Information Theory and Channel Coding

Prof. Rodrigo C. de Lamare
CETUC, PUC-Rio, Brazil
delamare@cetuc.puc-rio.br

X. Turbo codes

A. Introduction

B. Encoding and code structure

C. Decoding

D. MAP algorithm

A. Introduction

- Coding techniques until the 1990s were heavily based on the use of algebraic structures for BCH and Reed-Solomon (RS) codes.

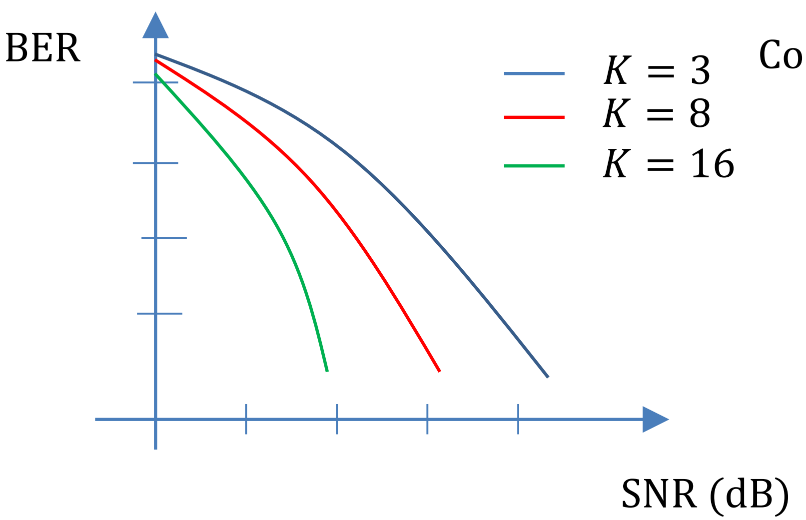
- Additionally, designers relied on intensive use of memory and shift registers for powerful convolutional codes.

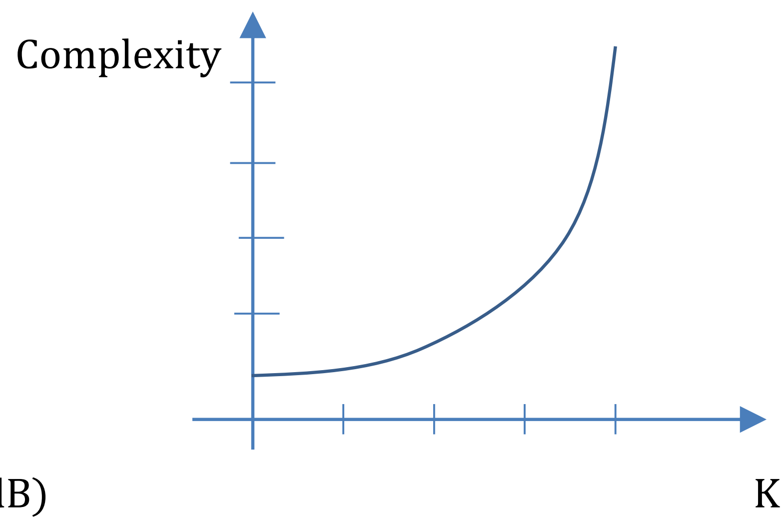
- Concatenated coding strategies involving BCH and RS along convolutional codes and interleavears have also been considered.

- Most of the above mentioned approaches would require long codes (n large) and/or large constraint lengths (K large) to approach capacity.

- The trouble with increasing n and/or K is the exponential computational complexity for decoding.

- For convolutional codes, we could illustrate this trade-off as follows:

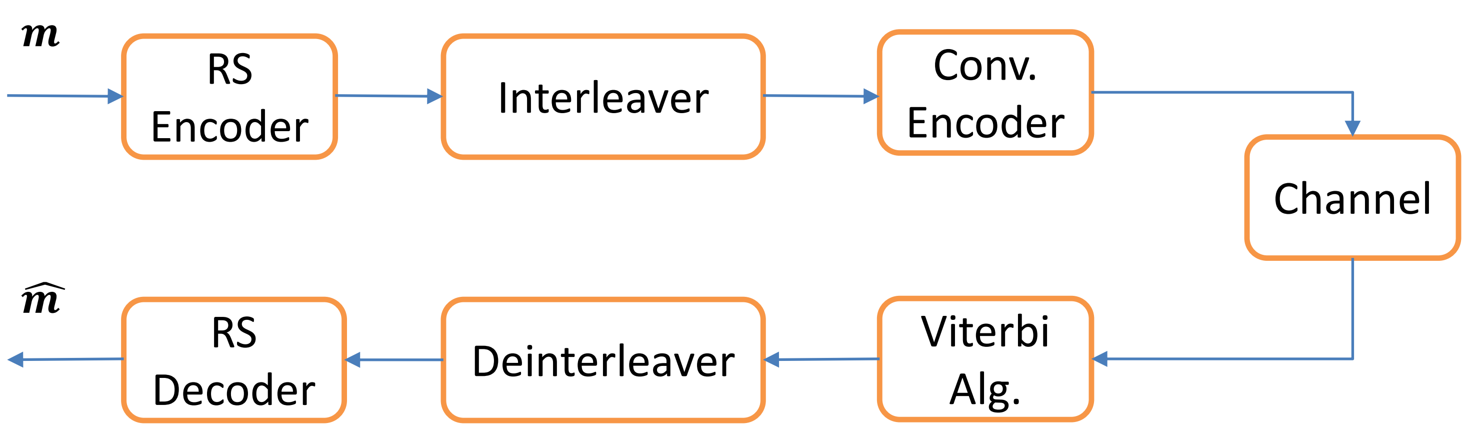




- Concatenated codes with interleavers have been introduced back in 1965 by Forney and could in principle limit the growth in complexity.

- The basic idea of concatenated codes relies on splitting the decoding into several simpler tasks.

- Concatenated codes based on RS and convolutional codes were very successful for satellite and space communications with the structure:



- Turbo codes are linear concatenated codes that were invented by Claude Berrou, Alan Glavieux and Punya Thitimajshima in 1993.

- Turbo codes can approach Shannon's theoretical limit by using concatenated convolutional codes with interleavers and iterative MAP decoding.

