EEE8099 – Information Theory & Coding
EEE8104 – Digital Communications

S. Le Goff

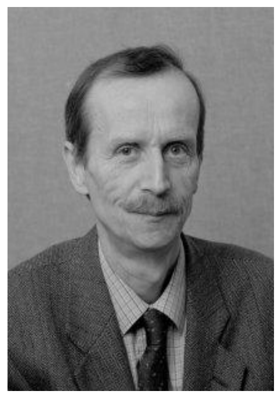
School of Engineering @ Newcastle University

Part 7
Practical Error-Correcting Codes
1993 – Today
Turbo Codes – Part 1

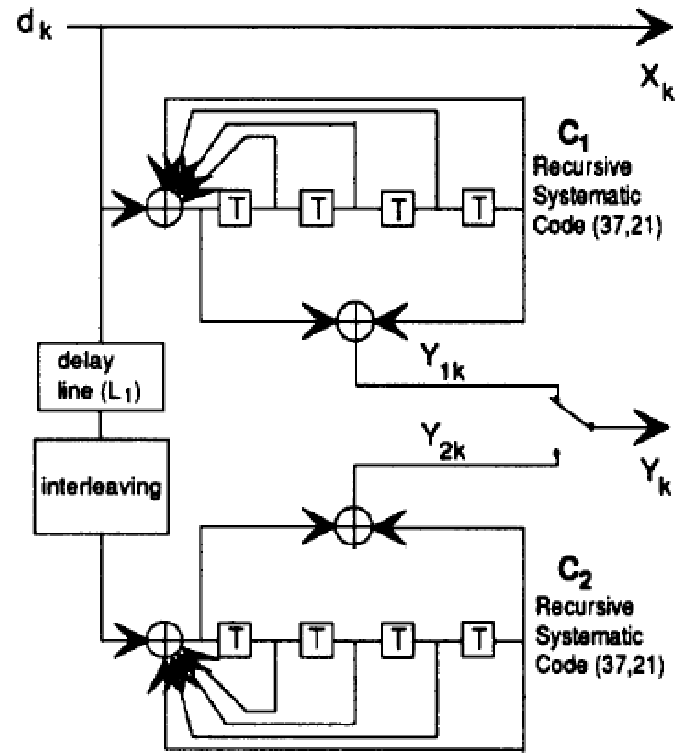
The turbo revolution (1993)

1993: Claude Berrou and Alain Glavieux introduce turbo codes*, a class of codes that perform very close to the capacity limit.

Claude Berrou



Alain Glavieux
(1949 - 2004)



The first turbo-encoder
($R_c = \frac{1}{2}$)

* With the help of their PhD student Punya Thitimajshima (1955-2006)

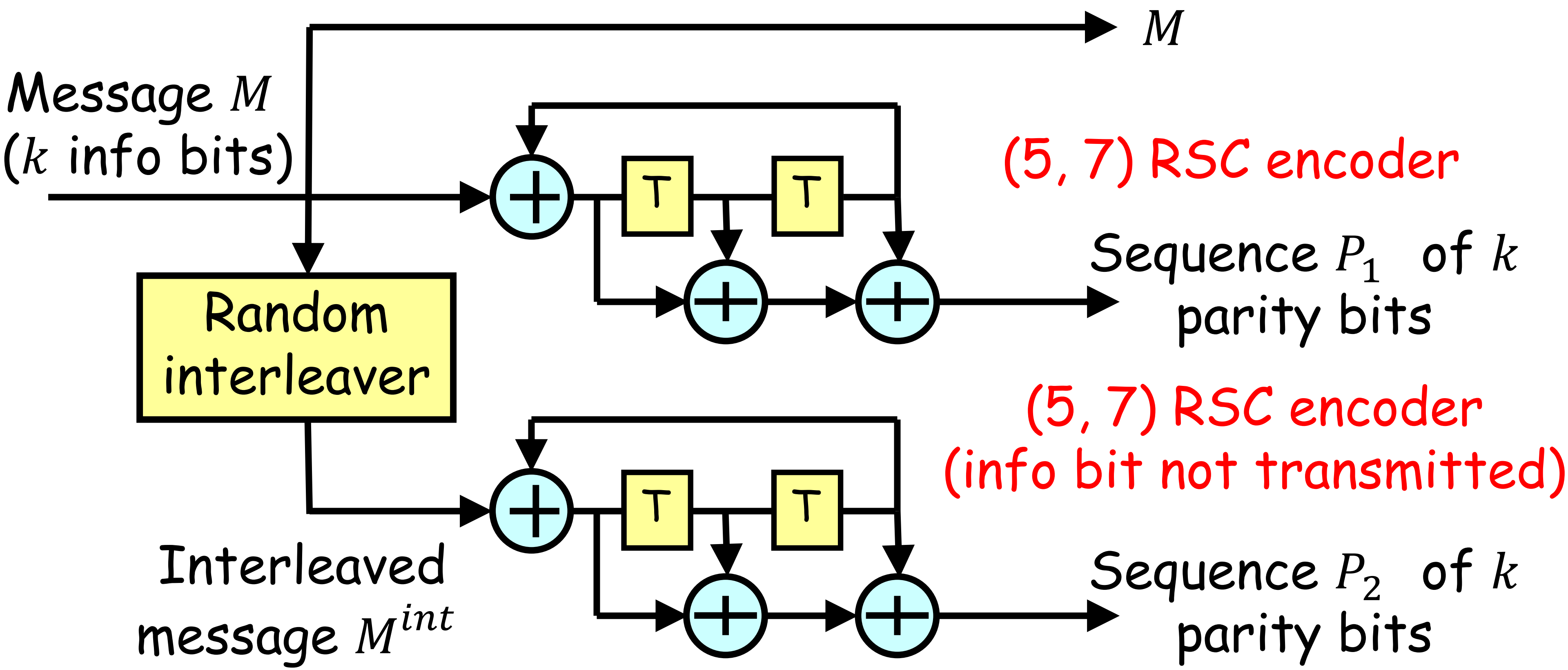
Turbo codes

Prof. Claude Berrou served as an external PhD examiner at our School.

Photo taken in December 2007, a few minutes after the PhD viva of Dr. Boon Kien Khoo.

Turbo codes

Turbo codes are parallel-concatenated recursive and systematic convolutional (RSC) codes: they are designed by concatenating "in parallel" two RSC codes, as shown in the example below.



Turbo codes

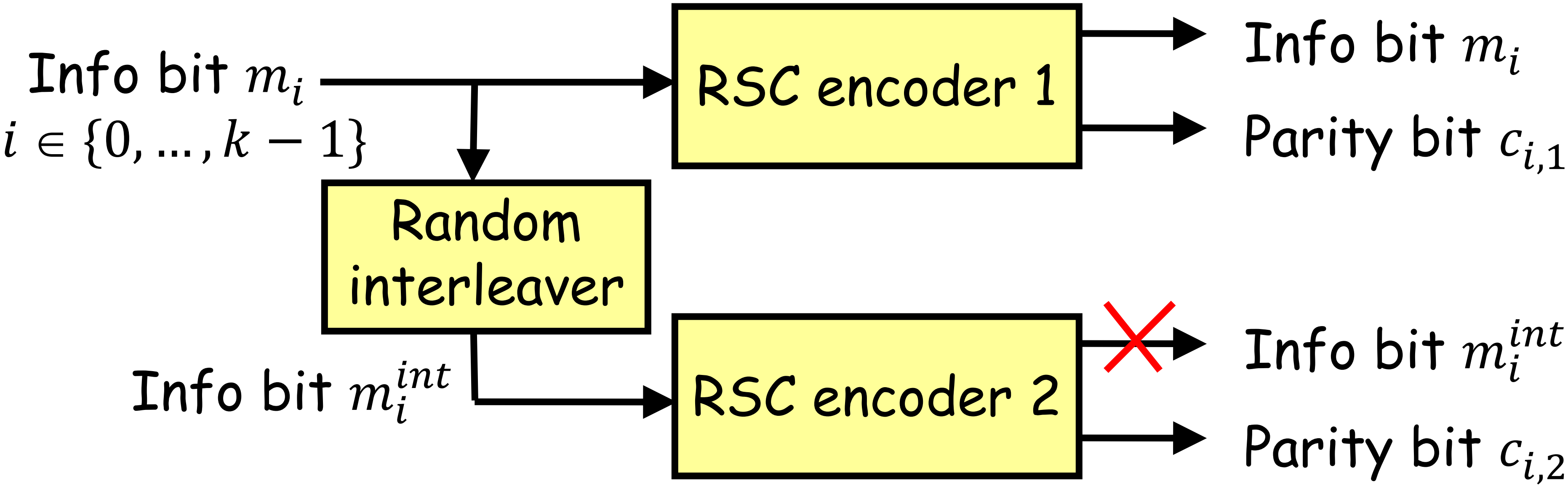
Notations

$$
\begin{aligned}
M &= (m_0, m_1, m_2, \dots, m_{k-1}),\\
P_1 &= \left(c_{0,1}, c_{1,1}, c_{2,1}, \dots, c_{k-1,1}\right),\\
P_2 &= \left(c_{0,2}, c_{1,2}, c_{2,2}, \dots, c_{k-1,2}\right),
\end{aligned}
$$

with $m_i, c_{i,1},\ c_{i,2} \in \{-1, +1\}$.

