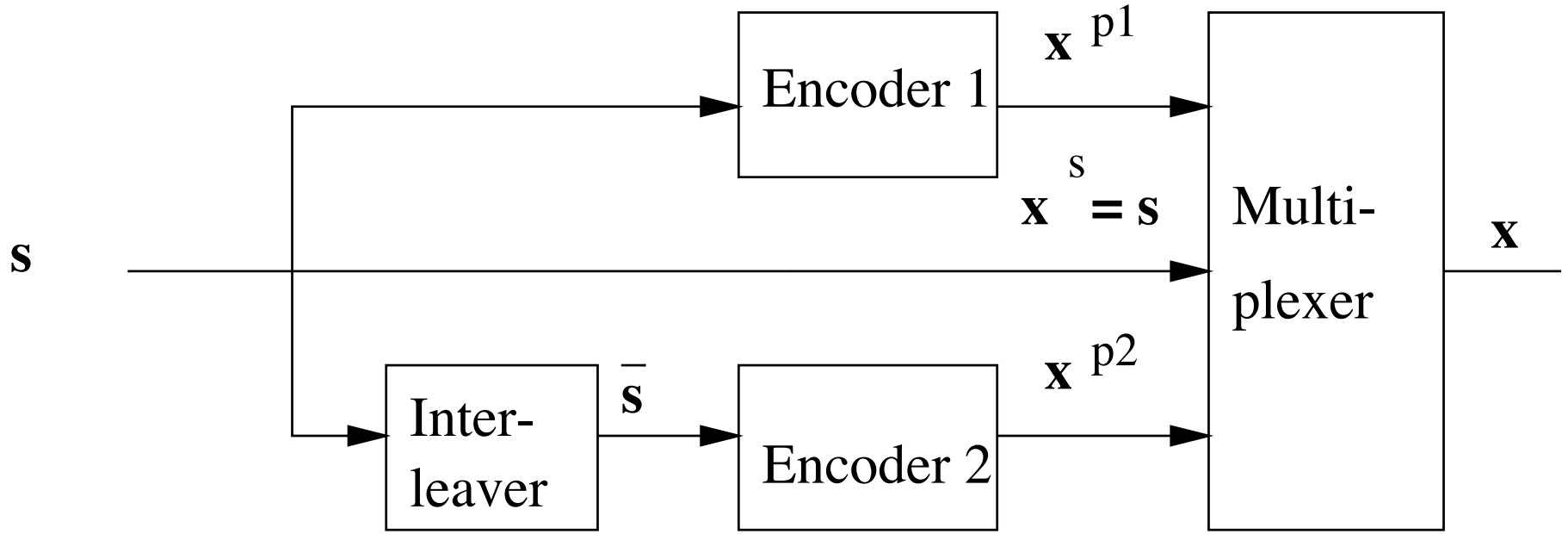
The BCJR algorithm works even if we drop some of the Y_i symbols (suppose we don't transmit them) : we have more flexibility in terms of code rate.

Now you know all you need to understand how Turbo-Codes work

Turbo-codes : *Bit error rate of different channel codes of rate R= 1/2*



Overview of the encoder of a Turbo-code (rate 1/3)



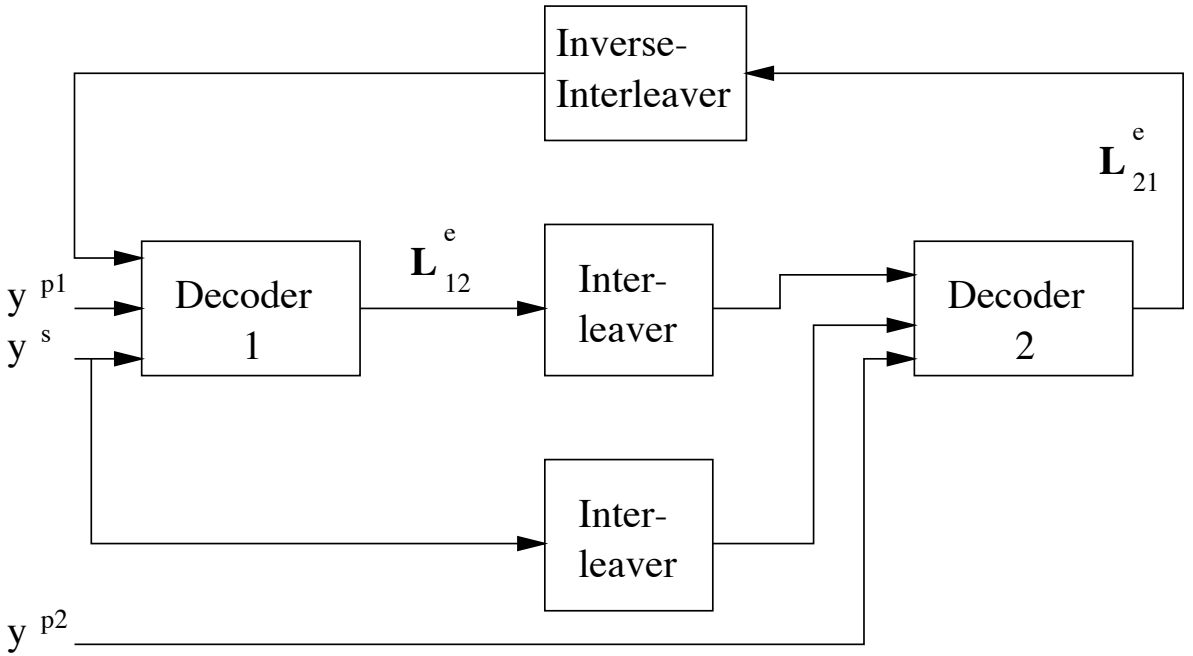
Components of the encoder

Encoder 1 and 2 are generally identical, so-called recursive convolutional encoders.