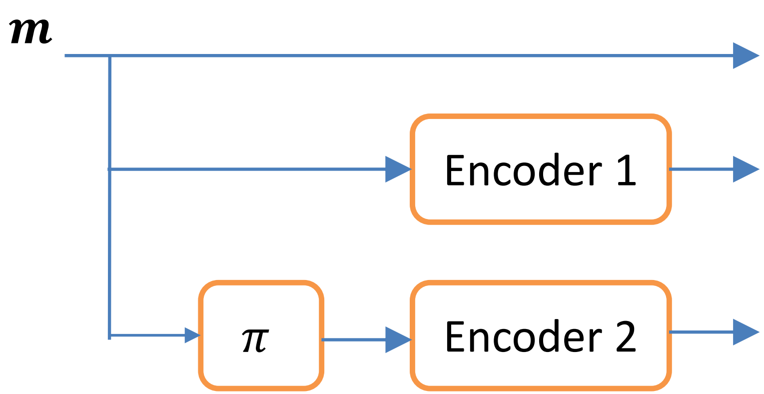
- The basic idea consists of designing a code that is a concatenation of a convolutional code and another convolutional code with interleaved input.

- Decoding for such "random"-like concatenated code with sufficient structure is carried out by 2 MAP decoders that exchange information.

Near optimum error correcting coding and decoding: Turbo-codes. C Berrou, A Glavieux. IEEE Transactions on communications 44 (10), 1261-1271, 1996

B. Encoding and code structure

- Let us now describe the encoding procedure of the original turbo codes, which employ a parallel concatenation structure.



$\boldsymbol{m} = [m_0 \; m_1 \; \dots \; m_{k-1}]$ Systematic bits

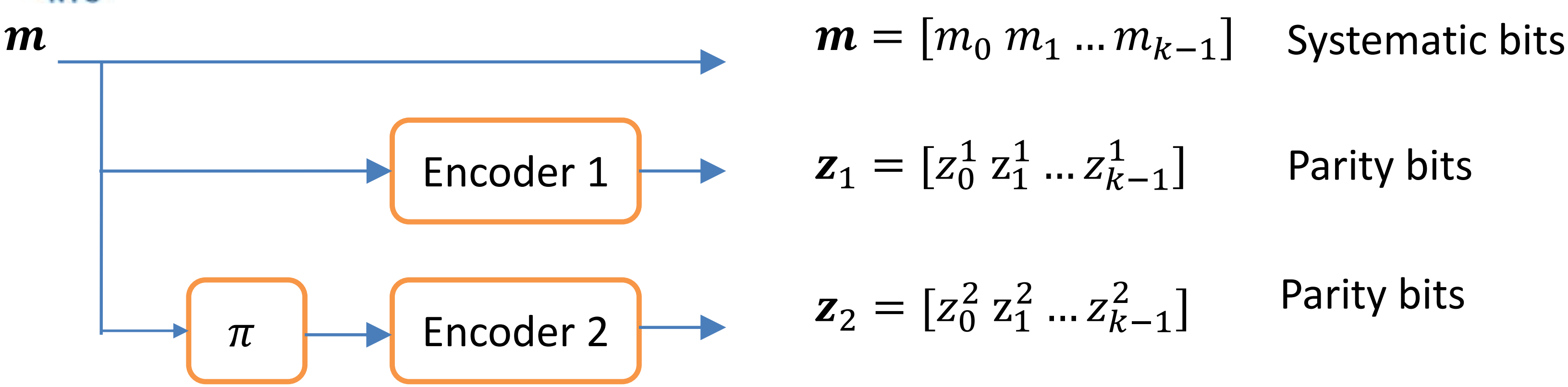
$\boldsymbol{z}_1 = [z_0^1 \; z_1^1 \; \dots \; z_{k-1}^1]$ Parity bits

$\boldsymbol{z}_2 = [z_0^2 \; z_1^2 \; \dots \; z_{k-1}^2]$ Parity bits

- The codeword is described by

$$\begin{aligned} \boldsymbol{c} &= [c_o \; c_1 \; \dots \; c_{n-1}] \\ &= [m_o \; z_0^1 \; z_0^2 \mid m_1 \; z_1^1 \; z_1^2 \mid \dots \mid m_{k-1} \; z_{k-1}^1 \; z_{k-1}^2], \end{aligned}$$

where the code rate is $R = \frac{1}{3}$.



- Recursive convolutional codes are often used as constituent codes.
- The constituent codes can be block codes.
- The concatenation can be serial.
- More than 2 encoders can be employed, resulting in turbo codes with lower code rates.

- Let us now assume transmission of the codewords produced by a turbo encoder using a parallel concatenated scheme.

- Assume that the channel is AWGN, which results in

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{n}, \qquad \mathbb{R}^{1\times n},$$

where $\boldsymbol{n} = [n_0\ n_1\ \ldots n_{k-1}]$ is the vector with noise samples.

- The received data of the turbo coding system can be written as

$$\boldsymbol{r} = [r_0\, r_1\ \dots\ r_{n-1}] = \boldsymbol{c} + \boldsymbol{n}$$
$$= [u_0\ \xi_0^1\ \xi_0^2\ |u_1\, \xi_1^1\, \xi_1^2|\ \dots\ |\ u_{n-1}\ \xi_{k-1}^1\, \xi_{k-1}^2],$$

where $\boldsymbol{u} = [u_0\, u_1\ \dots\ u_{k-1}] = \boldsymbol{m} + \boldsymbol{n}_m$ is the vector of noisy systematic bits,

$\boldsymbol{\xi}^1 = [\xi_0^1\, \xi_1^1\ \dots\ \xi_{k-1}^1] = \boldsymbol{z}_1 + \boldsymbol{n}_{z_1}$ is the vector of noisy parity bits of encoder 1,

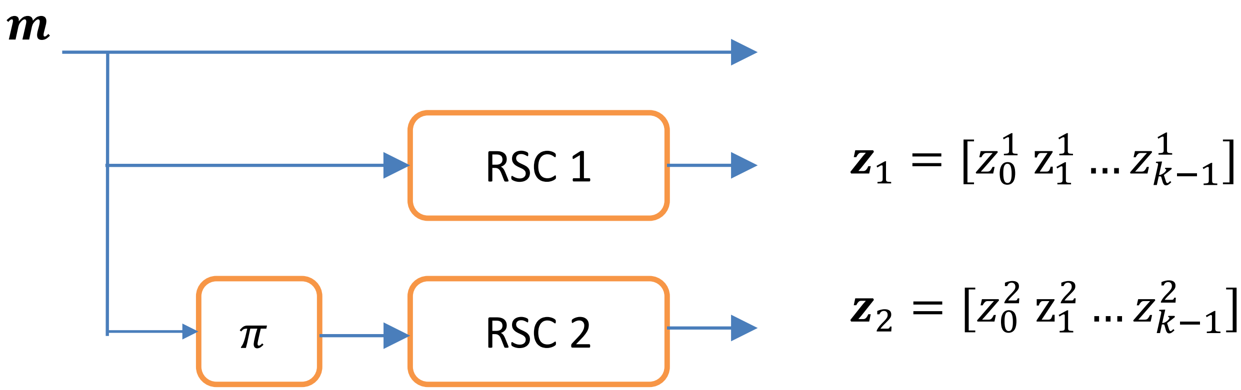
$\boldsymbol{\xi}^2 = [\xi_0^2\, \xi_1^2\ \dots\ \xi_{k-1}^2] = \boldsymbol{z}_2 + \boldsymbol{n}_{z_2}$ is the vector of noisy parity bits of encoder 2,