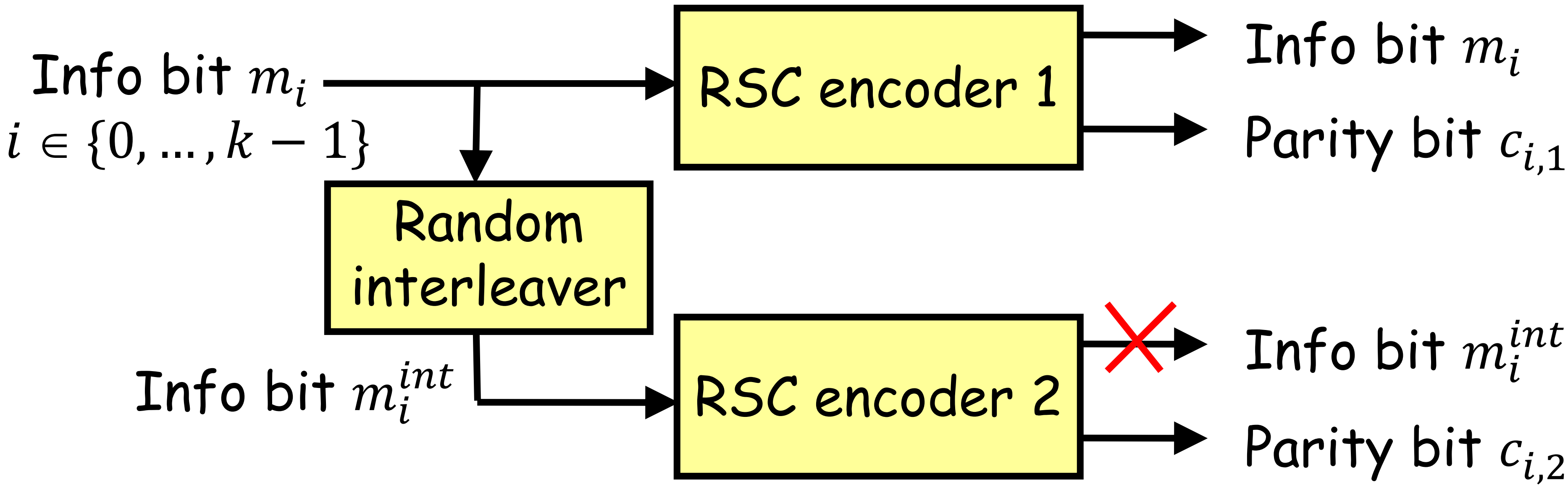
As usual, we prefer using the binary values -1 and +1 rather than the traditional 0 and 1 in order to account for the presence of the BPSK modulator between the encoder and the channel.

Turbo codes



The coding rate of a turbo-code is $R_c = \frac{1}{3}$, but higher rate can be obtained by periodically puncturing parity bits (same technique as with convolutional codes).

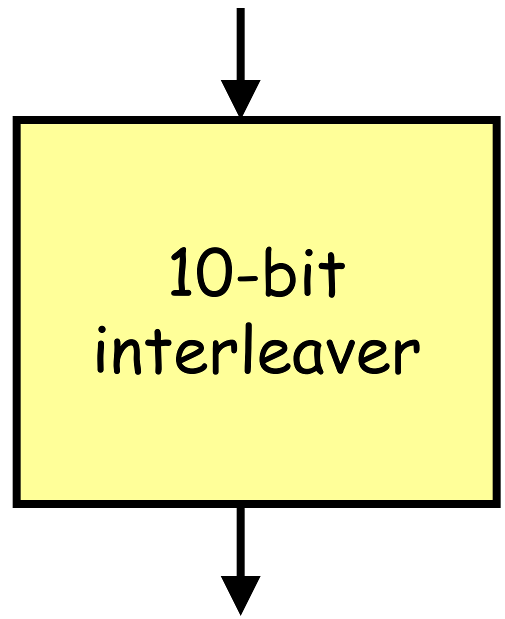
Turbo codes



The info bits m_i^{int}, $i \in \{0, \dots, k-1\}$, do not need to be transmitted because the sequence M^{int} is only an interleaved version of the message M. In fact, it is useless to transmit twice the same bits. This allows the coding rate to be equal to $\frac{1}{3}$ rather than $\frac{1}{4}$.

Turbo codes

$$M = (m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9)$$



$$M^{int} = (m_4, m_7, m_9, m_1, m_0, m_8, m_6, m_3, m_5, m_2)$$

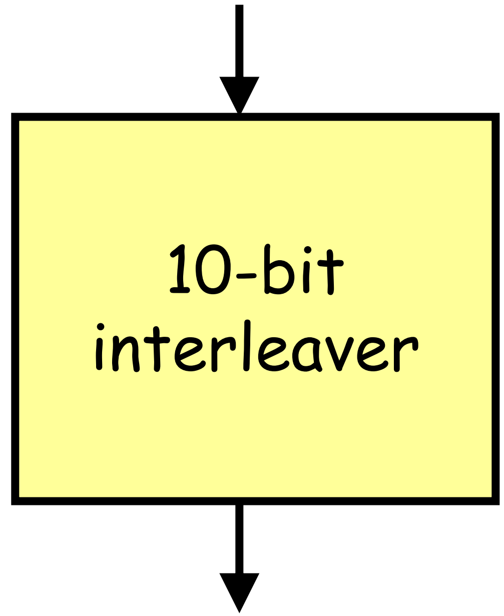
Example of a 10-bit interleaver

The message M is simply re-ordered by the interleaver.

The message interleaving is generally done pseudo-randomly, i.e. in a way that looks random.

Turbo codes

$$M = (m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9)$$



$$M^{int} = (m_4, m_7, m_9, m_1, m_0, m_8, m_6, m_3, m_5, m_2)$$
$$= (m_0^{int}, m_1^{int}, m_2^{int}, m_3^{int}, m_4^{int}, m_5^{int}, m_6^{int}, m_7^{int}, m_8^{int}, m_9^{int})$$

The 1st bit in the interleaved message M^{int} is the 5th bit in the message M. The 2nd bit in M^{int} is the 8th bit in M, etc.

The turbo decoder: iterative decoding algorithm

ML decoding is too complex to implement. We use an iterative decoding algorithm instead.

Samples available at the BPSK, AWGN channel output:

$R = (r_0, r_1, r_2, \dots, r_{k-1})$ with $r_i = m_i + n_i$,

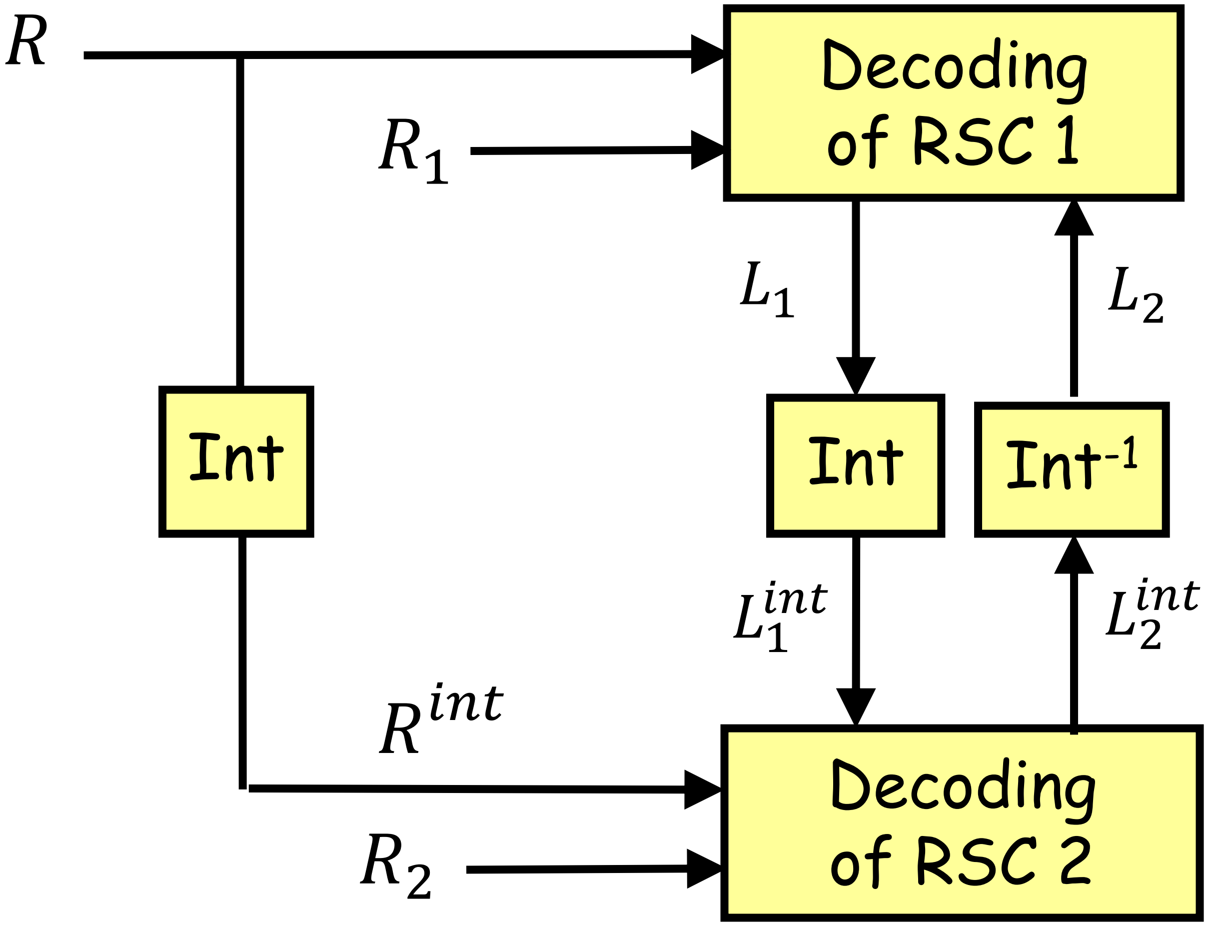
$R_1 = \left(r_{0,1}, r_{1,1}, r_{2,1}, \dots, r_{k-1,1}\right)$ with $r_{i,1} = c_{i,1} + n_{i,1}$,

$R_2 = \left(r_{0,2}, r_{1,2}, r_{2,2}, \dots, r_{k-1,2}\right)$ with $r_{i,2} = c_{i,2} + n_{i,2}$,

with $i \in \{0, \dots, k-1\}$ and $m_i, c_{i,1}, c_{i,2} \in \{-1, +1\}$.

The noise samples follow a Gaussian distribution, have a mean equal to zero, and a variance σ^2 that is a function of the channel SNR.

The turbo decoder: iterative decoding algorithm

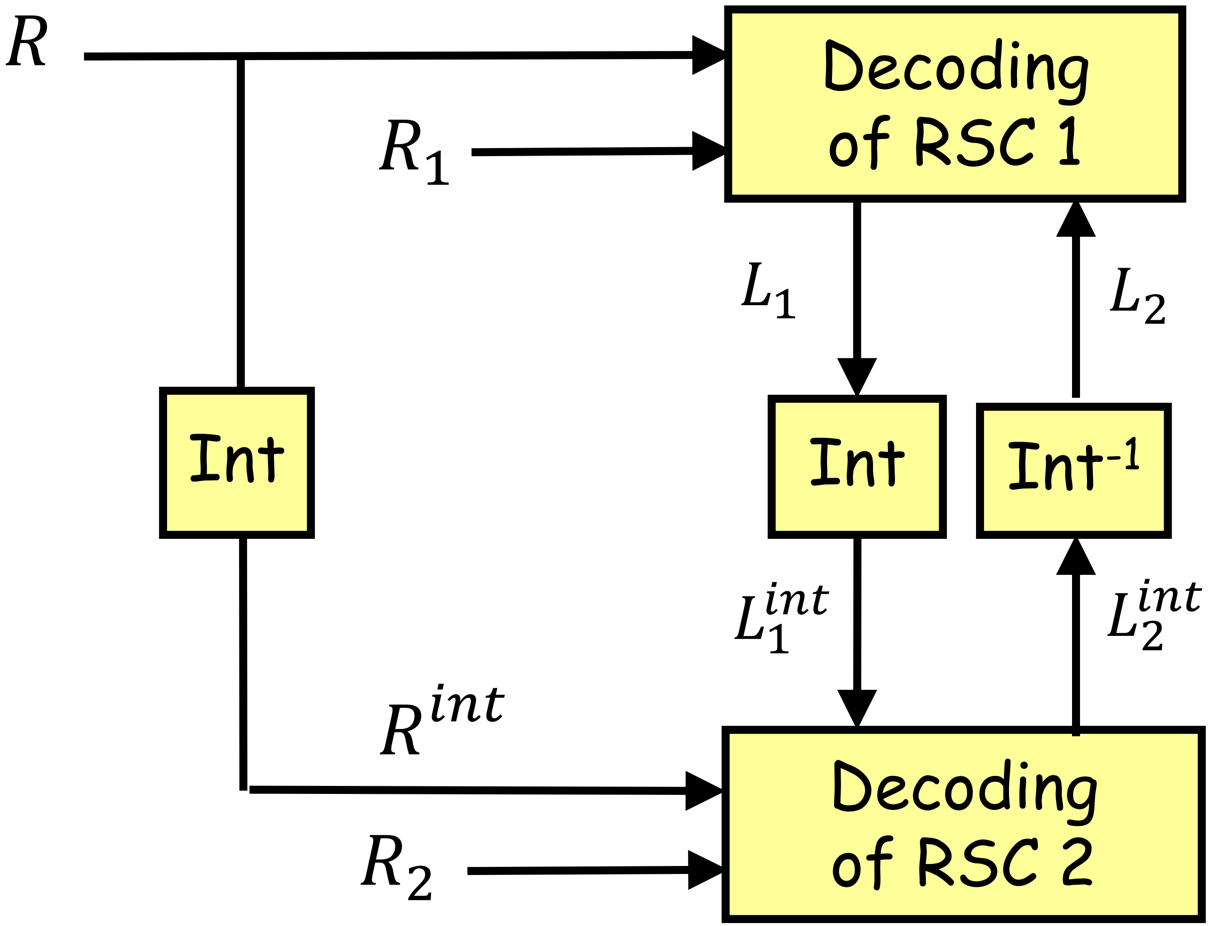


L_1, L_2 : New estimates of message M.

They are referred to as "extrinsic info".

Both elementary decoders exchange extrinsic info in order to improve their respective error-correction capabilities.

The turbo decoder: iterative decoding algorithm



$L_1 = (l_{0,1}, l_{1,1}, l_{2,1}, \ldots, l_{k-1,1})$ with $l_{i,1} = m_i + w_{i,1}$,

$L_2 = (l_{0,2}, l_{1,2}, l_{2,2}, \ldots, l_{k-1,2})$ with $l_{i,2} = m_i + w_{i,2}$,

with $i \in \{0, \ldots, k-1\}$ and $m_i \in \{-1, +1\}$.

The noise samples $w_{i,1}$ and $w_{i,2}$ follow zero-mean Gaussian distributions, with variances $\sigma_{1,ext}^2$ and $\sigma_{2,ext}^2$ that are not necessarily equal to the channel noise variance σ^2.

Decoding of RSC codes

Both decoders must produce soft decisions. They are referred to as soft-input, soft-output (SISO) decoders.

The Viterbi algorithm cannot be used to decode the RSC codes because it does not generate soft decisions. Recall that the Viterbi algorithm only generates hard decisions (0s and 1s).

Ii is possible to design a soft-output Viterbi algorithm (SOVA). The first one was in fact proposed by Joachim Hagenauer circa 1989.

But the SOVA is only a sub-optimal algorithm.

Decoding of RSC codes

The optimal algorithm to decode both RSC codes is the BCJR algorithm originally introduced in 1974 by L R Bahl, John Cocke (1925 – 2002), Frederik Jelinek (1932 – 2010), and Josef Raviv (1934 - 1999), and rediscovered by Alain Glavieux circa 1991.