Interleaver : computes a fixed "random" permutation of the source symbols

$\Rightarrow$ resulting code has good properties in terms of interword distances but is difficult to decode.

Iterative “relaxed” decoding algorithm



Decoders 1 and 2 use BCJR algorithm to compute the posterior probability of each source bit given the corresponding parity bits and prior probabilities of source bits.

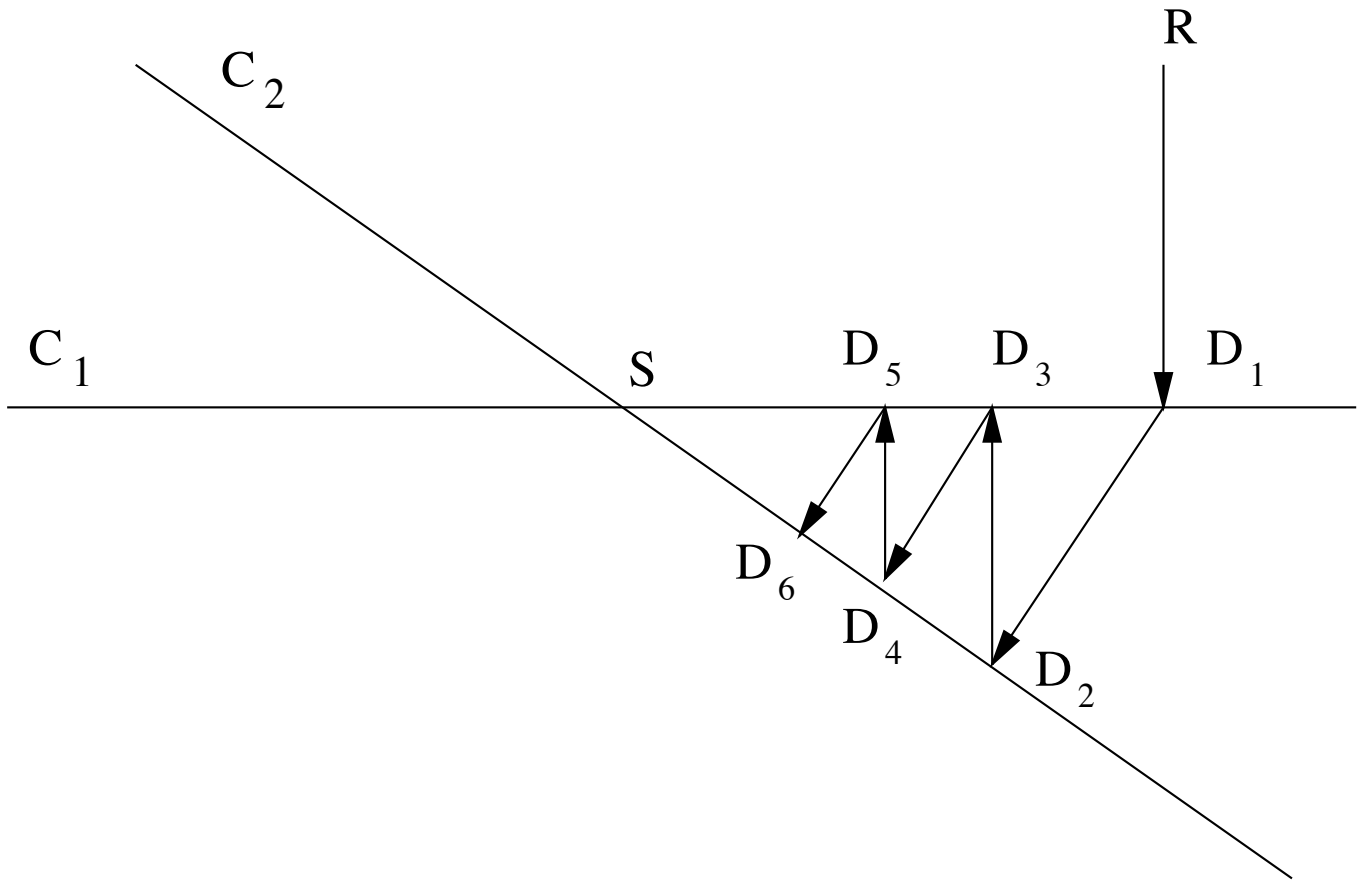
At the first stage, prior probabilities of source bits are equal to the actual prior probabilities (generally uniform).

After one decoder has computed the posteriors he hands this information over to the other decoder who uses these numbers to refresh his own priors etc.

Why does is work ?

Linear code is linear subspace $\Rightarrow$ decoding is like projecting the received word on this subspace.

Turbo code : one very big code which can be viewed as the intersection of two smaller codes (corresponding to 2 subspaces) :



Claude Shannon