and $\boldsymbol{n} = [n_0^m\, n_0^{z_1}\, n_0^{z_2}\ |n_1^m\, n_1^{z_1}\, n_1^{z_2}\ |\dots|n_{k-1}^m\, n_{k-1}^{z_1}\, n_{k-1}^{z_2}\ ]$ is the noise vector.

Example 1

Consider a turbo encoder with parallel concatenation whose constituent recursive systematic convolutional code (RSC) is given by

$$G(D) = \frac{1}{1+D^2}$$

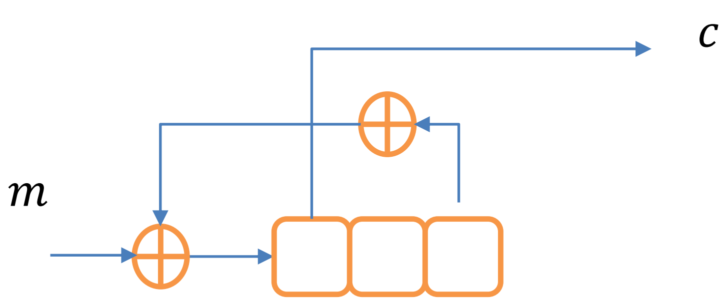


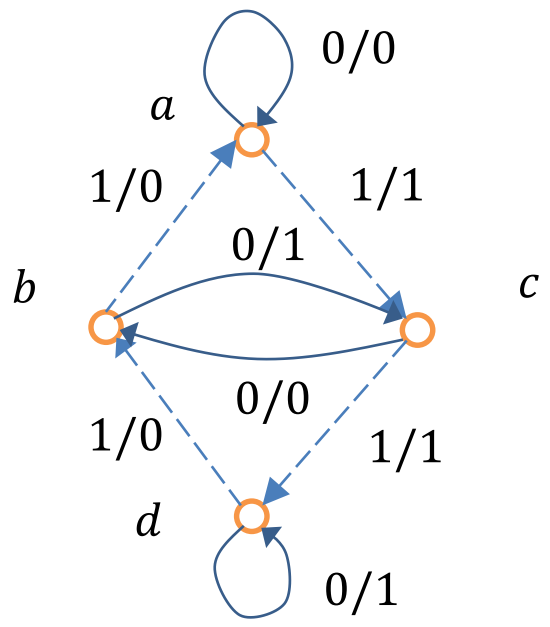
The turbo scheme is to encode the message $\boldsymbol{m} = [1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1]$ using the interleaver $\pi = \{8, 3, 7, 6, 9, 0, 2, 5, 1, 4\}$.

a) Obtain the encoder and the state diagram of the RSC.
b) Compute the parity bits of the RSCs and the interleaved message.
c) Determine the codeword $\mathbf{c}$.
d) Assume that the system uses puncturing to increase the code rate to $\frac{1}{2}$ and compute the punctured codeword.

Solution:

a) The encoder and the state diagram are given by





b) The parity bits of the first RSC with $\boldsymbol{m} = [1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1]$ is given by

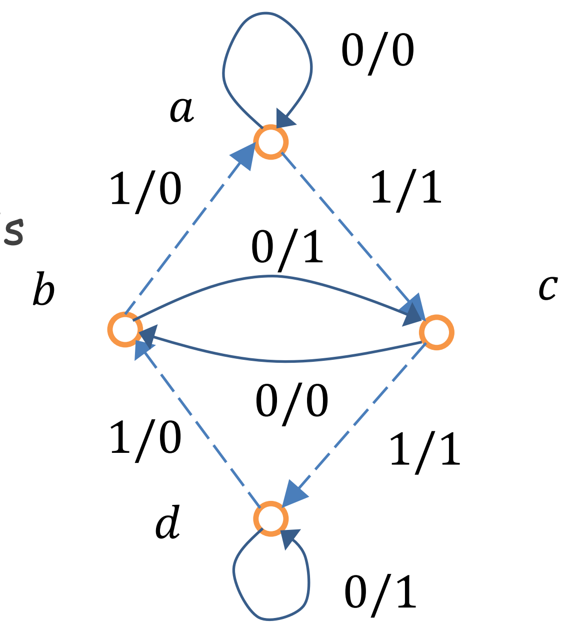
$$\mathbf{z}_1 = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ ]$$