It is an adaptation to convolutional codes of the more general maximum-a-posteriori (MAP) algorithm.

The MAP algorithm is optimal and simple in principle, but too complex to implement for practical codes.

The BCJR is a version of the MAP that is applicable to codes that can be described using a trellis.

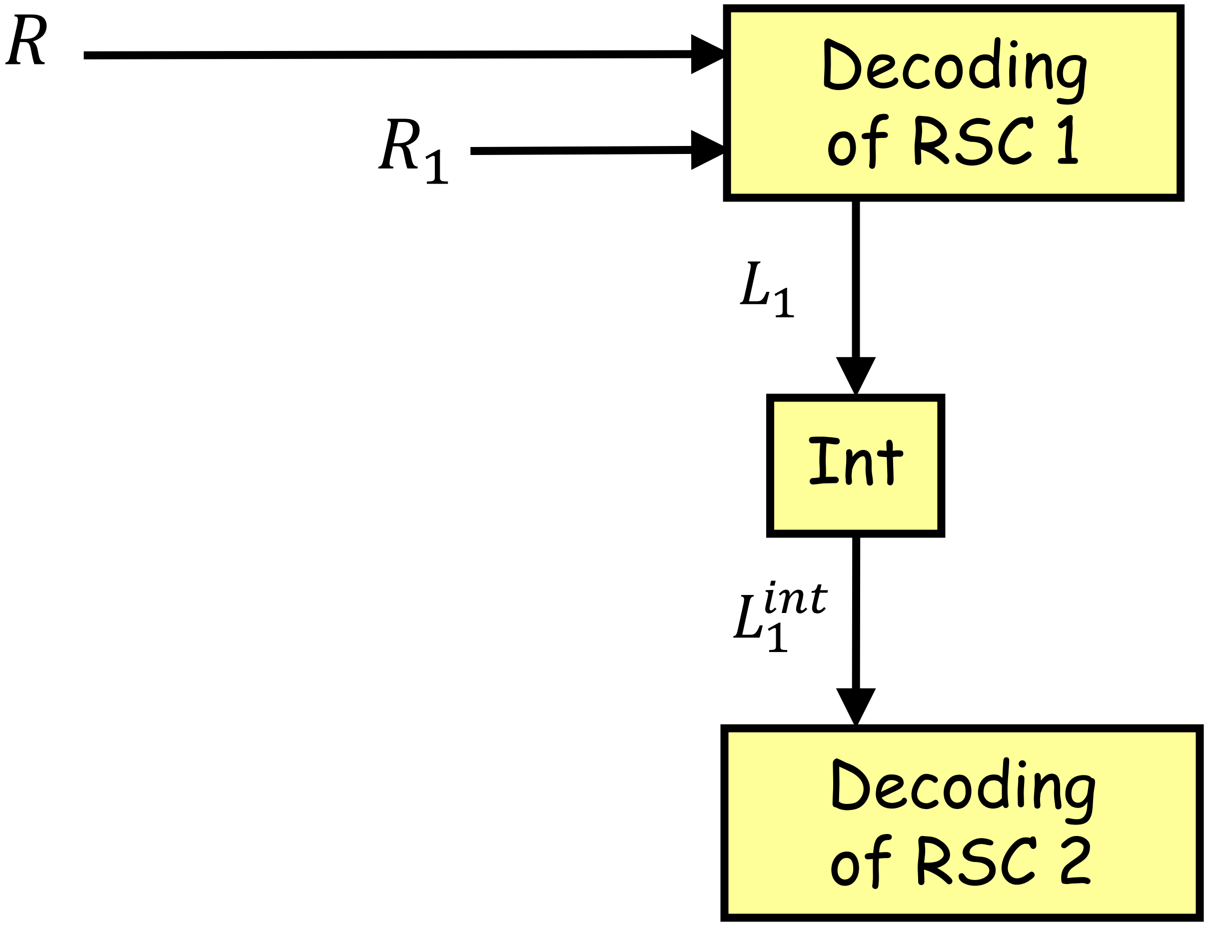
Decoding of RSC codes

The implementation complexity of the BCJR is reasonable, unlike that of the MAP.

The MAP/BCJR algorithm is a decoding algorithm that produces new estimates of the info bits based on the estimates it receives. It is thus a SISO algorithm.

Joachim Hagenauer once said that a SISO decoder could be seen as a “SNR amplifier”: it does not attempt to produce a binary decision. Instead, it generates new estimates of the info bits which are (on average) less noisy than the estimates present at its input.

How does the iterative decoding algorithm work in practice?

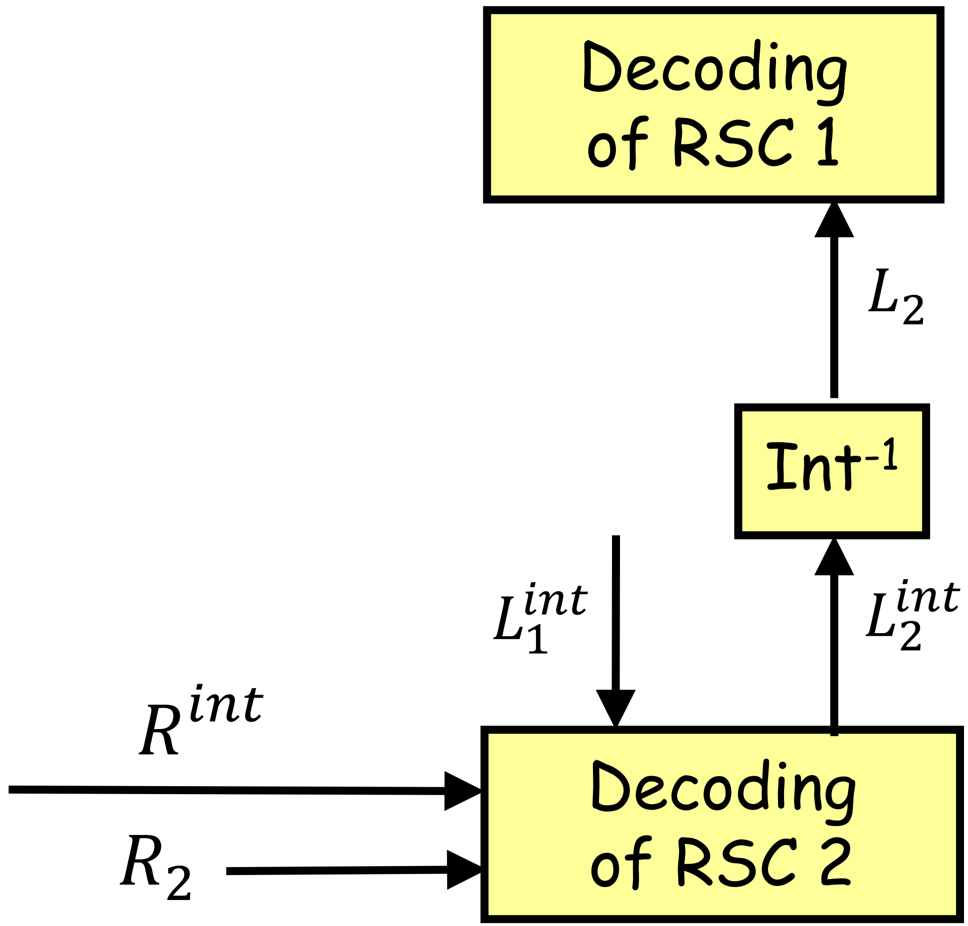


First decoding iteration

Using the channel samples only, the first decoder produces a new estimate, L_1, of message M.

L_1 is interleaved and sent to the second decoder.

How does the iterative decoding algorithm work in practice?

