“Turbo everywhere”

Como o “princípio turbo” está a mudar as técnicas de transmissão digital

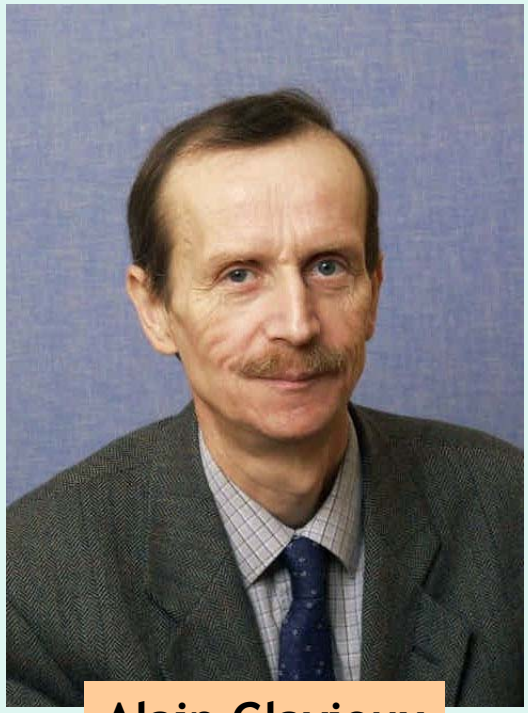
Sílvio A. Abrantes

FEUP

Turbo-códigos

- Inventados por Claude Berrou, Alain Glavieux e Punya Thitimajshima (ENST, Brest, França)
- Apresentados em 1993

Claude Berrou
(1951-)



Alain Glavieux
(1949-2004)

Turbo-códigos

- Predecessores:
  - 1979-1989: Gérard Battail
  - 1989: J. Hagenauer and P. Hoeher *(SOVA)*
  - 1992: J. Lodge, P. Hoeher and J. Hagenauer

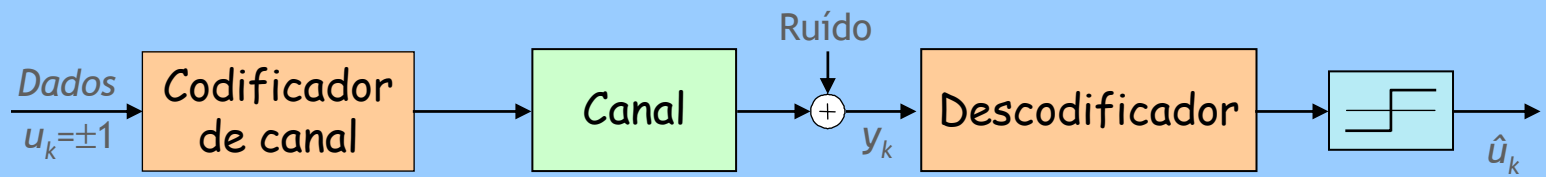
Turbo-códigos: uma revolução

- Durante décadas a comunidade da codificação tentou aproximar-se do limite de Shannon com novos códigos e técnicas

Sucesso? Algum, mas... não muito...

- Até que, com uma abordagem diferente, surgiram códigos que se aproximam a apenas décimas de dB desse limite!!

Diagrama de blocos genérico



Abordagem convencional:

- Com códigos descritos por treliças normalmente usa-se descodificação de máxima verosimilhança com o algoritmo de Viterbi (estimação ML)
- Pode também usar-se o algoritmo BCJR (estimação MAP)
  - calcula a probabilidade a posteriori $P(u_k|\mathbf{y})$ mas é muito mais complexo

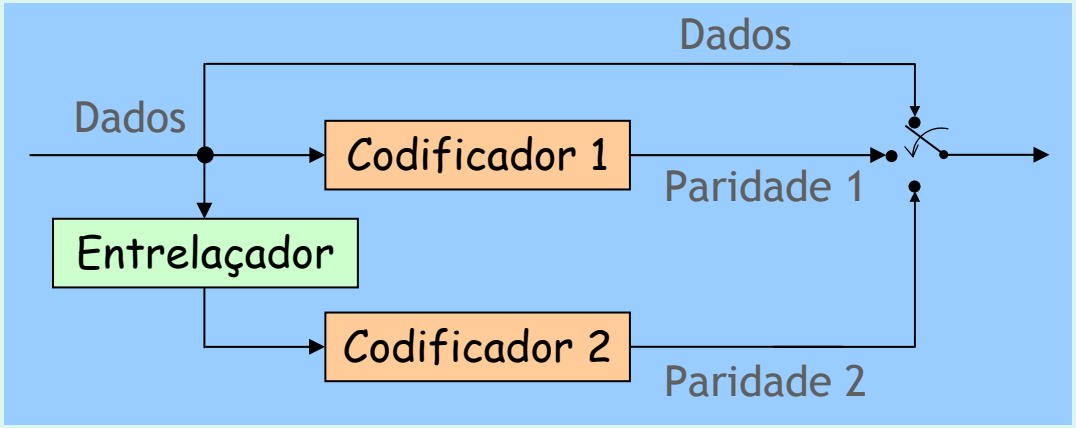
O princípio turbo

O desempenho de um receptor num sistema de comunicações digitais é significativamente melhorado se se usar:

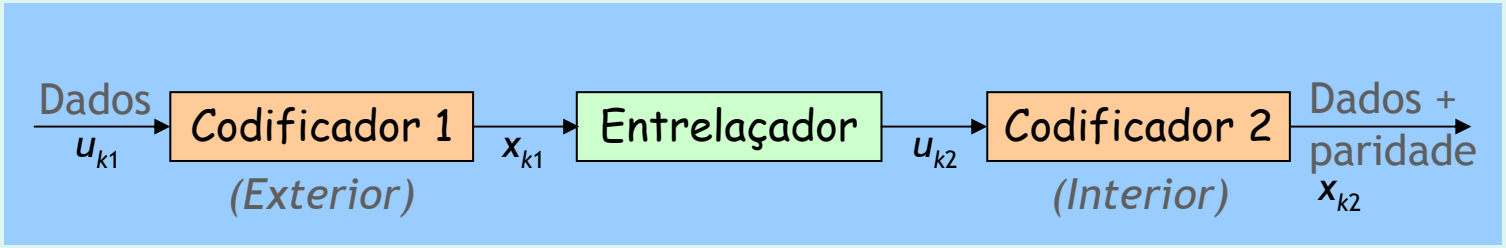
- **Codificação concatenada no emissor**
- **Detecção e descodificação iterativas no receptor**
  - Os descodificadores formam uma malha fechada de "feedback" onde circulam decisões brandas (Soft-Input Soft-Output, ou SISO) relacionadas com probabilidades *a posteriori* (APP)
  - A decisão final é tomada bit a bit (é uma decisão "rígida")
- **Entrelaçamento e desentrelaçamento**
  - Operações são realizadas em blocos de bits $\Rightarrow$ turbo-códigos são códigos de blocos

Concatenação para codificação turbo

- Em paralelo (primeira a surgir em turbo-códigos)



- Em série

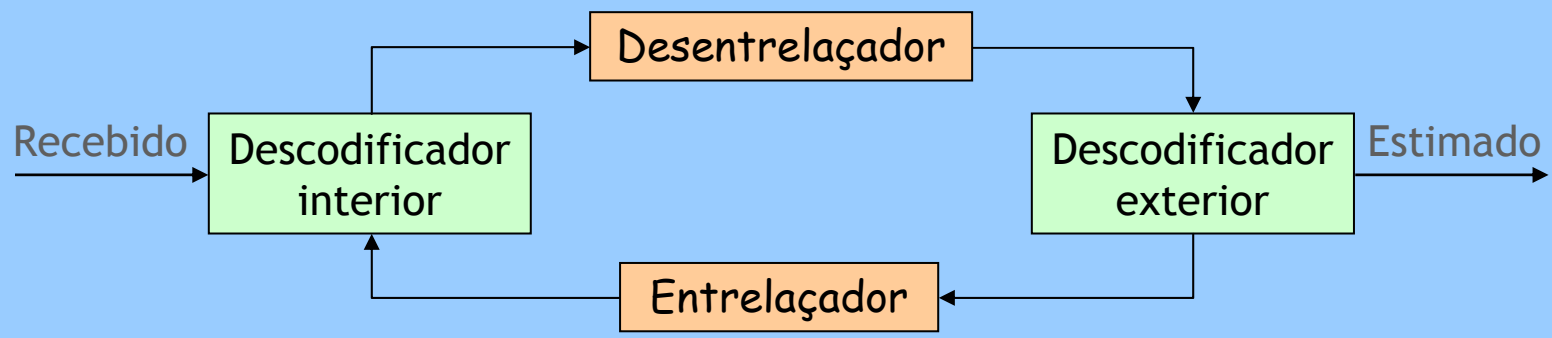


- Os codificadores estão separados por um entrelaçador

A descodificação iterativa (turbo)

Porquê "**turbo**"?