The interleaved message with $\pi = \{8, 3, 7, 6, 9, 0, 2, 5, 1, 4\}$ is

$$\boldsymbol{m}' = [1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1]$$

and the parity bits of the second RSC are

$$\mathbf{z}_2 = [1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ ]$$



c) The codeword produced by the turbo encoder is then

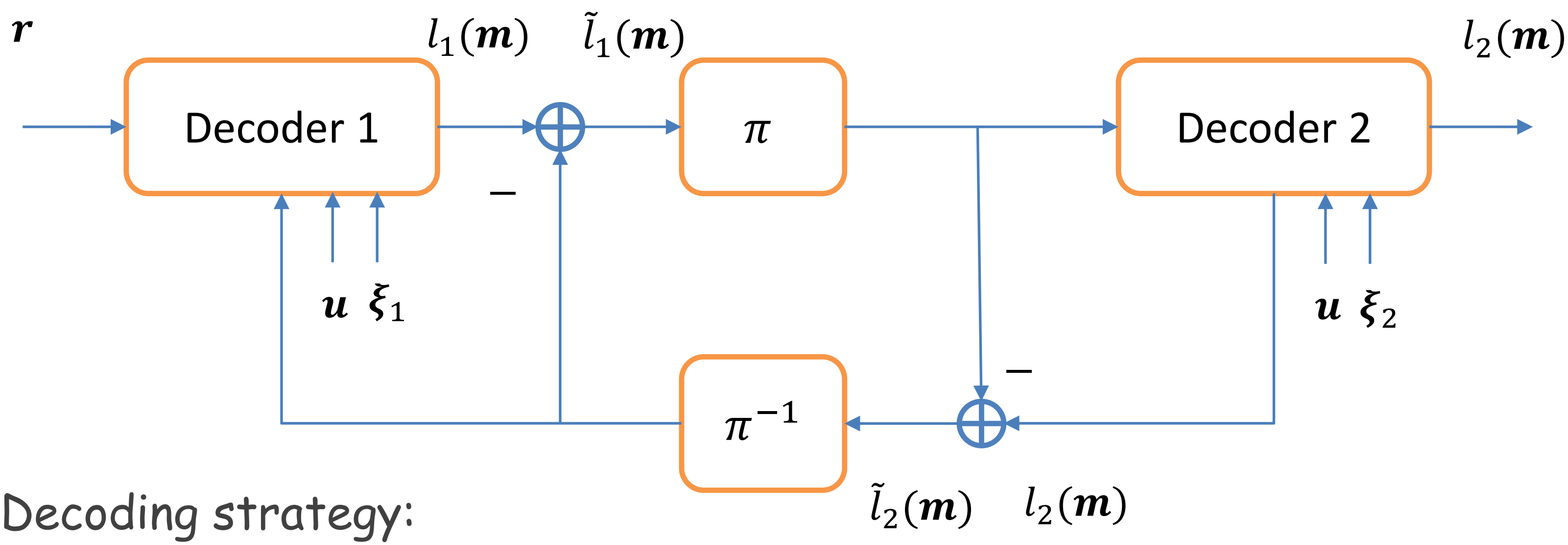
$$\boldsymbol{c} = [c_o\ c_1\ \dots c_{n-1}] = [m_o\ z_0^1\ z_0^2 |\ m_1\ z_1^1\ z_1^2 |\ \dots |m_{k-1}\ z_{k-1}^1\ z_{k-1}^2],$$
$$= [1\ 1\ 1\ |\ 1\ 1\ 0\ |\ 0\ 1\ 1 |\ 0\ 1\ 1 |\ 1\ 0\ 0\ |\ 0\ 1\ 0\ |\ 1\ 1\ 0\ |\ 01\ 0\ |\ 10\ 1 |\ 1\ 1\ 1]$$

where the code rate is $R = \frac{1}{3}$.

d) In order to increase the code rate to $\frac{1}{2}$ we puncture the output of the RSC encoders in an alternating fashion, which yields

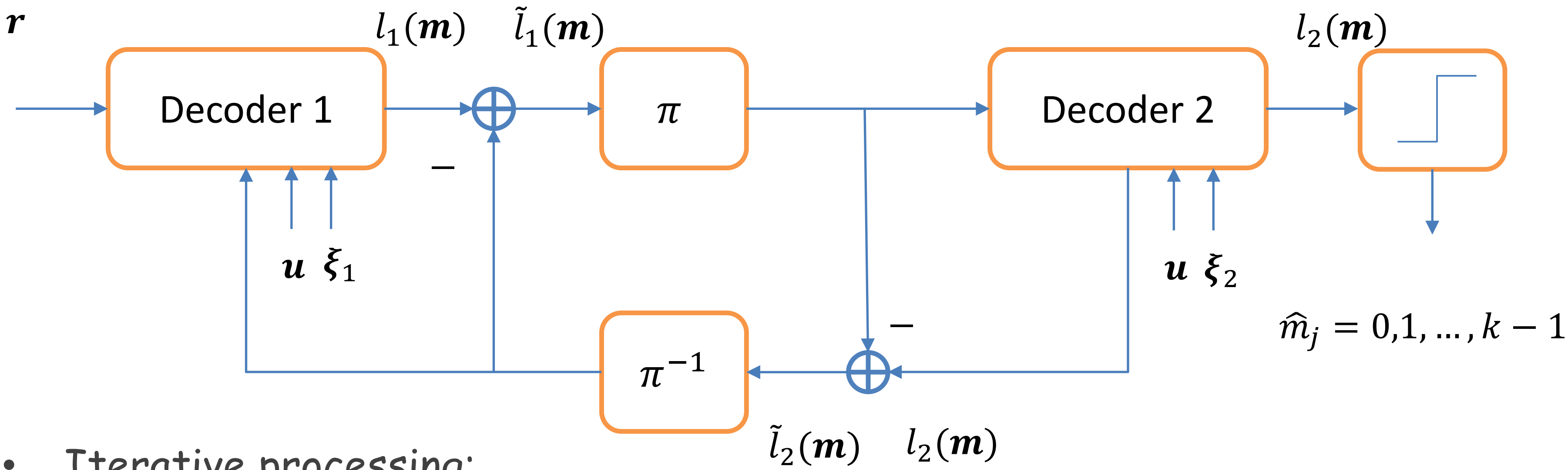
$$\boldsymbol{c} = [c_o\ c_1\ \dots c_{2k-1}] = [m_o\ z_0^1 \mid m_1\ z_1^2 \mid \dots \mid m_{k-1}\ z_{k-1}^2]$$
$$= [1\ 1 \mid 1\ 0 \mid 0\ 1 \mid 0\ 1 \mid 1\ 0 \mid 0\ 0 \mid 1\ 1 \mid 0\ 0 \mid 10 \mid 1\ 1]$$

B. Turbo decoding



- Decoding strategy:
  - Two MAP algorithms are used to calculate for each bit $m_j,\ j = 0,1,\dots,k-1$ if the bit probability is either $+1$ or -1.
  - MAP decision rule:

$$\frac{P(m_j = +1|\boldsymbol{r})}{P(m_j = -1|\boldsymbol{r})} \underset{-1}{\overset{+1}{\gtrless}} 1$$



- Iterative processing:
  - The second decoder makes the decision as follows:

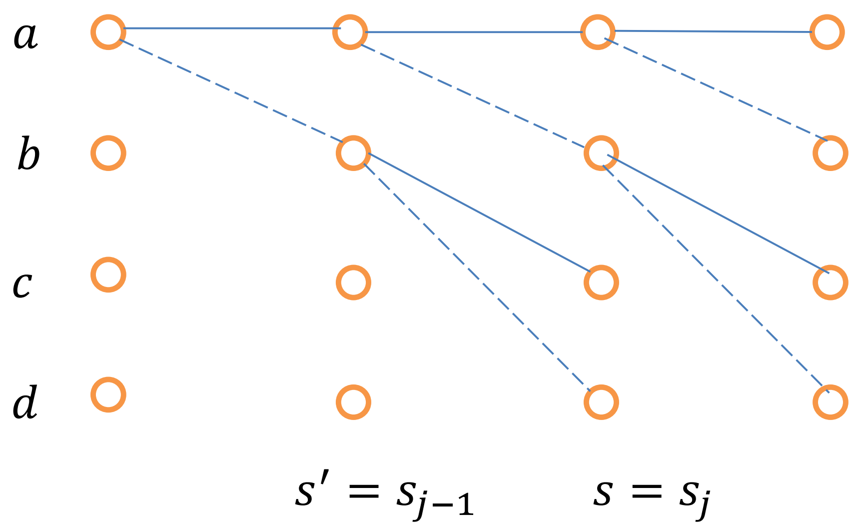
$$\hat{m}_j = \operatorname{sgn}\left\{\log\left[\frac{P(m_j = +1|\boldsymbol{r})}{P(m_j = -1|\boldsymbol{r})}\right]\right\} = \operatorname{sgn}\{l(m_j)\}$$

  - Key problem: to compute the a posteriori log-likelihood ratio (LLR):

$$l(m_j) = l(m_j|\boldsymbol{r}) = \log\left[\frac{P(m_j = +1|\boldsymbol{r})}{P(m_j = -1|\boldsymbol{r})}\right]$$

Decoding process

- Consider the previous turbo decoding scheme and the trellis associated to the recursive convolutional code.



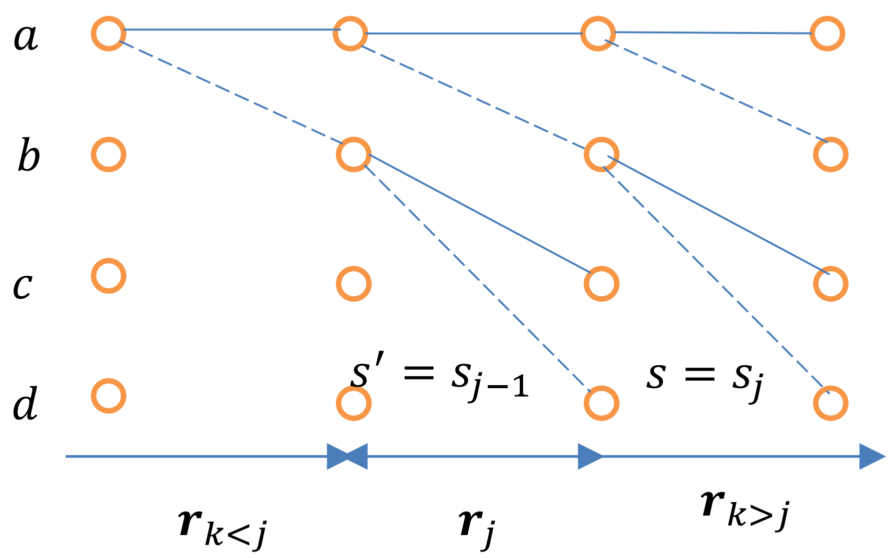
- The LLR can be expressed as a transition between the previous state $s_{j-1} = s'$ and the current state $s_j = s$:

$$l(m_j) = l(m_j|\boldsymbol{r}) = \log\left[\frac{P(m_j = +1|\boldsymbol{r})}{P(m_j = -1|\boldsymbol{r})}\right]$$

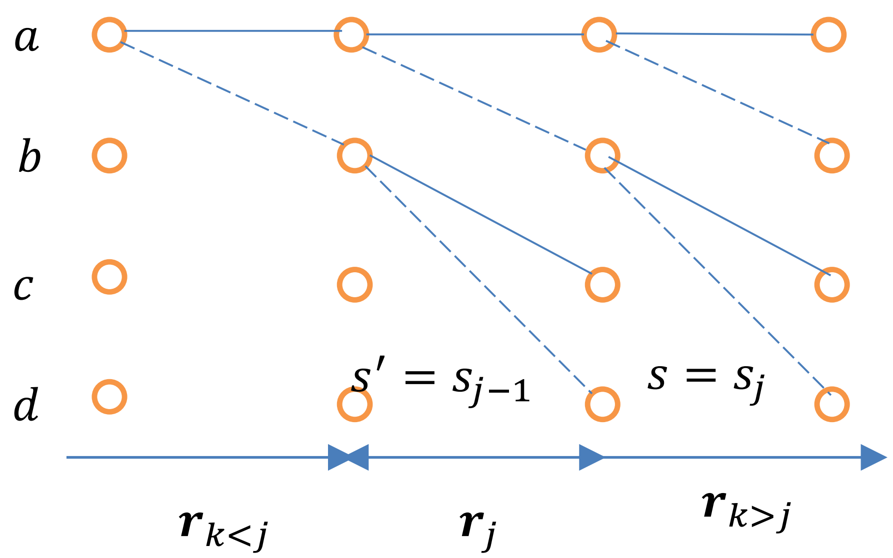
$$= \log\left[\frac{\sum_{(s',s)\to m_j=+1} p(s_{j-1}=s', s_j=s, \boldsymbol{r})}{\sum_{(s',s)\to m_j=-1} p(s_{j-1}=s', s_j=s, \boldsymbol{r})}\right]$$

- The LLR expression $l(m_j)$ can be rewritten because the states $s_{j-1} = s'$ and $s_j = s$ are assumed known.

- Therefore, we can determine the bit m_j that triggers the transition between $s_{j-1} = s'$ and $s_j = s$.



- Consider the joint probability density function $p(s_{j-1} = s', s_j = s, \boldsymbol{r})$ in $l(m_j)$ then the received signal $\boldsymbol{r}$ can be split into 3 parts:
  - $\boldsymbol{r}_{k<j}$ - sequence associated to the previous sequence.
  - $\boldsymbol{r}_j$ - sequence associated to the current transition.
  - $\boldsymbol{r}_{k>j}$ - sequence associated to the posterior sequence.