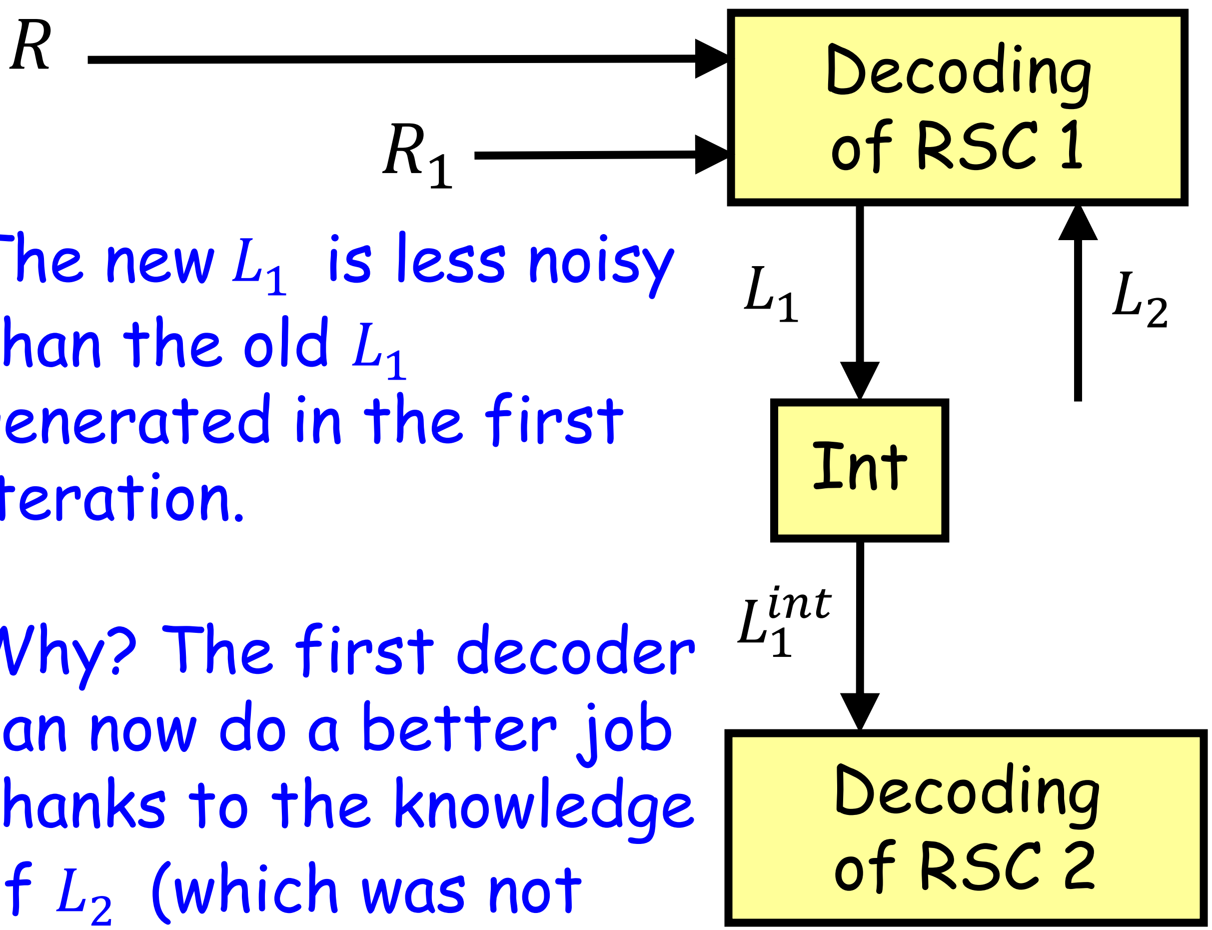


First decoding iteration

Using the channel samples and the extrinsic info L_1^{int}, the second decoder generates a new estimate, L_2^{int}, of the interleaved message M^{int}.

L_2^{int} is de-interleaved and sent to the first decoder.

How does the iterative decoding algorithm work in practice?



Second decoding iteration

Using the channel samples and the extrinsic info L_2, the second decoder generates a new estimate, L_1, of M.

L_1 is interleaved and sent to the second decoder.

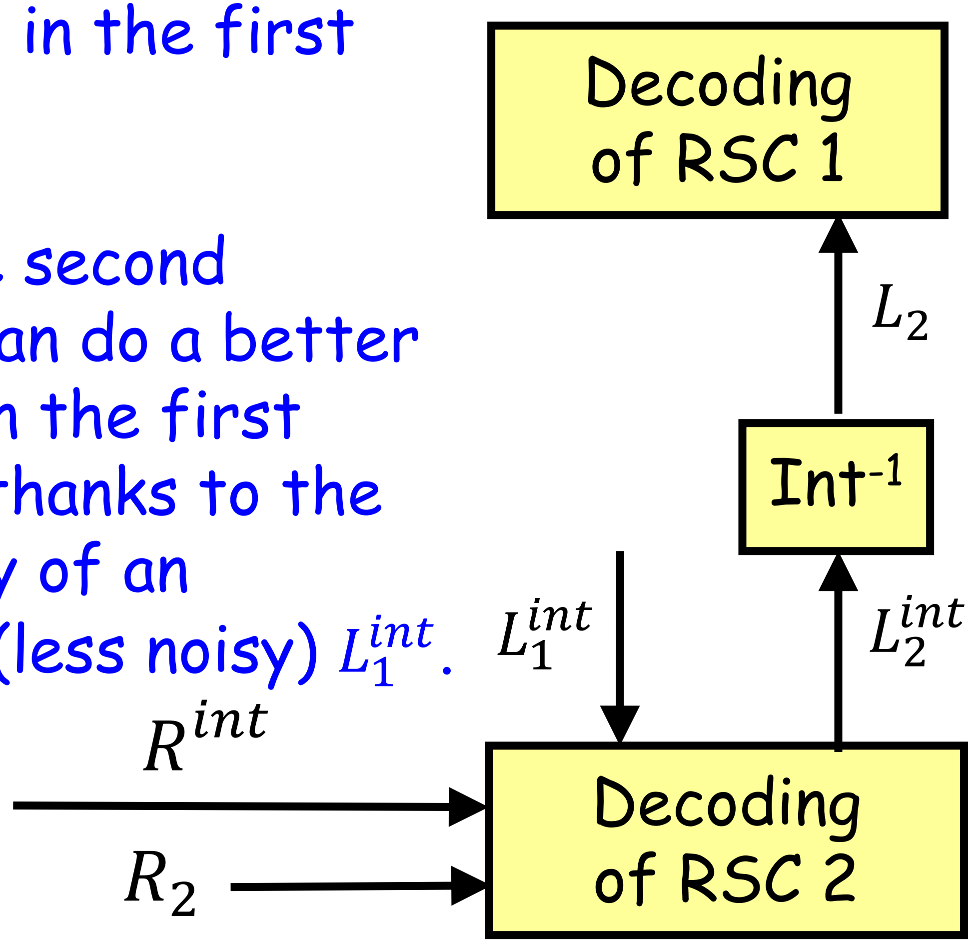
The new L_1 is less noisy than the old L_1 generated in the first iteration.

Why? The first decoder can now do a better job thanks to the knowledge of L_2 (which was not available in the first iteration).

How does the iterative decoding algorithm work in practice?

The new L_2^{int} is less noisy than the old L_2^{int} generated in the first iteration.

Why? The second decoder can do a better job than in the first iteration thanks to the availability of an improved (less noisy) L_1^{int}.