Por analogia com os motores turbo de automóveis

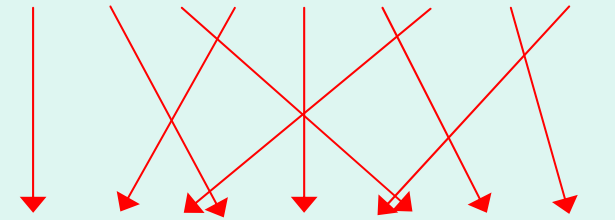


Entrelaçamento: um exemplo

Sequência original, ***x***

2	4	3	1	8	5	9	6	0

Padrão de permutação:
P = [1 4 7 2 5 9 3 6 8]



Sequência permutada, ***y***

2	1	9	4	8	0	3	5	6

Matriz de permutação:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$y[i] = x[P[i]]$$

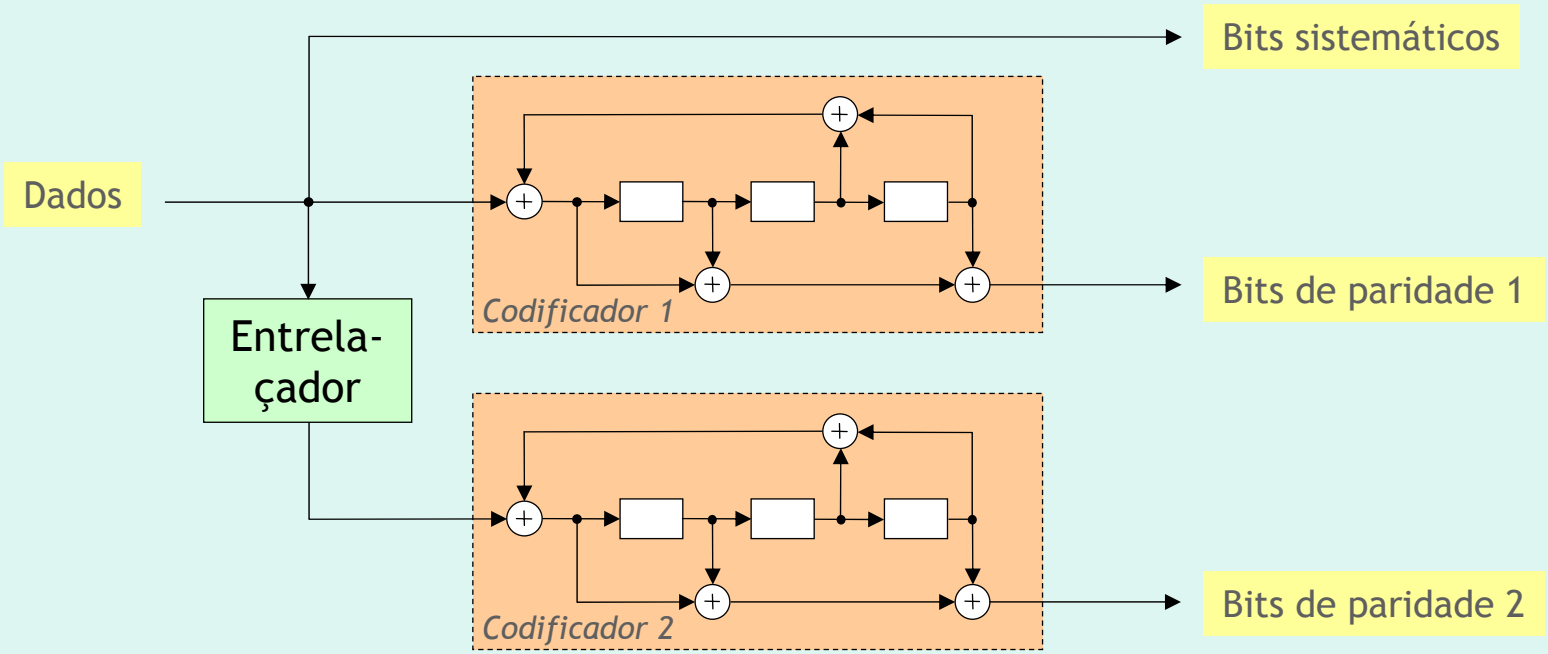
$$x[P[i]] = y[i]$$

$$\mathbf{y} = \mathbf{xP}$$

$$\mathbf{x} = \mathbf{yP}^T$$

Codificadores constituintes dos turbo-códigos

Geralmente usam-se codificadores convolucionais recursivos iguais:

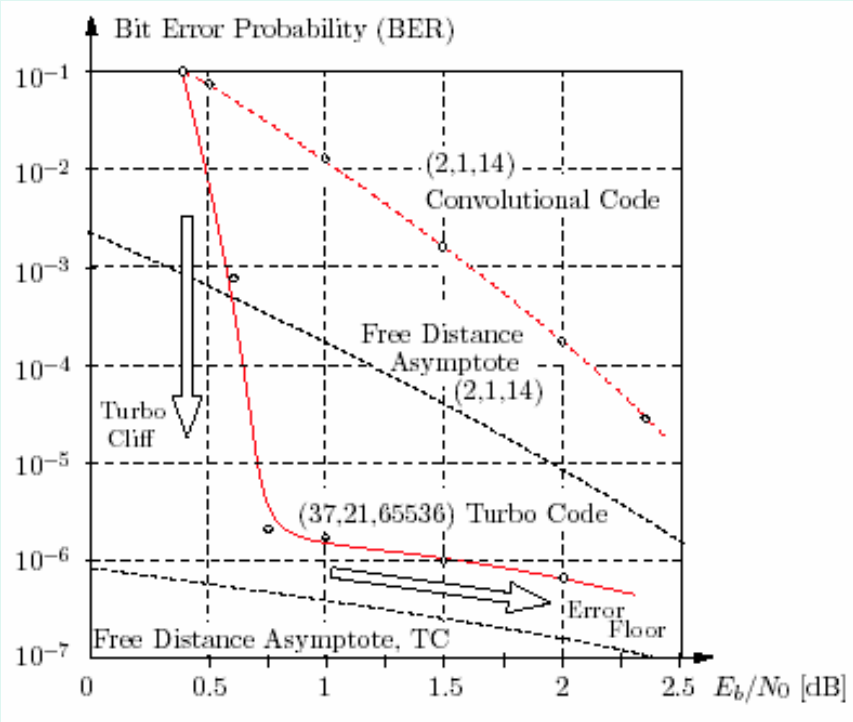


Gerador do ramo directo: 15_8

Gerador do ramo de "feedback": 13_8

Comparação de desempenho

Probabilidade de erro com códigos convolucionais e turbo



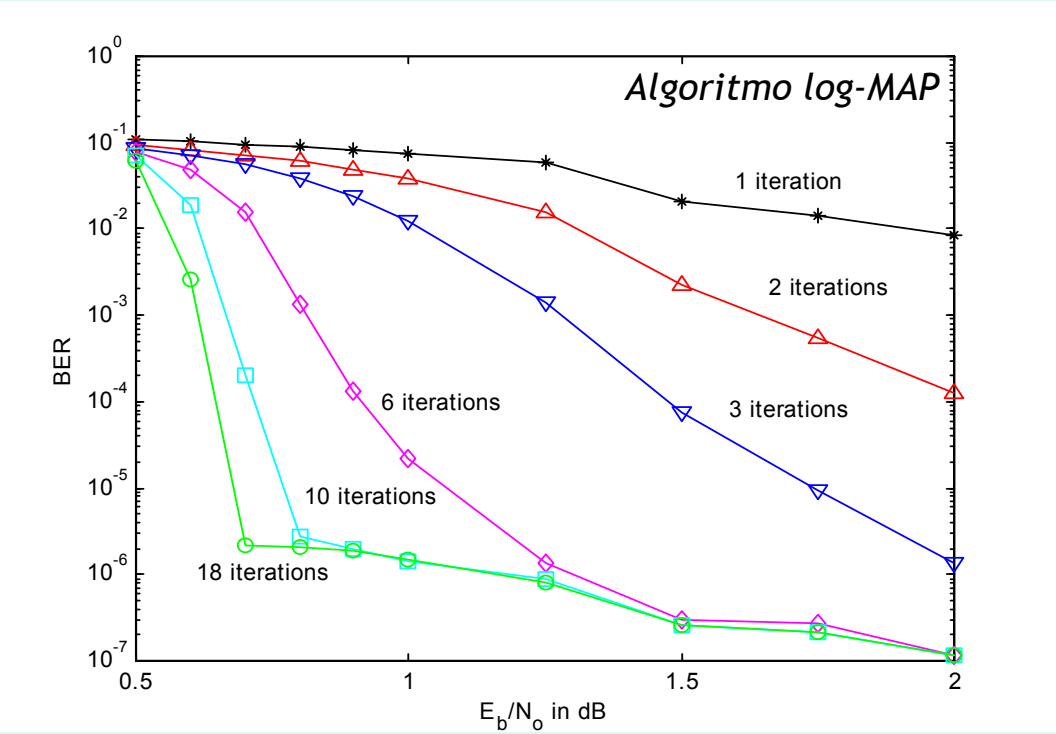
Fonte: C. Schlegel & L. Perez, *Trellis and Turbo Coding*, IEEE Press-Wiley, 2004

Aplicações dos turbo-códigos

- Telemóveis 3G
  - UMTS, WCDMA (3GPP/evolução de GSM)
  - CDMA2000 (3GPP2/evolução de IS-95)
- Televisão digital
  - Norma DVB-RCS ("Return Channel via Satellite")
- Redes locais e metropolitanas sem fios
  - Norma IEEE 802.16a ("Fixed Broadband Wireless Access Systems")
- Comunicações espaciais
  - NASA: Messenger ('04), Mars Reconnaissance Orbiter & Rover ('05), STEREO ('05), Pluto ('06), Kepler ('07), etc.
  - ESA: SMART-1 ('03), Rosetta
- Sistemas militares
- Gravação magnética

Descodificação turbo é iterativa

Probabilidade de erro em função do número de iterações



Fonte: M. Valenti, "Turbo Codes and Iterative Processing", *IEEE New Zealand Wireless Communications Symposium, Nov. 1998*