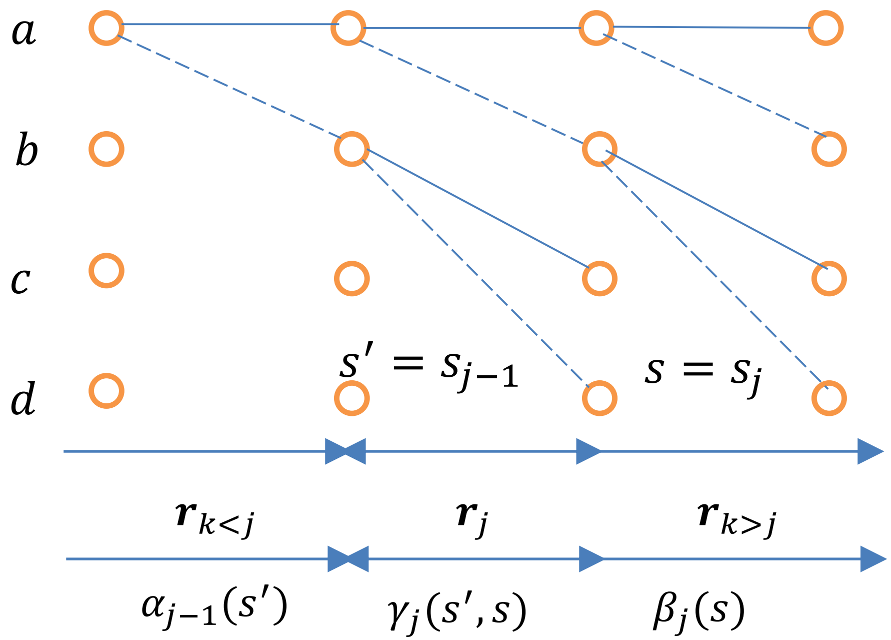
- We can rewrite the joint prob. density function $p(s_{j-1} = s', s_j = s, \boldsymbol{r})$ using Bayes' rule and the fact that the channel is memoryless as

$$
\begin{aligned}
p(s_{j-1} = s', s_j = s, \boldsymbol{r}) &= p(s_{j-1} = s', s_j = s, \boldsymbol{r}_{k<j}, \boldsymbol{r}_j, \boldsymbol{r}_{k>j}) \\
&= p(s', s, \boldsymbol{r}) \\
&= p(\boldsymbol{r}_{k>j}|s)p(s', s, \boldsymbol{r}_j, \boldsymbol{r}_{k>j}) \\
&= \underbrace{p(\boldsymbol{r}_{k>j}|s)}_{\beta_j(s)}\underbrace{p(\boldsymbol{r}_j, s|s')}_{\gamma_j(s',s)}\underbrace{p(s', \boldsymbol{r}_{k<j})}_{\alpha_{j-1}(s')} \\
&= \alpha_{j-1}(s')\gamma_j(s', s)\beta_j(s),
\end{aligned}
$$

where $\alpha_{j-1}(s')$, $\gamma_j(s', s)$ and $\beta_j(s)$ are the forward, branch and backward metrics, respectively.

- With these metrics we can replace the joint prob. density function.

- The forward, branch and backward metrics $\alpha_{j-1}(s')$, $\gamma_j(s',s)$ and $\beta_j(s)$ are depicted in the trellis below.



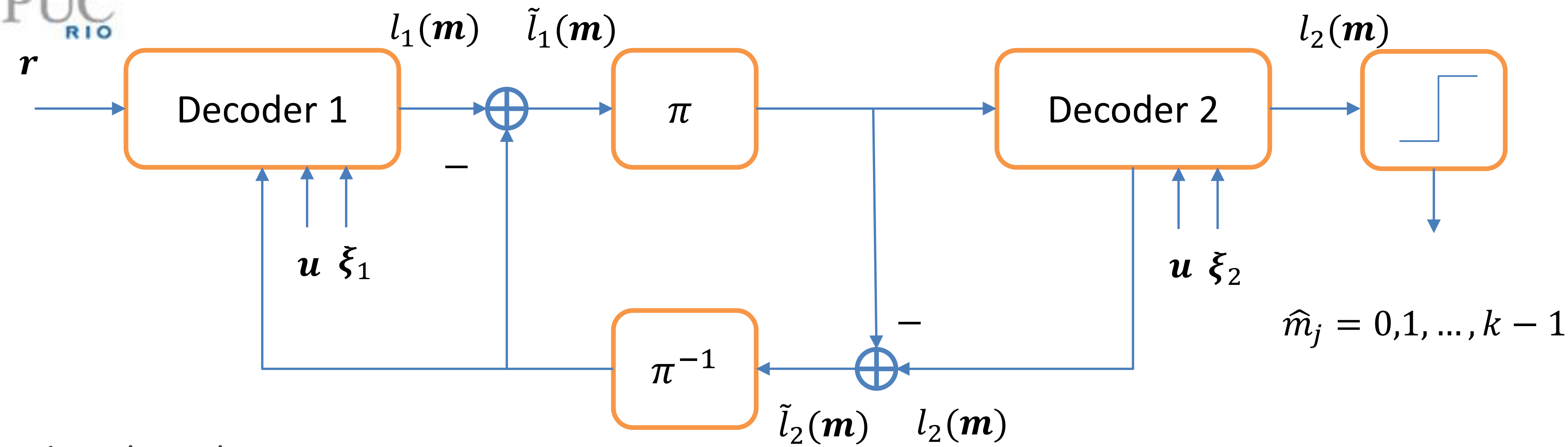
- These metrics contain information about the trellis that can be used to compute the LLR.

- The a posteriori LLR can then be rewritten as

$$
\begin{aligned}
l(m_j|\boldsymbol{r}) &= \log\left[\frac{\sum_{(s',s)\to m_j=+1} p(s_{j-1}=s', s_j=s, \boldsymbol{r})}{\sum_{(s',s)\to m_j=-1} p(s_{j-1}=s', s_j=s, \boldsymbol{r})}\right] \\
&= \log\left[\frac{\sum_{(s',s)\to m_j=+1} \alpha_{j-1}(s')\gamma_j(s',s)\beta_j(s)}{\sum_{(s',s)\to m_j=-1} \alpha_{j-1}(s')\gamma_j(s',s)\beta_j(s)}\right] \\
&= \log\left[\frac{P(m_j=+1)}{P(m_j=-1)}\right] + L_c r_j + \log\left[\frac{\sum_{(s',s)\to m_j=+1} \alpha_{j-1}(s')\chi_j(s',s)\beta_j(s)}{\sum_{(s',s)\to m_j=-1} \alpha_{j-1}(s')\chi_j(s',s)\beta_j(s)}\right] \\
&= \underbrace{l_a(m_j)}_{\text{a priori information}} + \underbrace{L_c}_{\text{channel reliability}} r_j + \underbrace{\log\left[\frac{\sum_{(s',s)\to m_j=+1} \alpha_{j-1}(s')\chi_j(s',s)\beta_j(s)}{\sum_{(s',s)\to m_j=-1} \alpha_{j-1}(s')\chi_j(s',s)\beta_j(s)}\right]}_{\text{extrinsic information}},
\end{aligned}
$$

where $L_c = \frac{4}{2\sigma^2}$ and $\chi_j(s', s) = e^{(\frac{L_c}{2}\sum_{l=j+1}^{n} r_l c_l)}$.