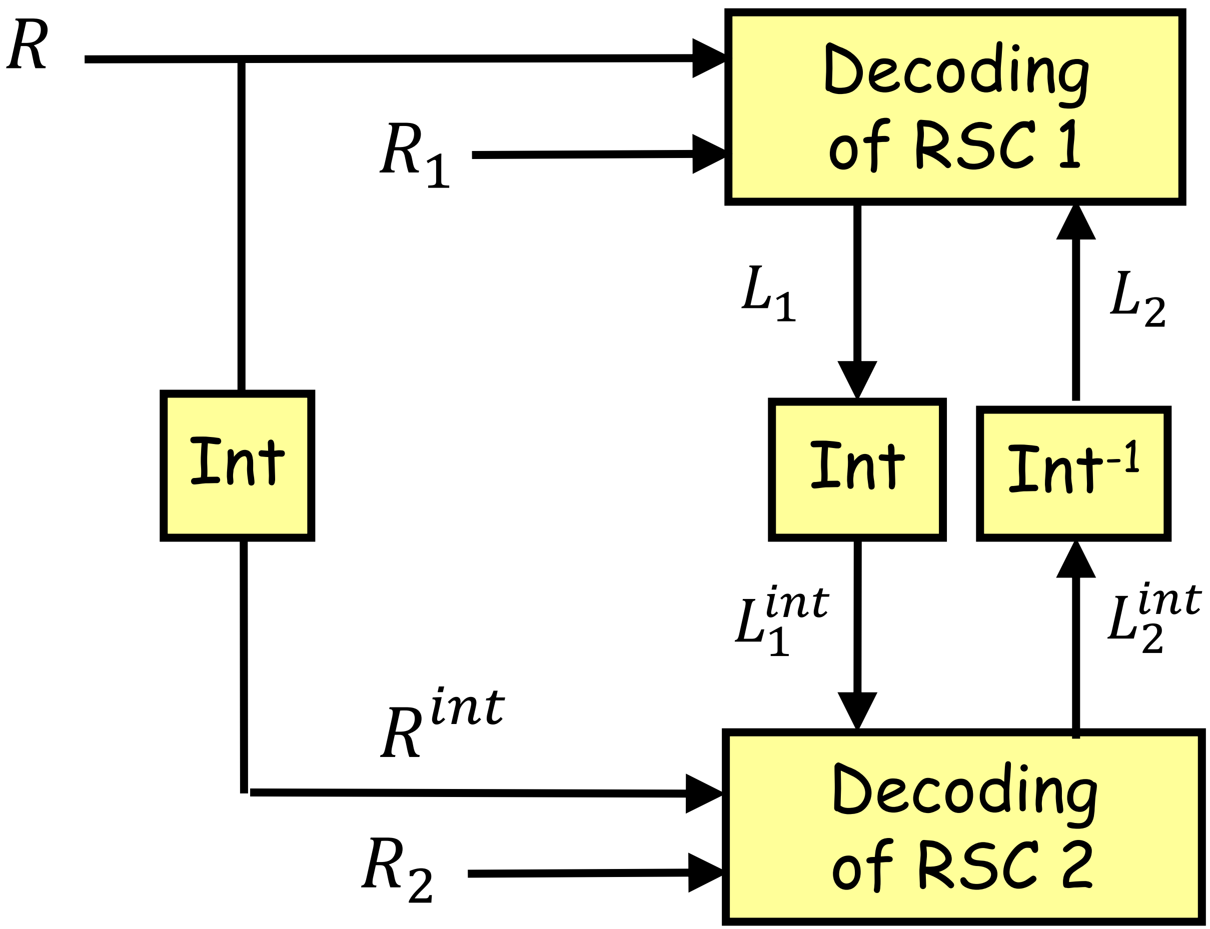


Second decoding iteration

Using the channel samples and the extrinsic info L_1^{int}, the second decoder generates a new estimate, L_2^{int}, of the interleaved message M^{int}.

L_2^{int} is de-interleaved and sent to the first decoder.

How does the iterative decoding algorithm work in practice?



We can perform as many iterations as we want.

As the number of iterations increase, the extrinsic info L_1 and L_2 becomes more and more reliable (less and less noisy).

Once we are satisfied with the reliability of L_1 and L_2, we can stop the iterative decoding.

Decoding of RSC codes

We are now going to study how to decode each RSC code.

Each RSC code is decoded using the BCJR algorithm. But, because the BCJR is rather complex to explain in the context of this module, we are going to focus instead on the more general MAP algorithm.

The MAP algorithm is simpler than the BCJR to explain, and the latter is only a particular case of it.

The BCJR is a version of the MAP that is applicable to trellis codes only. The MAP is more general in the sense that it can be applied to any type of code.

MAP algorithm

The MAP algorithm is a SISO decoding algorithm that is considered as optimal because it minimizes the bit error probability at the decoder output.

Consider, without loss of generality, the decoding of the first RSC decoder inside the turbo decoder.

The job of the MAP decoder is to produce the extrinsic info $L_1 = \left(l_{0,1}, l_{1,1}, l_{2,1}, \dots, l_{k-1,1}\right)$ by using the channel samples $R = (r_0, r_1, r_2, \dots, r_{k-1})$ and $R_1 = \left(r_{0,1}, r_{1,1}, r_{2,1}, \dots, r_{k-1,1}\right)$ as well as the extrinsic info $L_2 = \left(l_{0,2}, l_{1,2}, l_{2,2}, \dots, l_{k-1,2}\right)$ generated by the second RSC decoder in the previous decoding iteration.

MAP algorithm

Remember that all the samples fed to the RSC decoder follow a Gaussian distribution and can be written as

$r_i = m_i + n_i,$
$r_{i,1} = c_{i,1} + n_{i,1},$
$l_{i,2} = m_i + w_{i,2},$
with $i \in \{0, \dots, k-1\}$ and $m_i, c_{i,1}, \in \{-1, +1\}$.

The RSC decoder computes an estimate $z_{i,1}$ of each info bit m_i using the generic expression

$$z_{i,1} = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\Pr\{m_i{=}{+}1; r_j; r_{j,1}; L_{j,2}, j{=}0{\to}k{-}1\}}{\Pr\{m_i{=}{-}1; r_j; r_{j,1}; L_{j,2}, j{=}0{\to}k{-}1\}}\right),$$

where σ^2 is the variance of Gaussian noise over the channel.

MAP algorithm

The quantity $z_{i,1}$ is referred to as log-likelihood ratio (LLR). It is NOT the extrinsic info mentioned above, but will be used to compute it.

We can show that

$$z_{i,1} = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\sum_{C \in S_{i,+1}} \Pr\{C; r_j; r_{j,1}; L_{j,2}, j=0 \to k-1\}}{\sum_{C \in S_{i,-1}} \Pr\{C; r_j; r_{j,1}; L_{j,2}, j=0 \to k-1\}}\right),$$

where

$S_{i,+1}$ denotes the set of all codewords associated with the messages in which $m_i = +1$,

$S_{i,-1}$ is the set of all codewords associated with the messages in which $m_i = -1$.

A codeword is defined as a vector composed of a message M and its corresponding parity sequence P_1 .

MAP algorithm in turbo decoding

In order to lighten the expressions, we adopt the following notation for the LLRs:

$$z_{i,1} = \frac{\sigma^2}{2} \cdot \ln\left(\frac{A_{+1}}{A_{-1}}\right),$$

where $A_p = \sum_{C \in S_{i,p}} \Pr\left\{C;\ r_j;\ r_{j,1};\ L_{j,2}, j = 0 \to k-1\right\}$, $p \in \{+1, -1\}$.

We can show that the term A_p can be written as $A_p = \sum_{C \in S_{i,p}} \prod_{j=0}^{k-1} p\left(r_j \middle| m_j\right) \cdot p\left(r_{j,1} \middle| c_{j,1}\right) \cdot p\left(L_{j,2} \middle| m_j\right)$, which is equivalent to $A_p =$ $\sum_{C \in S_{i,p}} \exp\left(-\sum_{j=0}^{k-1}\left[\frac{\left(r_j - m_j\right)^2 + \left(r_{j,1} - c_{j,1}\right)^2}{2\sigma^2} + \frac{\left(l_{j,2} - m_j\right)^2}{2\sigma_{2,ext}^2}\right]\right)$.

MAP algorithm

After simplification of this equation, the LLR for each message bit m_i, $i \in \{0, \dots, k-1\}$, can be computed using the following expression:

$$z_{i,1} = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\sum_{C \in S_{i,+1}} \exp\left(\frac{1}{\sigma^2} \sum_{j=0}^{k-1} \left[\left(r_j + \frac{\sigma^2}{\sigma_{2,ext}^2} l_{j,2} \right) \cdot m_j + r_{j,1} \cdot c_{j,1} \right] \right)}{\sum_{C \in S_{i,-1}} \exp\left(\frac{1}{\sigma^2} \sum_{j=0}^{k-1} \left[\left(r_j + \frac{\sigma^2}{\sigma_{2,ext}^2} l_{j,2} \right) \cdot m_j + r_{j,1} \cdot c_{j,1} \right] \right)} \right).$$

This equation indicates that the MAP algorithm is simple from a theoretical viewpoint, but its practical implementation is far too complex in most cases because there are 2^{k-1} codewords in each set $S_{i,p}, p \in \{+1, -1\}$.

MAP algorithm

We can show that the LLR $z_{i,1}$ can also be written as

$$z_{i,1} = r_i + \frac{\sigma^2}{\sigma_{2,ext}^2} l_{i,2} + l_{i,1},$$

where $l_{i,1}$ is the LLR computed by replacing the actual channel sample r_i and the extrinsic info $l_{i,2}$ by neutral values ($r_i = l_{i,2} = 0$) as if the knowledge of these estimates was not available.

The two first terms r_i and $\frac{\sigma^2}{\sigma_{2,ext}^2} l_{i,2}$ were already known before the MAP decoding started. This means that the true contribution of the first decoder to the knowledge of the message bit m_i is represented by the third term $l_{i,1}$.

MAP algorithm

This third term $l_{i,1}$ is in fact the extrinsic info produced by the first decoder and represents the contribution of this code to the knowledge of the info bit m_i.