Probabilidades e logaritmos

- Razão de probabilidades *a posteriori*

$$\frac{P(u_k = +1 | \mathbf{y})}{P(u_k = -1 | \mathbf{y})} \underset{H_0}{\overset{H_1}{\gtrless}} 1$$

Critério MAP

- LLR ("*log-likelihood ratio*") *a posteriori*

$$L(u_k | \mathbf{y}) = \ln \frac{P(u_k = +1 | \mathbf{y})}{P(u_k = -1 | \mathbf{y})} \underset{H_0}{\overset{H_1}{\gtrless}} 0$$

Critério MAP

- LLR *a priori*

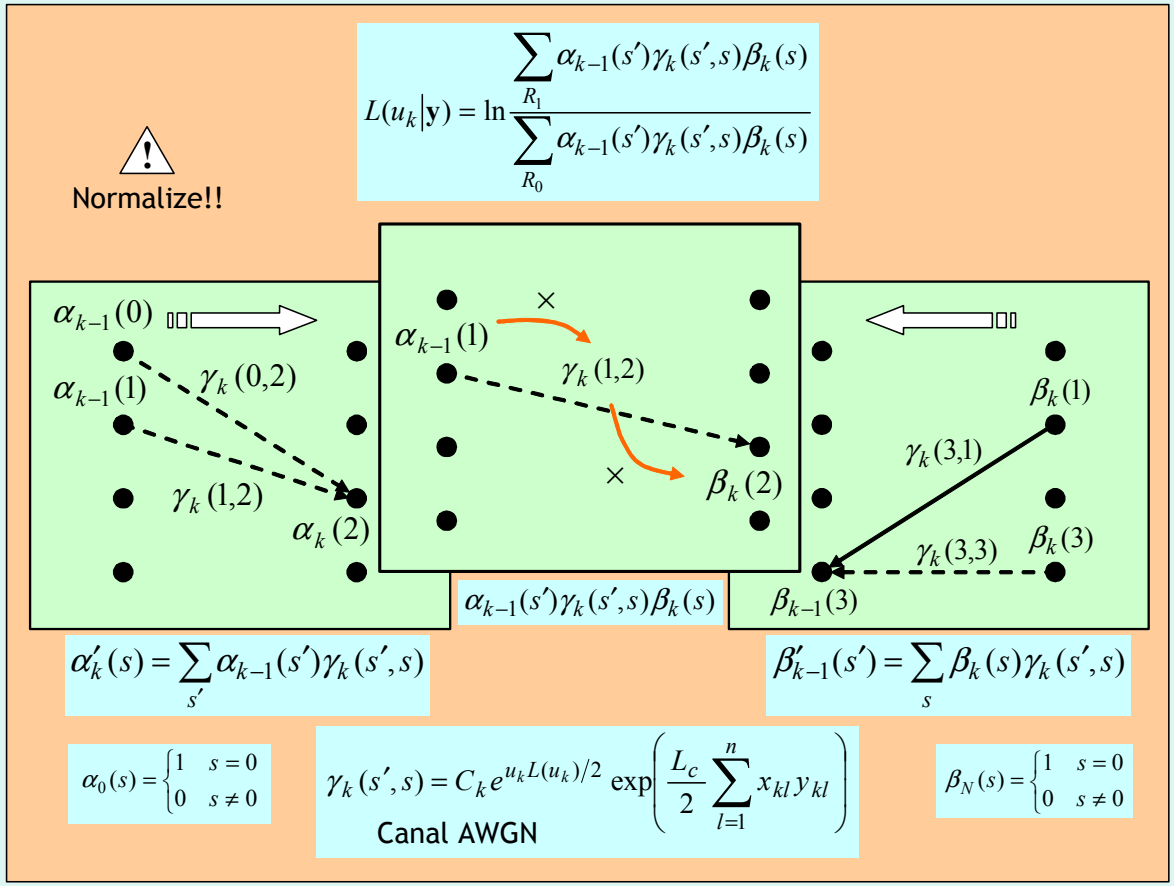
$$L(u_k) = \ln \frac{P(u_k = +1)}{P(u_k = -1)}$$

Estimação MAP e algoritmo BCJR

- O receptor óptimo usa o critério da probabilidade *a posteriori* máxima (MAP) para tomar decisões sobre u_k.
- Algoritmo de cálculo de $L(u_k|\mathbf{y})$: BCJR
  (Bahl, Cocke, Jelinek e Raviv, 1974)
  - Outros nomes para o mesmo: algoritmo MAP, algoritmo APP, “forward-backward algorithm”
  - Variantes mais simples: log-MAP, max-log-MAP, SOVA

O algoritmo BCJR, ou MAP

Sumário de expressões usadas



Variantes simplificativas do algoritmo BCJR

- Algoritmo SOVA (Soft Output Viterbi Algorithm)
- Algoritmo log-MAP
- Algoritmo max-log-MAP

São definidas novas variáveis, *A*, Γ e *B*:

$$\Gamma_k(s',s) = \ln \gamma_k(s',s) = \ln C_k + \frac{u_k L(u_k)}{2} + \frac{L_c}{2}\sum_{l=1}^{n} x_{kl} y_{kl}$$

$$A_k(s) = \ln \alpha_k(s) = \max_{s'}{}^{*}\left[A_{k-1}(s') + \Gamma_k(s',s)\right] \qquad A_0(s) = \begin{cases} 0 & s = 0 \\ -\infty & s \neq 0 \end{cases}$$

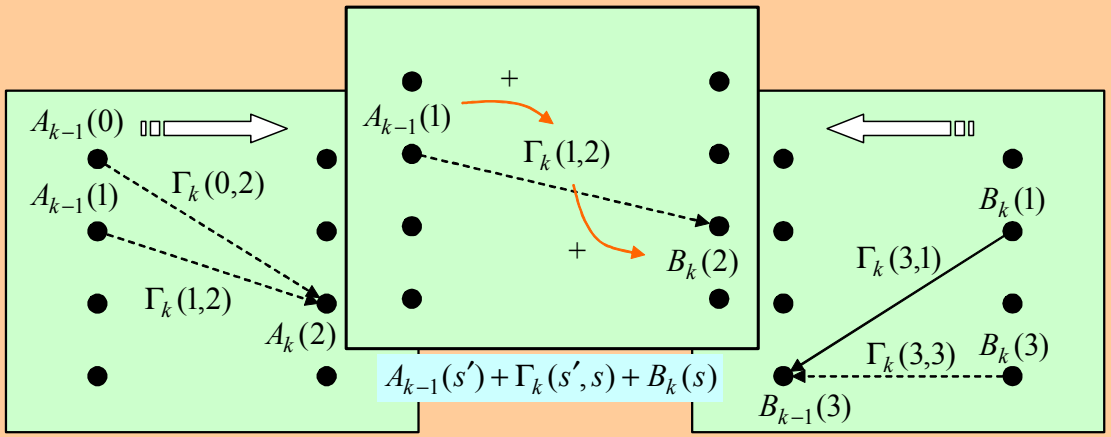
$$B_{k-1}(s') = \ln \beta_{k-1}(s') = \max_{s}{}^{*}\left[B_k(s) + \Gamma_k(s',s)\right] \qquad B_N(s) = \begin{cases} 0 & s = 0 \\ -\infty & s \neq 0 \end{cases}$$

$$\max{}^{*}(a,b) = \begin{cases} \max(a,b) + \ln(1 + e^{-|a-b|}) & \text{algoritmo log} - \text{MAP} \\ \max(a,b) & \text{algoritmo max} - \log - \text{MAP} \end{cases}$$

Os algoritmos log-MAP e max-log-MAP

Sumário de expressões usadas

$$L(u_k|\mathbf{y}) = \max_{R_1}{}^{*}\left[A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)\right] - \max_{R_0}{}^{*}\left[A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)\right]$$

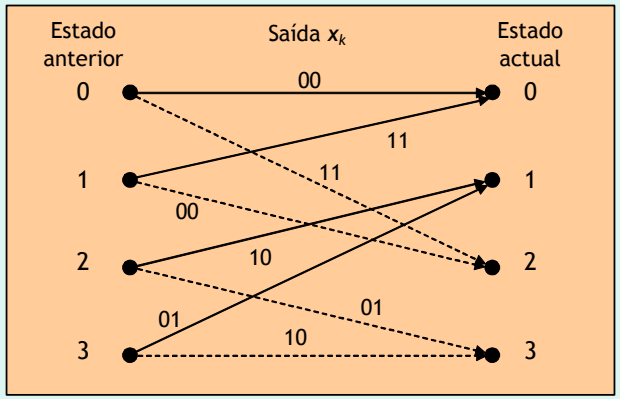


$$\Gamma_k(s',s) = \ln C_k + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} x_{kl} y_{kl}$$

Canal AWGN

Um exemplo numérico com o algoritmo BCJR

- Canal AWGN: $a = 1$ e $\frac{E_c}{N_0} = 1 \ \ dB \quad \Rightarrow L_c = 4aE_c/N_0 = 5$
- Treliça do codificador convolucional de taxa ½:



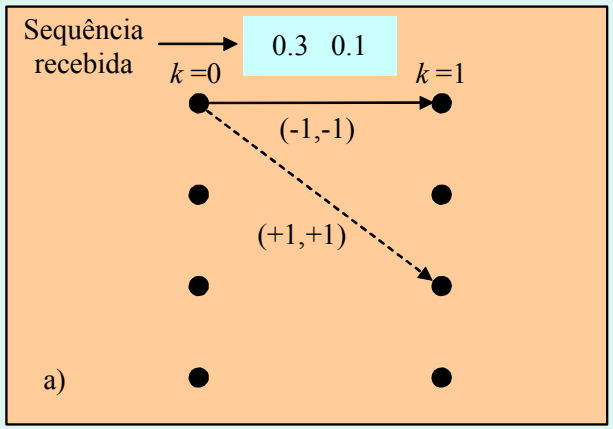
- Sequência ±1 enviada começa e termina no estado nulo
- Sequência recebida:

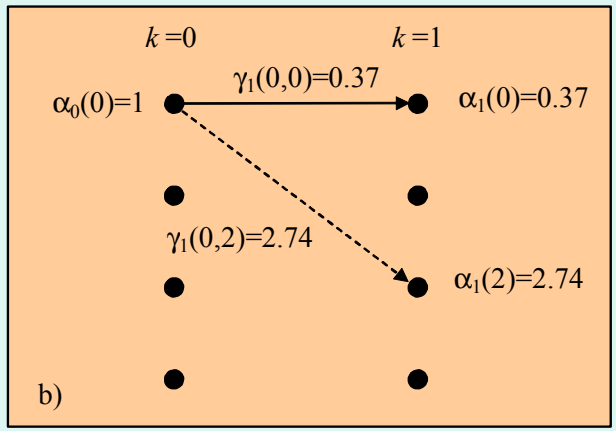
$$\mathbf{y} = \overbrace{0.3 \quad 0.1}^{y_1} \ \overbrace{-0.5 \quad 0.2}^{y_2} \ \overbrace{0.8 \quad 0.5}^{y_3} \ \overbrace{-0.5 \quad 0.3}^{y_4} \ \overbrace{0.1 \quad -0.7}^{y_5} \ \overbrace{1.5 \quad -0.4}^{y_6}$$

Qual terá sido a sequência de seis bits enviada?

Um exemplo numérico com o algoritmo BCJR

- No instante $k = 1$ temos





$$\gamma_1(0,0) = C_k e^{u_k L(u_k)/2} \exp\left[\frac{L_c}{2}\left(x_{11}y_{11} + x_{12}y_{12}\right)\right] = e^{[2.5\times(-1\times 0.3 - 1\times 0.1)]} = e^{-1} = 0.37 \quad \text{(com } L(u_k)=0\text{)}$$

$$\gamma_1(0,2) = e^{\left[2.5\times(1\times 0.3 + 1\times 0.1)\right]} = e = 2.74$$

$$\alpha_0(s) = \begin{cases} 1 & s = 0 \\ 0 & s \neq 0 \end{cases}$$

$$\alpha_0(0)\gamma_1(0,0) = 1\times 0.37 = 0.37 \Rightarrow \alpha_1(0) = \frac{0.37}{0.37 + 2.74} = 0.12$$

$$\alpha_0(0)\gamma_1(0,2) = 1\times 2.74 = 2.74 \Rightarrow \alpha_1(2) = \frac{2.74}{0.37 + 2.74} = 0.88$$

Um exemplo numérico com o algoritmo BCJR

- Mais exemplos de cálculo de α, γ e β

$$\gamma_3(2,3) = e^{[2.5\times(-1\times(0.8)+1\times0.5)]} = e^{-0.75} = 0.47$$

$$\alpha_3(3) = \frac{\alpha_2(2)\gamma_3(2,3)+\alpha_2(3)\gamma_3(3,3)}{4.31} = \frac{0.01\times0.47+0.92\times2.13}{4.31} = 0.45$$

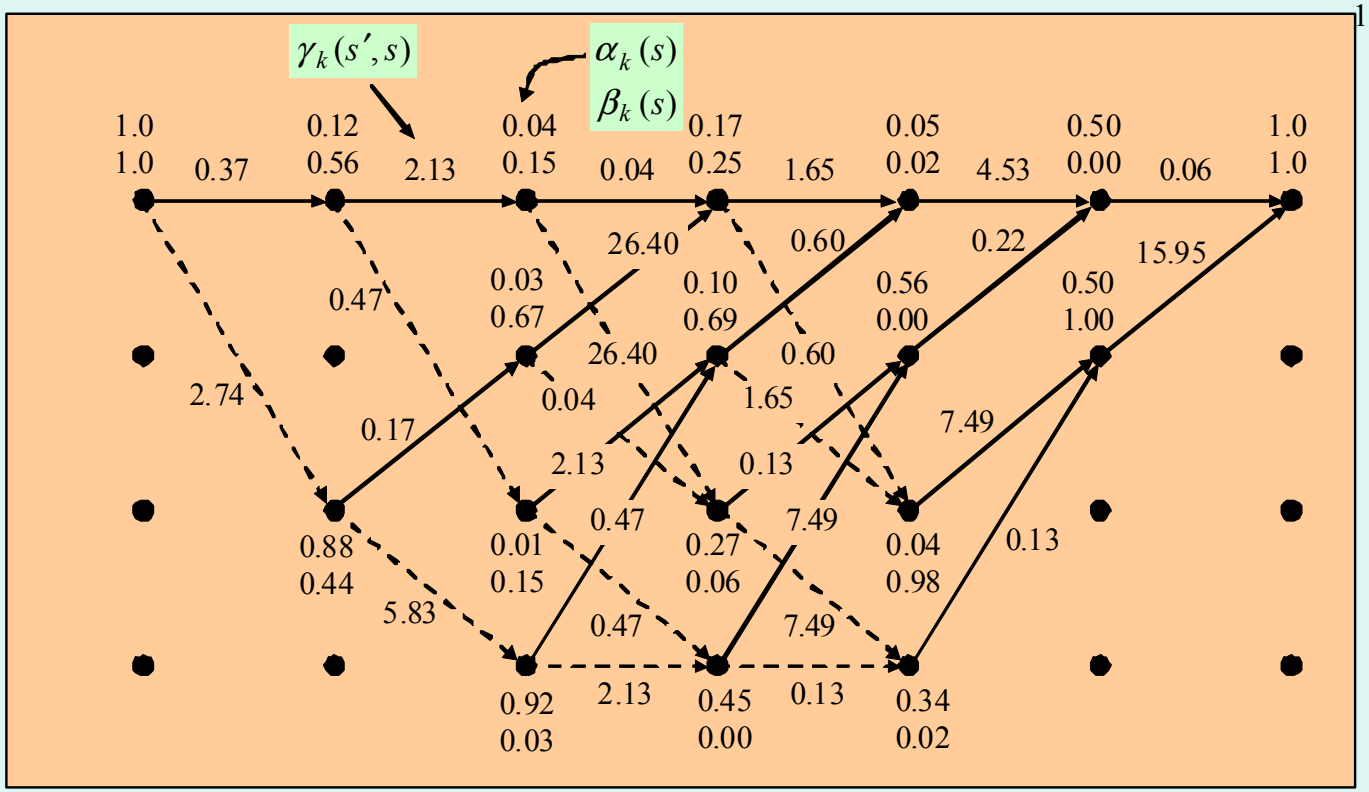
$$\beta_2(2) = \frac{\beta_3(1)\gamma_3(2,1)+\beta_3(3)\gamma_3(2,3)}{9.96} = \frac{0.69\times2.13+0.001\times0.47}{9.96} = 0.15$$

$$\sum_s \alpha'_3(s) = \alpha'_3(0)+\alpha'_3(1)+\alpha'_3(2)+\alpha'_3(3) = 0.716+0.452+1.182+1.959 = 4.31$$

$$\sum_{s'} \beta'_2(s') = \beta'_2(0)+\beta'_2(1)+\beta'_2(2)+\beta'_2(3) = 1.476+6.689+1.469+0.327 = 9.96$$

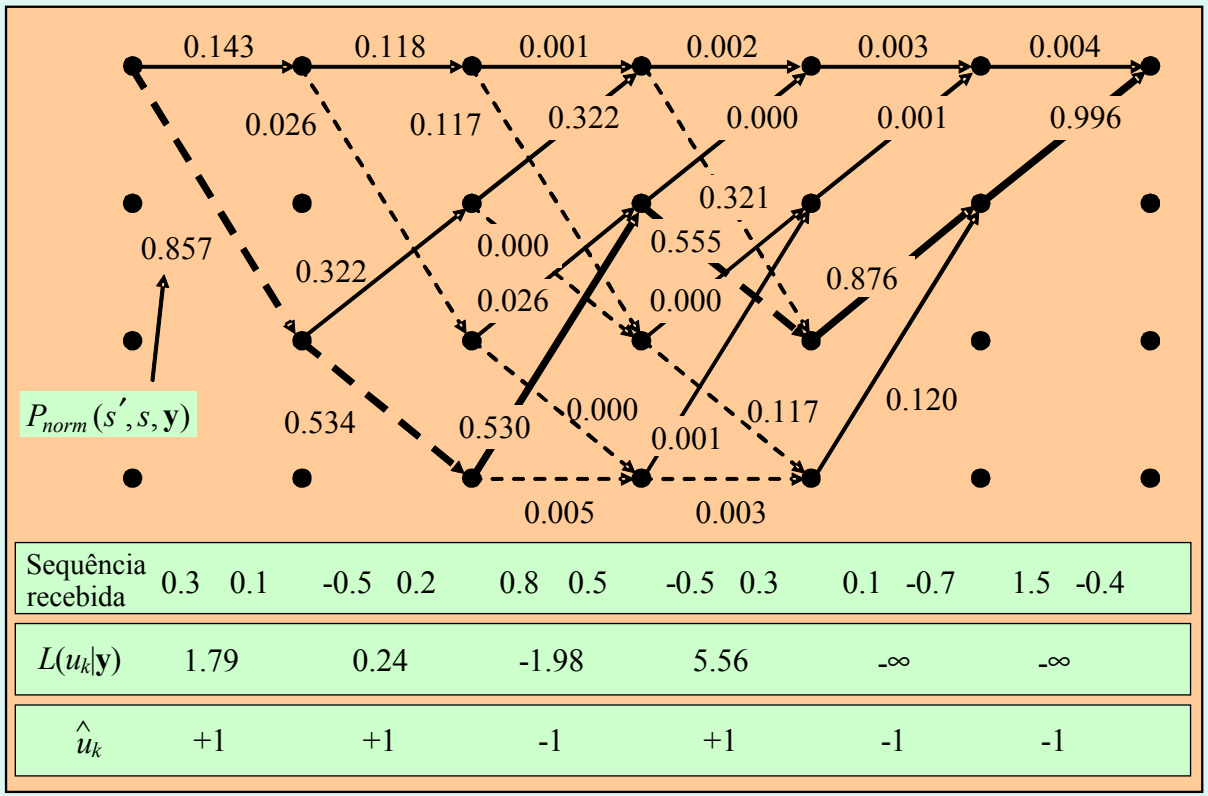
Um exemplo numérico com o algoritmo BCJR