Turbo decoding algorithm



1st decoder:

- Computes $l_1(\boldsymbol{m})$ based on ξ_1 and $\boldsymbol{u}$.
- Obtains extrinsic information $\tilde{l}_1(\boldsymbol{m}) = l_1(\boldsymbol{m}) - \tilde{l}_2(\boldsymbol{m})$

2nd decoder:

- Computes $l_2(\boldsymbol{m})$ based on ξ_2 and $\boldsymbol{u}$.
- Obtains extrinsic information $\tilde{l}_2(\boldsymbol{m}) = l_2(\boldsymbol{m}) - \tilde{l}_1(\boldsymbol{m})$

Decision:

- $\hat{m}_j = \mathrm{sgn}\left\{l(m_j)\right\},\ j = 0,1,\ldots,k-1$

D. MAP algorithm

- The task of the MAP algorithm is to compute

$$l(m_j|\boldsymbol{r}) = \log\left[\frac{\sum_{(s',s)\rightarrow m_j=+1} \alpha_{j-1}(s')\gamma_j(s',s)\beta_j(s)}{\sum_{(s',s)\rightarrow m_j=-1} \alpha_{j-1}(s')\gamma_j(s',s)\beta_j(s)}\right]$$

- This requires the computation of
  - Forward metric $\alpha_{j-1}(s')$
  - Branch metric $\gamma_j(s', s)$
  - Backward metric $\beta_j(s)$

- The approach described here has been devised by Bahl, Cocke, Jelinek and Raviv, which is known as the BCJR algorithm.

L.Bahl, J.Cocke, F.Jelinek, and J.Raviv, "Optimal Decoding of Linear Codes for minimizing symbol error rate", IEEE Transactions on Information Theory, vol. IT-20(2), pp. 284-287, March 1974

i) Computation of forward metric $\alpha_j(s)$:

$$\alpha_j(s) = p\left(s_j = s, s_{j-1} = s', \boldsymbol{r}_{k<j+1}\right) = p(s, s', \boldsymbol{r}_{j>k}, \boldsymbol{r}_j)$$
$$= \textstyle\sum_{\text{all } s'} p(s, s', \boldsymbol{r}_{j>k}, \boldsymbol{r}_j)$$

Using Bayes' rule and the fact that the channel is memoryless, we obtain

$$\begin{aligned}\alpha_j(s) &= \textstyle\sum_{\text{all } s'} p(s, s', \boldsymbol{r}_{j>k}, \boldsymbol{r}_j)\\ &= \textstyle\sum_{\text{all } s'} p(s', \boldsymbol{r}_{j>k}) p(\{s, \boldsymbol{r}_j\} | \{s', \boldsymbol{r}_{j>k}\})\\ &= \textstyle\sum_{\text{all } s'} p(s', \boldsymbol{r}_{j>k}) p(\{s, \boldsymbol{r}_j\} | s')\\ &= \textstyle\sum_{\text{all } s'} \alpha_{j-1}(s') \gamma_j(s', s)\end{aligned}$$

Initial conditions: $\alpha_0(s_0 = 0) = 1$ and $\alpha_0(s_0 = s) = 0, \text{ for all } s \neq 0$

ii) Computation of backward metric $\beta_j(s)$

$$\beta_{j-1}(s') = p(\boldsymbol{r}_{j-1} < \boldsymbol{r}_k | s') \\ = \sum_{\text{all } s'} \beta_j(s) \gamma_j(s', s)$$

Initial conditions:

$$\beta_j(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases}$$

iii) Computation of branch metric $\gamma_j(s',s)$

$$\gamma_j(s',s) = p\big(\{\boldsymbol{r}_j, s\}\big|s'\big) = p(\boldsymbol{r}_j|\{s',s\})p(s|s')$$
$$= p(\boldsymbol{r}_j|\{s',s\})p\big(m_j\big)$$
$$= p(\boldsymbol{r}_j|m_j)\ p\big(m_j\big),$$

where m_j is the necessary input for a transition from $s' = s_{j-1}$ to $s = s_j$ and $p\big(m_j\big)$ is the a priori probability of the bit.

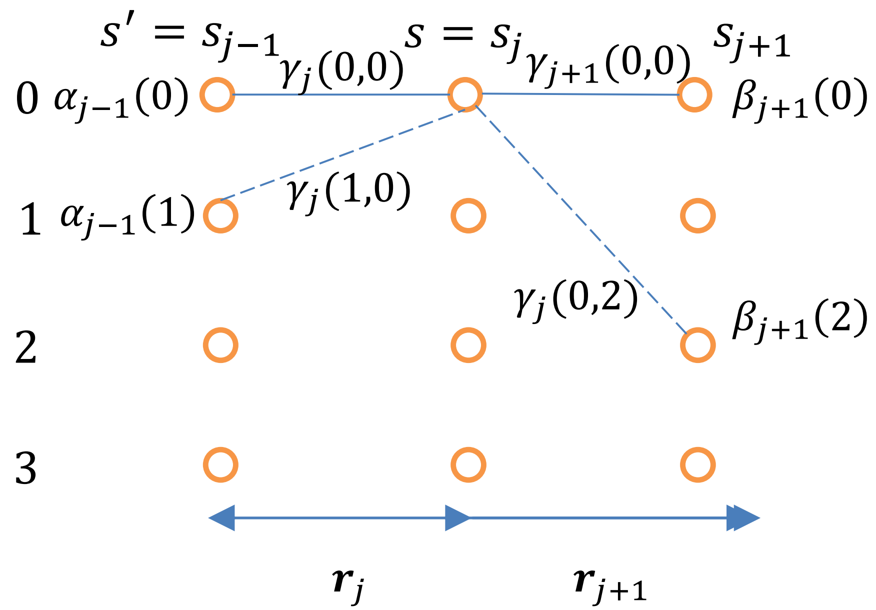
Assuming that the channel is AWGN and the modulation is BPSK or PAM-2, $p(\boldsymbol{r}_j|m_j)$ is given by

$$p(\boldsymbol{r}_j\big|m_j\big) = \prod_{l=1}^{n} p\big(r_{jl}\big|m_j\big)$$
$$= \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{\left(\frac{-E_b R}{2\sigma^2}\left(r_{jl}-c_{jl}\right)^2\right)},$$

where E_b is the bit energy and R is the code rate.