This is the only info that must be forwarded to the second decoder.

We must NOT send to the second decoder the complete LLR $z_{i,1}$ because the latter includes the components r_i and $\frac{\sigma^2}{\sigma^2_{2,ext}} l_{i,2}$.

MAP algorithm

The channel sample r_i does not need to be sent to the second decoder because the latter receives it directly from the channel (in interleaved form).

There is indeed no need to send twice the same info to the second decoder.

The extrinsic info sample $l_{i,2}$ does not need to be sent to the second decoder either because it is the latter which produced $l_{i,2}$ in the previous iteration.

There is indeed no need to send to the second decoder an info that it has previously generated.

MAP algorithm

To conclude, the first decoder generates the LLRs $z_{i,1}$, $i \in \{0, \dots, k-1\}$.

Each extrinsic info sample $l_{i,1}$, $i \in \{0, \dots, k-1\}$, is then produced by subtracting the already available samples r_i and $\frac{\sigma^2}{\sigma_{2,ext}^2} l_{i,2}$ from it:

$$l_{i,1} = z_{i,1} - r_i - \frac{\sigma^2}{\sigma_{2,ext}^2} l_{i,2}.$$

The vector of samples $l_{i,1}$, $i \in \{0, \dots, k-1\}$, is used by the second decoder in conjunction with the channel estimates r_i (in interleaved form) and $r_{i,2}$ to produce updated extrinsic info samples $l_{i,2}$ (in interleaved form).

Iterative decoding

In a turbo decoder, all RSC decoders use the same SISO algorithm: the BCJR which is an adaptation to convolutional codes of the generic MAP algorithm.

Each first RSC decoder computes the LLR for each message bit m_i, $i \in \{0, \dots, k-1\}$, using

$$z_{i,1} = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\sum_{C \in S_{i,+1}} \exp\left(\sum_{j=0}^{k-1}\left[\frac{r_j' \cdot m_j + r_{j,1} \cdot c_{j,1}}{\sigma^2}\right]\right)}{\sum_{C \in S_{i,-1}} \exp\left(\sum_{j=0}^{k-1}\left[\frac{r_j' \cdot m_j + r_{j,1} \cdot c_{j,1}}{\sigma^2}\right]\right)}\right),$$

with $r_j' = r_j$ in the first iteration and $r_j' = r_j + \frac{\sigma^2}{\sigma_{2,ext}^2} l_{j,2}$ starting in the second iteration.

Iterative decoding

Each second RSC decoder computes the LLR for each message bit m_i^{int}, $i \in \{0, \dots, k-1\}$, using

$$z_{i,2}^{int} = \frac{\sigma^2}{2} \cdot \ln\left(\frac{\sum_{C \in S_{i,+1}} \exp\left(\sum_{j=0}^{k-1}\left[\frac{r_j^{int\prime} \cdot m_j^{int} + r_{j,2} \cdot c_{j,2}}{\sigma^2}\right]\right)}{\sum_{C \in S_{i,-1}} \exp\left(\sum_{j=0}^{k-1}\left[\frac{r_j^{int\prime} \cdot m_j^{int} + r_{j,2} \cdot c_{j,2}}{\sigma^2}\right]\right)}\right),$$

with $r_j^{int\prime} = r_j^{int} + \frac{\sigma^2}{\sigma_{1,ext}^2} l_{j,1}^{int}$.

So, all RSC decoders are strictly identical as they rely on the same generic MAP expression to generate their LLRs.

Iterative decoding

We are now going to illustrate the benefit of using an iterative decoding algorithm where extrinsic info can be exchanged between the first and the second RSC decoder.

Without loss of generality, let us focus once again on the first RSC decoder.

At the first iteration, the extrinsic info samples $l_{j,2}$, $j \in \{0, \dots, k-1\}$, are not available yet. Hence, they cannot be fed to the the first RSC decoder. This is why we initially use $r_j' = r_j$ in the MAP expression.

Iterative decoding

At the first iteration, this first decoder implements the MAP algorithm by using the samples $r_j' = r_j$ and $r_{j,1}$, $j \in \{0, \dots, k-1\}$.

These estimates can be written in the form
$r_j' = r_j = m_j + n_j(0, \sigma^2)$,
$r_{j,1} = c_{j,1} + n_{j,1}(0, \sigma^2)$,
with $j \in \{0, \dots, k-1\}$ and $m_j \in \{-1, +1\}$.

The reliability of these samples can be assessed using their “SNR” γ. A higher SNR value corresponds to samples that are “less noisy”, i.e. more reliable.

Iterative decoding

Here, this SNR is defined as the ratio between the energy of the "signal" m_j, which is the square of $m_j \in \{-1, +1\}$, and the power of the Gaussian noise sample represented by its variance σ^2.

So we have $\gamma = \frac{1}{\sigma^2}$ for both samples r_j' and $r_{j,1}$ at the first iteration.

Starting in the second iteration, the first RSC decoder has access to the extrinsic info samples $l_{j,2}$ which can be expressed as
$l_{j,2} = m_j + w_{j,2}\left(0, \sigma_{2,ext}^2\right)$,
with $j \in \{0, \ldots, k-1\}$ and $m_j \in \{-1, +1\}$.

Iterative decoding

It is easy to show that the SNR of samples $l_{j,2}$ is given by $\gamma_{2,ext} = \frac{1}{\sigma_{2,ext}^2}$.

After the first iteration, the first RSC decoder is fed with samples
$r_j' = r_j + \frac{\sigma^2}{\sigma_{2,ext}^2} l_{j,2}$ and $r_{j,1}$, $j \in \{0, \dots, k-1\}$.

We have seen that the SNR of samples $r_{j,1}$ is given by $\gamma = \frac{1}{\sigma^2}$. This result is valid at any decoding iteration.

What about the SNR of samples $r_j' = r_j + \frac{\sigma^2}{\sigma_{2,ext}^2} l_{j,2}$?

Iterative decoding

We can write

$$r_j' = r_j + \frac{\sigma^2}{\sigma_{2,ext}^2} l_{j,2} = m_j + \frac{\sigma^2}{\sigma_{2,ext}^2} m_j + n_j(0, \sigma^2) + w_{j,2}\left(0, \sigma_{2,ext}^2\right), \ j \in \{0, \dots, k-1\}.$$

We assume that samples r_j and $l_{j,2}$ are independent. Therefore, samples r_j' follow a Gaussian distribution with a mean given by $m_j\left(1 + \frac{\sigma^2}{\sigma_{2,ext}^2}\right)$ and a variance expressed as $\sigma^2 + \sigma_{2,ext}^2\left(\frac{\sigma^2}{\sigma_{2,ext}^2}\right)^2$.

Iterative decoding

As a result, we can show that the SNR of samples r_j' is given by $\gamma' = \gamma + \gamma_{2,ext}$.

The SNR of samples r_j' is thus the sum of the SNR of the channel samples and the SNR of the extrinsic info.

This means that the availability of the extrinsic info does result in an increase in the SNR of samples r_j' which can become very significant.

Remember that, without extrinsic info, i.e. without iterative decoding, we only have $\gamma' = \gamma$.

Iterative decoding

As the number of iterations is increased, the SNR $\gamma_{2,ext}$ of the extrinsic info is also increased.

This means that the first RSC decoder is fed with samples r_j' that are more and more reliable (since $\gamma' = \gamma + \gamma_{2,ext}$).

As a result, this decoder is able to produce extrinsic info (to be used by the second RSC decoder) that becomes progressively more reliable as the number of iterations increases.

Iterative decoding

This is a virtuous cycle that allows the iterative decoding algorithm to eventually correct all transmission errors.

This typically happens when the SNR $\gamma' = \gamma + \gamma_{2,ext}$ has reached a value that is sufficiently high.

How does the iterative decoding end in practice?

Assume the last RSC decoding is done by the second RSC decoder, which is generally the case.

Iterative decoding

After a sufficient number of iterations, the LLRs $z_{i,2}^{int}$ generated by the second decoder have become very reliable estimates of the transmitted info bits m_i^{int}, $i \in \{0, \dots, k-1\}$.

These LLRs are then ready for the final step: they are de-interleaved in order to produce the sequence of LLRs $z_{i,2}$ which is finally processed by a simple decision block as follows:

• If $z_{i,2} > 0$ then the turbo decoder selects $m_i' = +1 (\equiv 1)$ as the most likely value for the transmitted info bit.