Valores de α, β e γ ao fim de toda a sequência de seis símbolos recebida



Um exemplo numérico com o algoritmo BCJR

Valores da probabilidade conjunta $P_{norm}(s',s,y)$ e da LLR $L(u_k|y)$



Sequência recebida	0.3 0.1	-0.5 0.2	0.8 0.5	-0.5 0.3	0.1 -0.7	1.5 -0.4
$L(u_k\|\mathbf{y})$	1.79	0.24	-1.98	5.56	-∞	-∞
$\hat{u}_k$	+1	+1	-1	+1	-1	-1

Um exemplo numérico com o algoritmo BCJR

Valores das LLR a posteriori

$$L(u_1|\mathbf{y}) = \ln\frac{P_{norm}(0,2,\mathbf{y})}{P_{norm}(0,0,\mathbf{y})} = \ln\frac{0.857}{0.143} = 1.79$$

$$L(u_2|\mathbf{y}) = \ln\frac{P_{norm}(0,2,\mathbf{y}) + P_{norm}(2,3,\mathbf{y})}{P_{norm}(0,0,\mathbf{y}) + P_{norm}(2,1,\mathbf{y})} = \ln\frac{0.560}{0.440} = 0.24$$

$$L(u_3|\mathbf{y}) = \ln\frac{P_{norm}(0,2,\mathbf{y}) + P_{norm}(1,2,\mathbf{y}) + P_{norm}(2,3,\mathbf{y}) + P_{norm}(3,3,\mathbf{y})}{P_{norm}(0,0,\mathbf{y}) + P_{norm}(1,0,\mathbf{y}) + P_{norm}(2,1,\mathbf{y}) + P_{norm}(3,1,\mathbf{y})} = \ln\frac{0.122}{0.878} = -1.98$$

$$L(u_4|\mathbf{y}) = \ln\frac{P_{norm}(0,2,\mathbf{y}) + P_{norm}(1,2,\mathbf{y}) + P_{norm}(2,3,\mathbf{y}) + P_{norm}(3,3,\mathbf{y})}{P_{norm}(0,0,\mathbf{y}) + P_{norm}(1,0,\mathbf{y}) + P_{norm}(2,1,\mathbf{y}) + P_{norm}(3,1,\mathbf{y})} = \ln\frac{0.996}{0.004} = 5.56$$

$$L(u_5|\mathbf{y}) = \ln\frac{0}{P(0,0,\mathbf{y}) + P(1,0,\mathbf{y}) + P(2,1,\mathbf{y}) + P(3,1,\mathbf{y})} = -\infty$$

$$L(u_6|\mathbf{y}) = \ln\frac{0}{P(0,0,\mathbf{y}) + P(1,0,\mathbf{y})} = -\infty$$

Sequência estimada

$\hat{u}_k$ = +1 +1 -1 +1 -1 -1

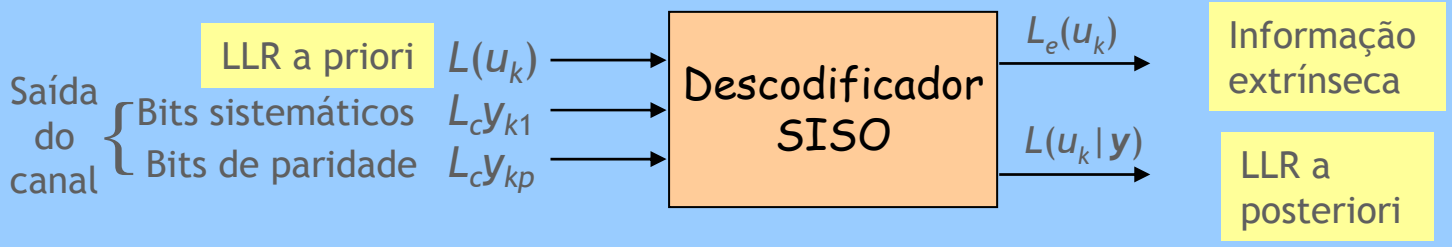
Descodificação iterativa

- É uma abordagem relativamente recente (1993).
- É um intercâmbio iterativo de decisões brandas ("soft") entre dois descodificadores no receptor.
- Que decisões brandas? Estimativas cada vez mais apuradas da informação *a priori:*

$$L(u_k) = \ln \frac{P(u_k = +1)}{P(u_k = -1)}$$

LLR a priori

Descodificador SISO (*Soft-In Soft-Out*)



$$L(u_k|\mathbf{y}) = L(u_k) + L_c y_{k1} + L_e(u_k)$$

$$\Downarrow$$

$$L_e(u_k) = L(u_k|\mathbf{y}) - L(u_k) - L_c y_{k1}$$

Taxa do código

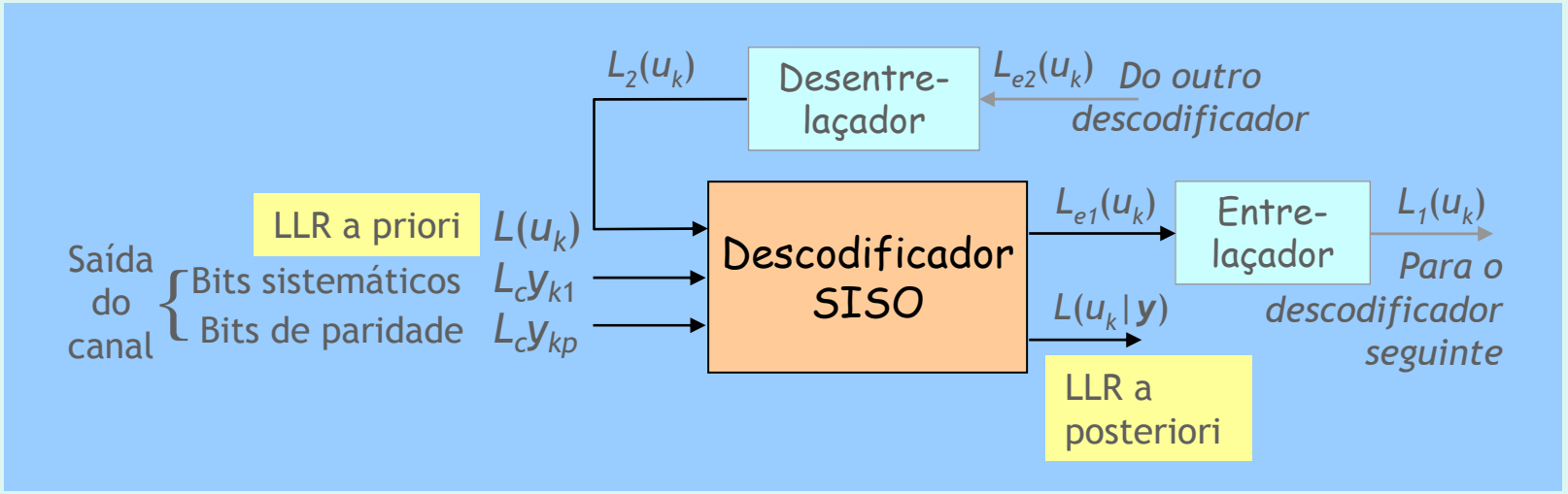
$$L_c = 4aR_c \frac{E_b}{N_0}$$

AWGN: $a = 1$

Mas... é preciso calcular $L(u_k|\mathbf{y})$! Como? Com BCJR...