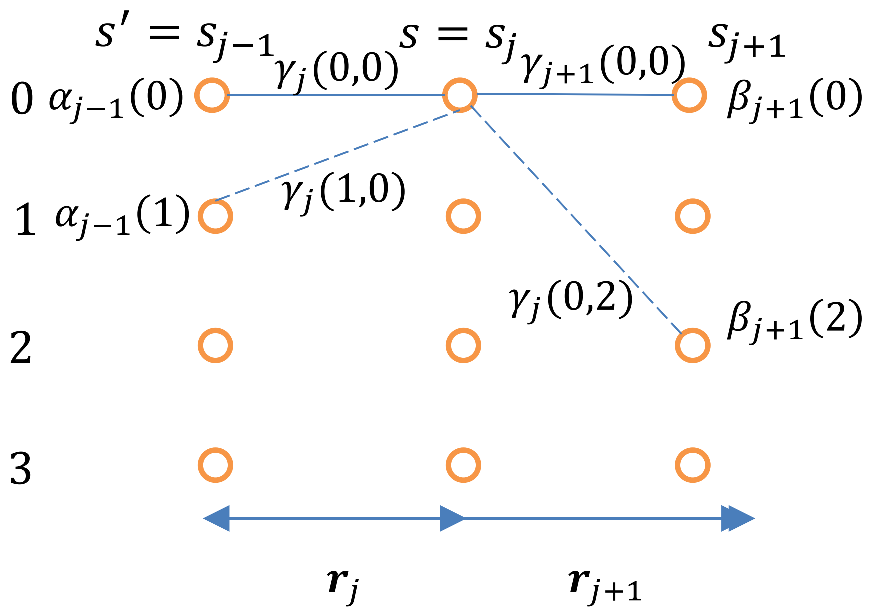
Example 2

Consider the following trellis and branch metrics



Compute $\alpha_j(0)$ and $\beta_j(0)$.

Solution:



These metrics can be computed as follows:

$$\alpha_j(0) = \alpha_{j-1}(0)\gamma_j(0,0) + \alpha_{j-1}(1)\gamma_j(1,0)$$

$$\beta_j(0) = \beta_{j+1}(0)\gamma_{j+1}(0,0) + \beta_{j+1}(2)\gamma_j(0,2)$$

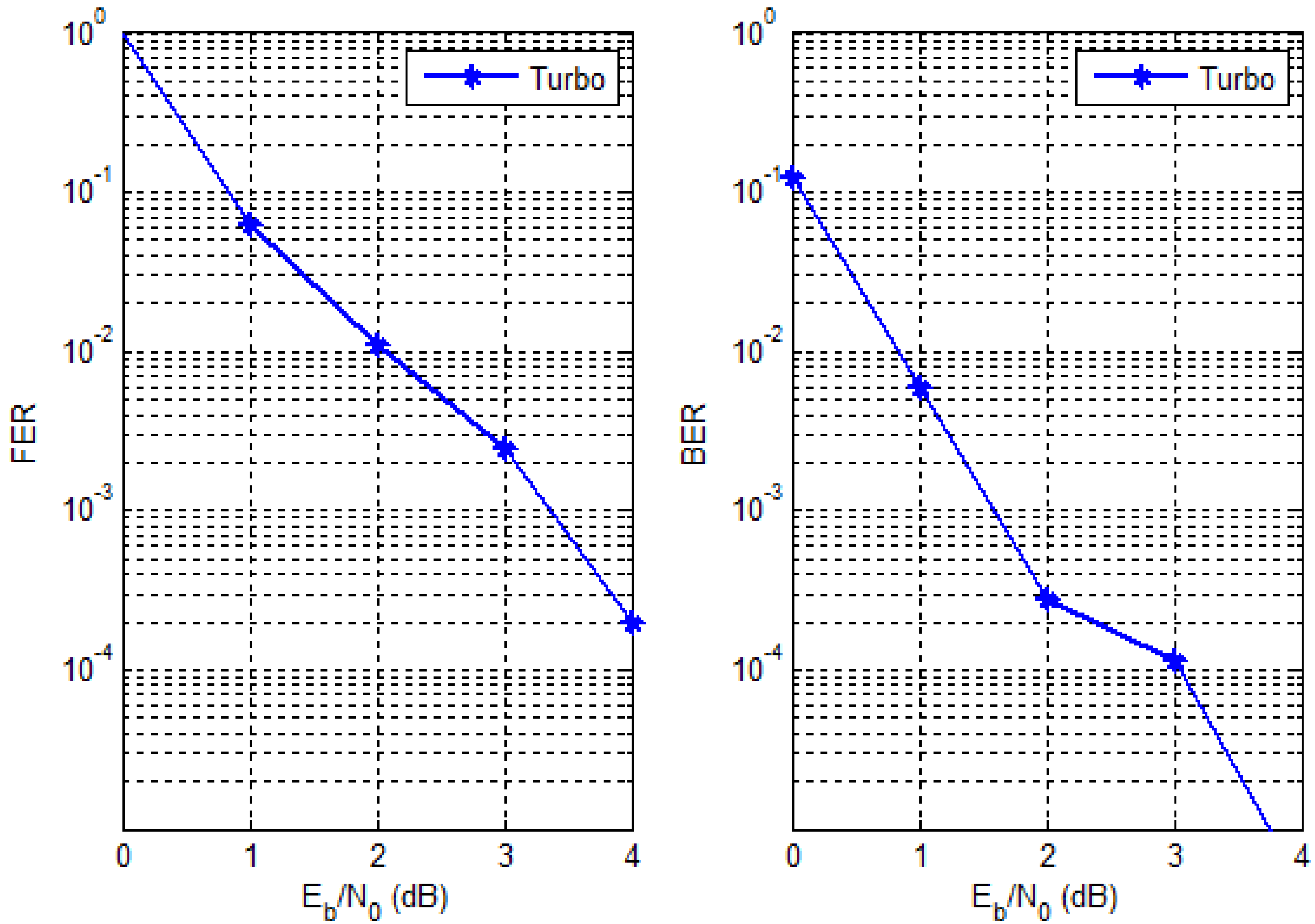
Example 3

Consider a turbo coding system with parallel concatenation and a constituent code given by

$$G(D) = 1 + D + D^2$$

Simulate the BER and the FER of the system against the SNR for a block size of $n = 256$ using a MAP decoder, AWGN channel, a range of 0 to 4 dB and 5 decoding iterations.

Solution:



Turbo
FER
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
E_b/N_0 (dB)
Turbo
BER
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
E_b/N_0 (dB)