Iterative decoding

• If $z_{i,2} < 0$ then the turbo decoder selects $m_i' = -1 (\equiv 0)$ as the most likely value for the transmitted info bit.

In practice, the iterative decoding algorithm that we have just described only works well if the channel SNR is greater than a certain threshold SNR, often called “convergence threshold”.

If the SNR is smaller than the convergence threshold, the iterative algorithm does not converge, meaning that the bit error probability does not fall from one iteration to the next.

Iterative decoding

The iterative decoding algorithm is almost as good as ML decoding.

Nowadays, iterative algorithms are commonly applied in all kinds of communication receivers (turbo equalization, turbo demodulation, ...).

The "message passing"/"belief propagation" algorithm, according to Claude Berrou:

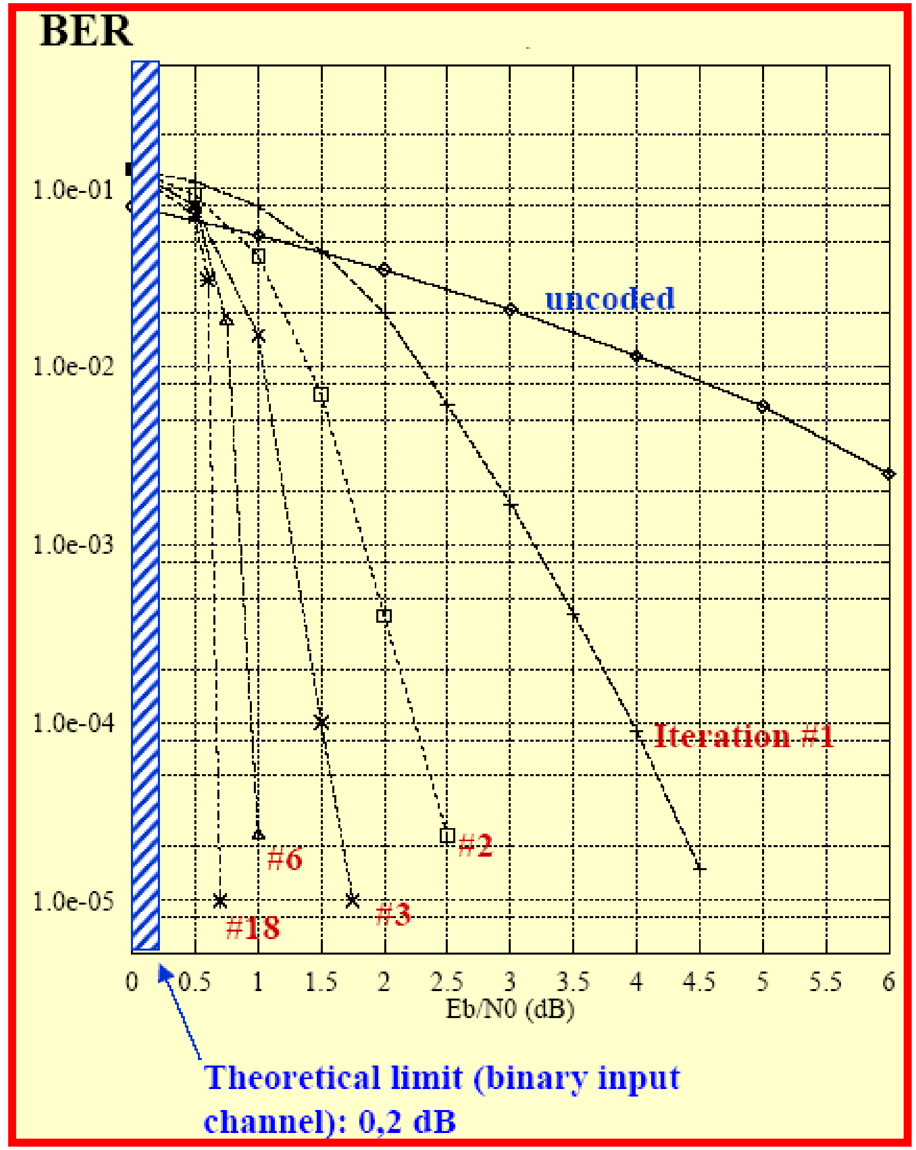
"When several probabilistic machines work together on the estimation of a common set of symbols, all the machines must ultimately give the same decision about each symbol, as a single (global) decoder would."

Turbo codes

Performance of a rate-1/2 turbo code over BPSK, AWGN channel.

$P_{eb} = 10^{-5}$ is achieved for $E_b/N_0 = 0.7$ dB → This turbo code performs only 0.5 dB away from the channel capacity limit !

This level of performance is not achievable by using any other coding technique invented before 1993.

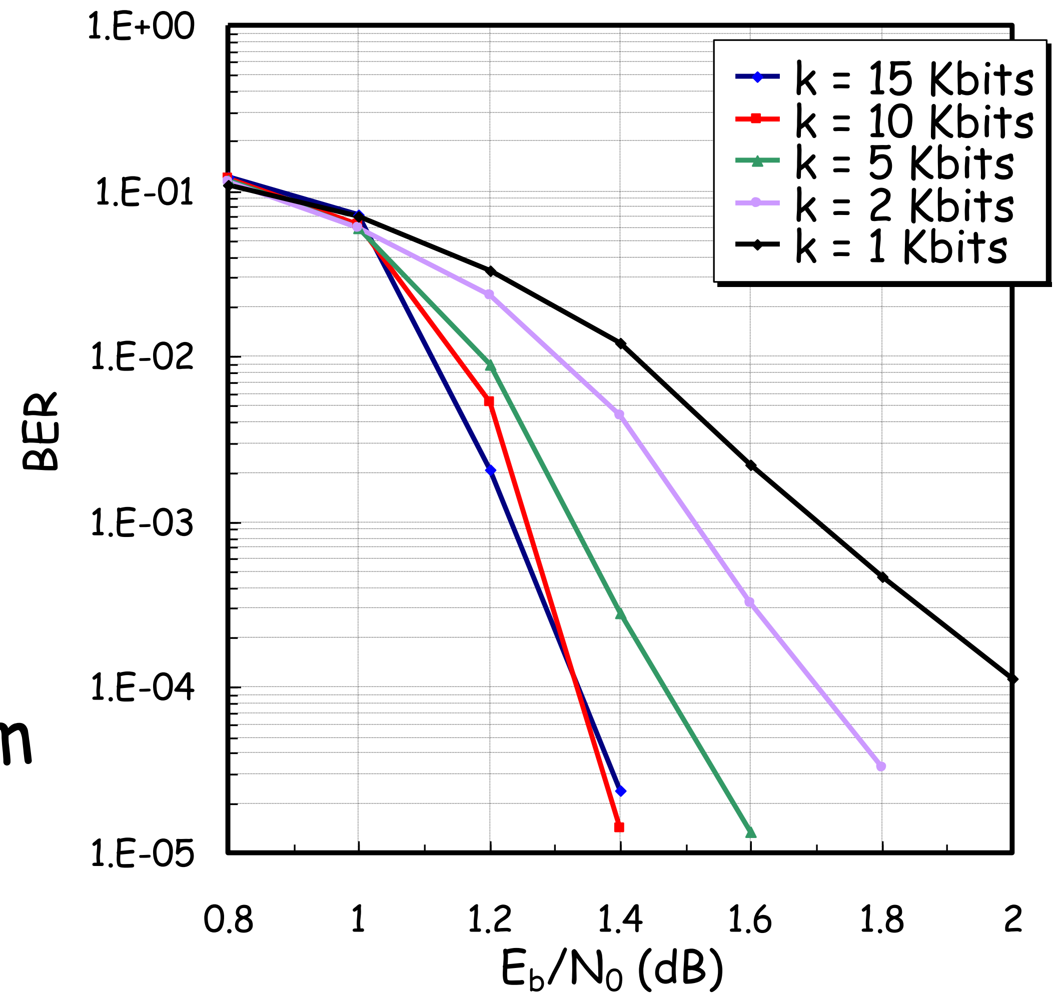


Turbo codes

BER vs. E_b/N_0 curves for a turbo code obtained by parallel concatenation of two rate-2/3, K = 5, (23, 31) RSC codes.

k is the length of a message M. It is also the size of the random interleaver.

R_c = 1/2, BPSK, AWGN channel, random interleaving, 10 iterations, max-log-MAP decoding



Performance of a rate-1/2 turbo code (Max-log-MAP decoding, 10 iterations, k = 50 Kbits, AWGN, BPSK channel)