Descodificação iterativa

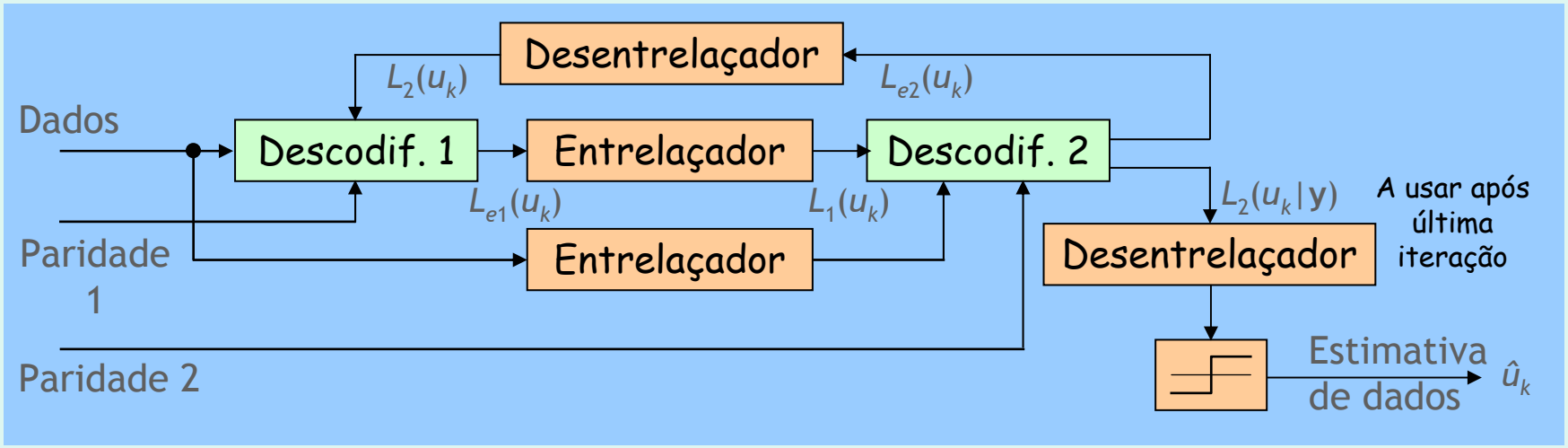
Concatenação em paralelo: descodificador 1



- Só a informação extrínseca circula entre descodificadores.
- O entrelaçador e o desentrelaçador convertem as informações extrínsecas em estimativas melhoradas da LLR a priori.
- No descodificador 2 troca-se o entrelaçador e o desentrelaçador.

Descodificação iterativa

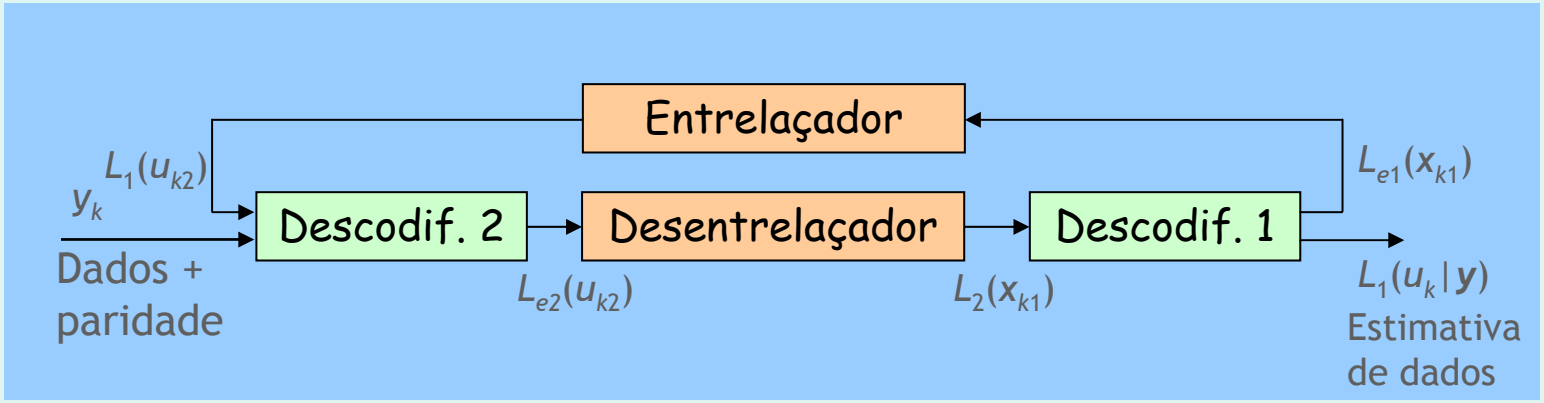
Para concatenação em paralelo



- $L_2(u_k)$ é normalmente inicializado em zero.

Descodificação iterativa

Para concatenação em série



- Repare-se que o descodificador 1 só tem uma entrada (a LLR a priori estimada pelo descodificador 2)

Passos da descodificação iterativa

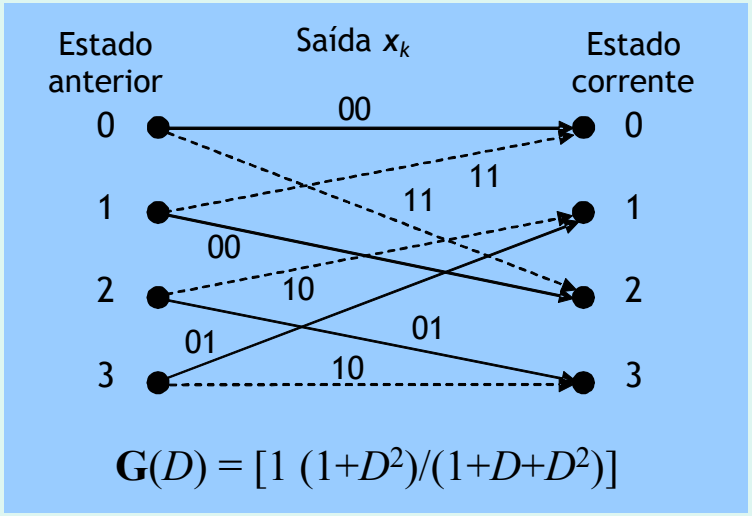
1. Descodificador 1 envia informação "extrínseca" ao descodificador 2

2. Descodificador 2 envia ao descodificador 1 nova informação extrínseca

3. O processo $1 \rightarrow 2 \rightarrow 1 \rightarrow \ldots$ prossegue iterativamente até se satisfazer o critério de paragem pré-estabelecido.

Um turbo-exemplo numérico
(concatenação em paralelo)

Treliça dos codificadores convolucionais



$$\mathbf{G}(D) = [1 \ (1+D^2)/(1+D+D^2)]$$

Padrão de entrelaçamento:

P = [1 4 7 2 5 9 3 6 8]

Se sequência original

⇩

Sequência entrelaçada

2	4	3	1	8	5	9	6	0

2	1	9	4	8	0	3	5	6

- Perfuração de códigos para que R_c = ½
- Canal AWGN com E_c/N_0 = 0.25 e a = 1 ⇒ L_c = 1
- Sequência de nove bits enviada: **-1 -1 -1 … -1**
- Sequência de 18 valores reais recebida:

Como estimar a sequência enviada?

y = **0.3 -4.0 -1.9 -2.0 -2.4 -1.3 1.2 -1.1 0.7 -2.0 -1.0 -2.1 -0.2 -1.4 -0.3 -0.1 -1.1 0.3**

paridade 1 *paridade 2*

Turbo-exemplo numérico

1) Sequências de entrada do descodificador turbo

P

k	1	2	3	4	5	6	7	8	9
$L_c \mathbf{y}_{k1}$	0.3	-1.9	-2.4	1.2	0.7	-1.0	-0.2	-0.3	-1.1
$L_c \mathbf{y}_{k1}^{(P)}$	0.3	1.2	-0.2	-1.9	0.7	-1.1	-2.4	-1.0	-0.3
$L_c \mathbf{y}_{kp}^{(1)}$	-4.0	0	-1.3	0	-2.0	0	-1.4	0	0.3
$L_c \mathbf{y}_{kp}^{(2)}$	0	-2.0	0	-1.1	0	-2.1	0	-0.1	0

Todos os zeros de paridade são devidos à perfuração nos codificadores

Turbo-exemplo numérico

2) Saída do descodificador 1 após 1ª meia iteração

4 Erros

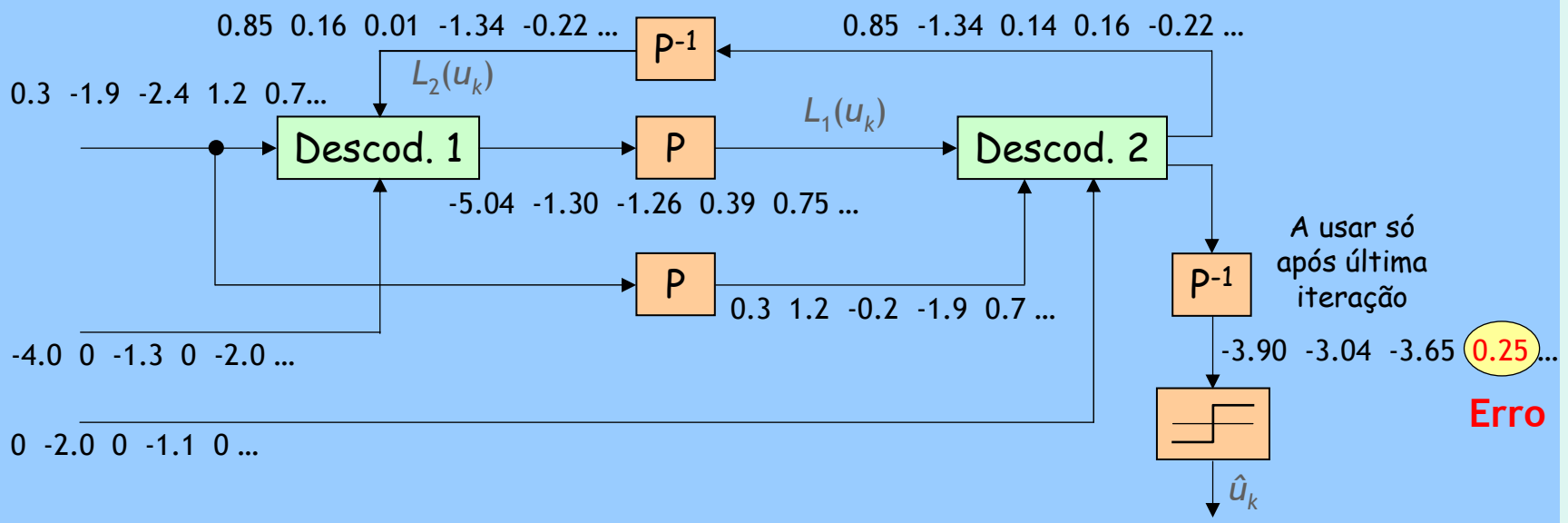
	$L_1(u_k \mid y)$	-4.74	-3.20	-3.66	1.59	1.45	-0.74	0.04	0.04	-1.63
	$L(u_k)$	0	0	0	0	0	0	0	0	0
	$L_c\, y_{k1}$	0.30	-1.90	-2.40	1.20	0.70	-1.00	-0.20	-0.30	-1.10
P	$L_{e1}(u_k)$	-5.04	-1.30	-1.26	0.39	0.75	0.26	0.24	0.34	-0.53
	$L_1(u_k)$	-5.04	0.39	0.24	-1.30	0.75	-0.53	-1.26	0.26	0.34

3) Saída do descodificador 2 após 1ª iteração

	$L_2(u_k \mid y)$	-3.90	0.25	0.18	-3.04	1.23	-1.44	-3.65	-0.72	0.04
	$L_1(u_k)$	-5.04	0.39	0.24	-1.30	0.75	-0.53	-1.26	0.26	0.34
	$L_c \mathbf{y}_{k1}^{(P)}$	0.30	1.20	-0.20	-1.90	0.70	-1.10	-2.40	-1.00	-0.30
P^{-1}	$L_{e2}(u_k)$	0.85	-1.34	0.14	0.16	-0.22	0.19	0.01	0.02	0.00
	$L_2(u_k)$	0.85	0.16	0.01	-1.34	-0.22	0.02	0.14	0.00	0.19

Turbo-exemplo numérico

Sequências após primeira iteração



Turbo-exemplo numérico

4) Saídas dos descodificadores turbo durante cinco iterações

Erros

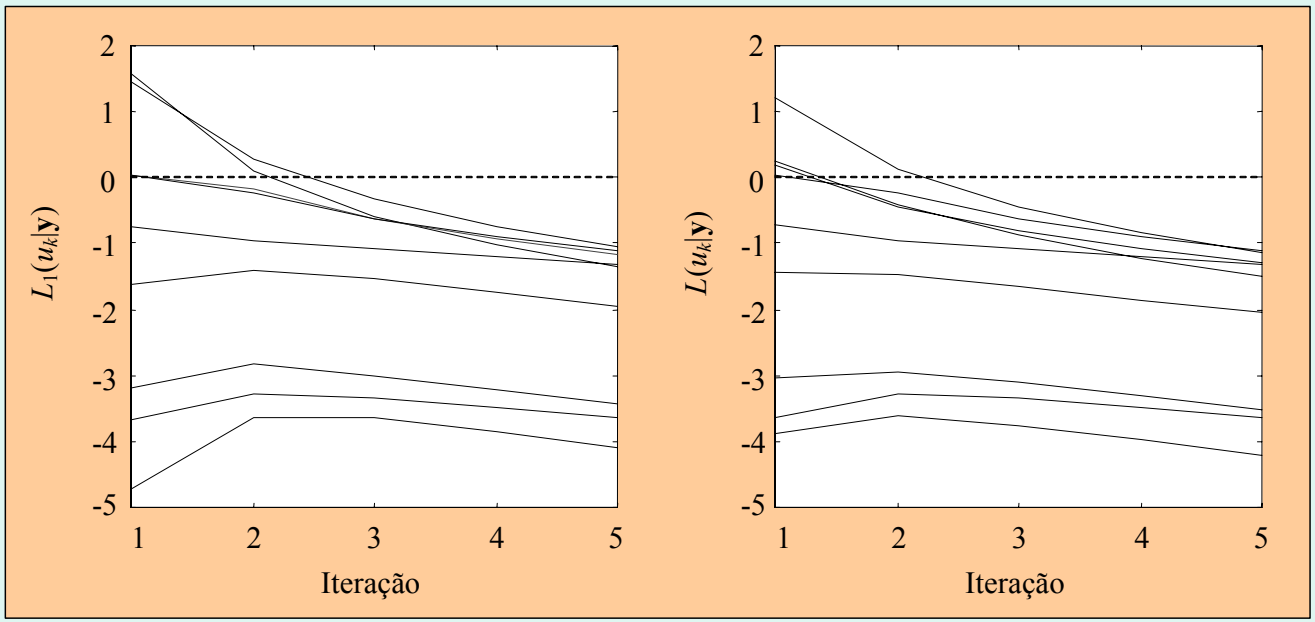
Iteração	$k \rightarrow$	1	2	3	4	5	6	7	8	9
1	$L_1(u_k \mid \mathbf{y})$	-4.74	-3.20	-3.66	1.59	1.45	-0.74	0.04	0.04	-1.63
2	$L_1(u_k \mid \mathbf{y})$	-3.64	-2.84	-3.28	0.11	0.27	-0.95	-0.17	-0.25	-1.40
3	$L_1(u_k \mid \mathbf{y})$	-3.65	-3.00	-3.35	-0.58	-0.34	-1.07	-0.61	-0.63	-1.53
4	$L_1(u_k \mid \mathbf{y})$	-3.85	-3.21	-3.49	-1.02	-0.74	-1.20	-0.93	-0.90	-1.75
5	$L_1(u_k \mid \mathbf{y})$	-4.08	-3.42	-3.64	-1.35	-1.05	-1.32	-1.18	-1.11	-1.95
1	$L(u_k \mid \mathbf{y})$	-3.90	-3.04	-3.65	0.25	1.23	-0.72	0.18	0.04	-1.44
2	$L(u_k \mid \mathbf{y})$	-3.61	-2.96	-3.29	-0.41	0.13	-0.97	-0.43	-0.25	-1.48
3	$L(u_k \mid \mathbf{y})$	-3.75	-3.11	-3.35	-0.87	-0.45	-1.08	-0.80	-0.63	-1.66
4	$L(u_k \mid \mathbf{y})$	-3.98	-3.32	-3.50	-1.22	-0.85	-1.21	-1.07	-0.90	-1.86
5	$L(u_k \mid \mathbf{y})$	-4.21	-3.52	-3.65	-1.51	-1.15	-1.33	-1.28	-1.11	-2.06

Após 2 iterações todas as decisões rígidas são correctas!

Turbo-exemplo numérico

LLR a posteriori, $L(u_k|\mathbf{y})$

Descodificador 1 **Descodificador 2**



Os valores de $L(u_k|\mathbf{y})$ são todos negativos (como deviam!) após a 2ª iteração.

Concatenação para codificação turbo

- Na concatenação em paralelo o projecto do entrelaçador é crucial
- Vantagens da concatenação em série:
  - Projecto do entrelaçador é mais fácil
  - Maior flexibilidade $\Rightarrow$ âmbito de aplicação mais alargado

Entrelaçadores propostos na literatura*

- Entrelaçadores aleatórios (de Berrou & Glavieux)
- Entrelaçadores de espalhamento S
  ("S-random interleavers", de Divsalar & Pollara)
  - "High Spread-Random (HSR) interleavers"
  - "Dithered-Diagonal (DD) interleavers"
  - "Dithered golden interleavers"
  - "Dithered Relative Prime (DRP) interleavers"

* Haykin, S. et al, "Turbo-MIMO for Wireless Communications", *IEEE Communications Magazine*, Out. 2004, e outras fontes

Aplicações do “princípio turbo”

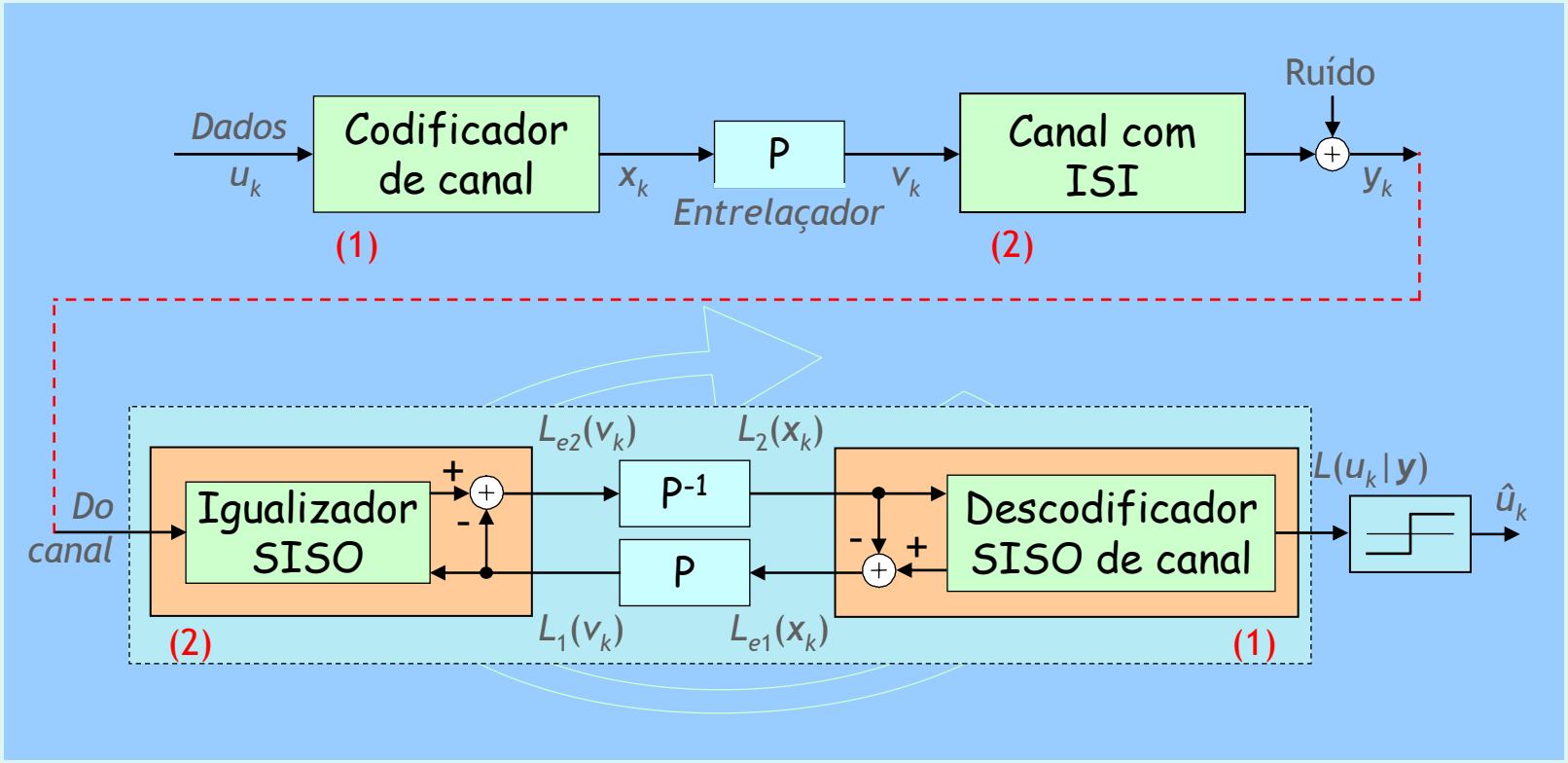
- Codificação de canal
- Igualização e estimação de canal
- Modulação codificada
- Codificação conjunta de canal e de fonte
- Sincronização
- Sistemas MIMO
- Detecção multi-utilizador e com espalhamento espectral
- Etc.

Configurações de aplicações turbo (concatenação em série)

Aplicação	Codific./Descodific. exterior (1)	Codific./Descodific. interior (2)
Codificação (FEC) concatenada em série	Codificador/ Descodificador (FEC)	Codificador/ Descodificador (FEC)
Igualização	Codificador/ Descodificador (FEC)	Canal/Detector (igualizador)
Modulação codificada (TCM)	Codificador/ Descodificador (FEC)	Mapeador/ Desmapeador
Codificação de fonte e de canal conjunta	Codific./Descodific. de fonte	Codificador/ Descodificador (FEC)
MIMO	Codificador/ Descodificador (FEC)	Mapeador e Detector MIMO

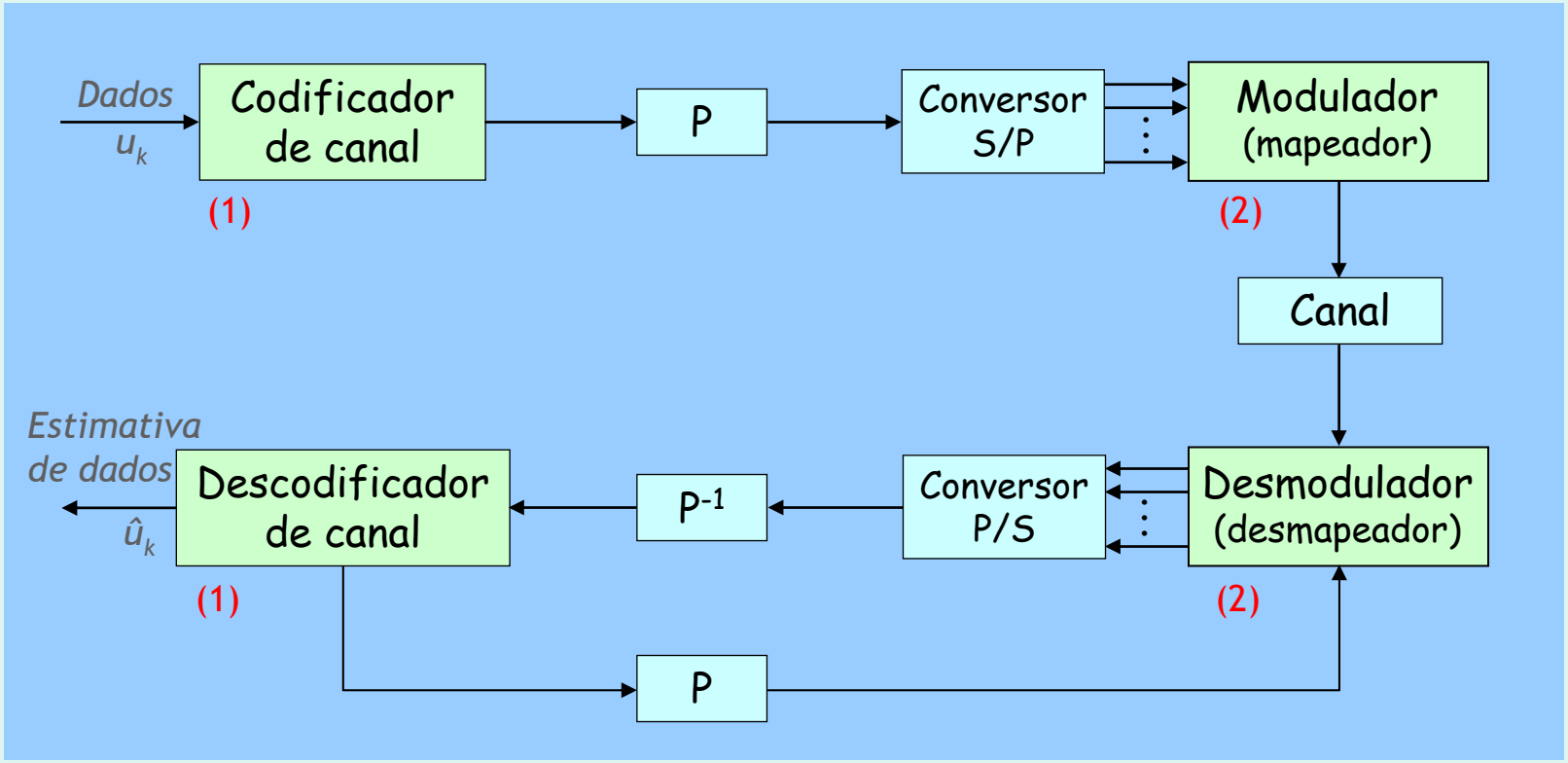
Fonte: J. Hagenauer, "The Turbo Principle in Wireless Communications", *Nordic Radio Symposium*, Oulu, Finlândia, Agosto 2004

Igualização turbo



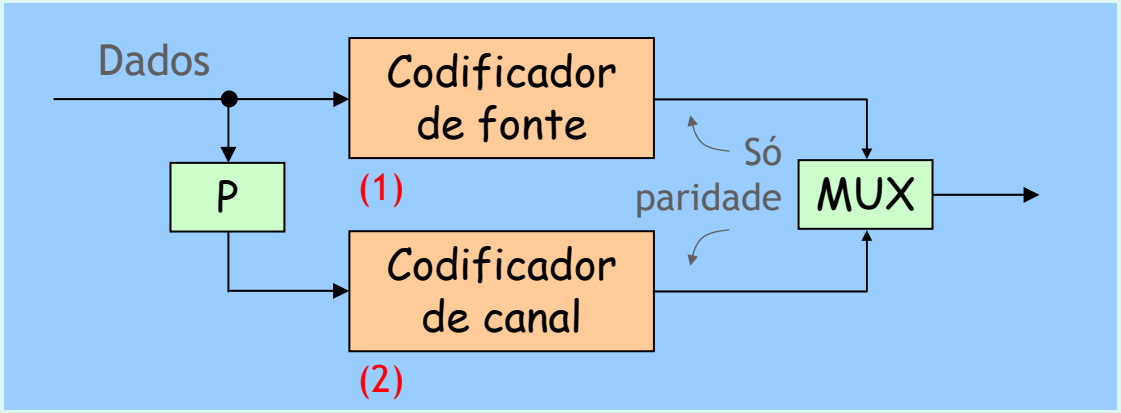
- Canal ISI pode ser encarado como um codificador de taxa 1 no corpo dos reais
- Igualizador tem duas entradas: sinal recebido do canal e $L_1(v_k)$
- Descodificador só tem uma entrada: $L_2(x_k)$

Modulação codificada turbo

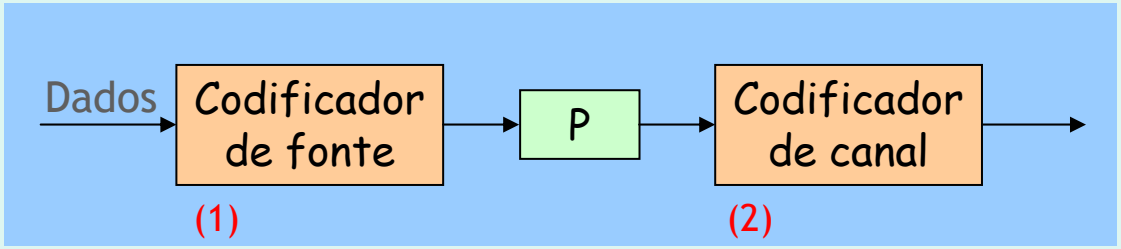


Codificação turbo conjunta de fonte e de canal

- Em paralelo

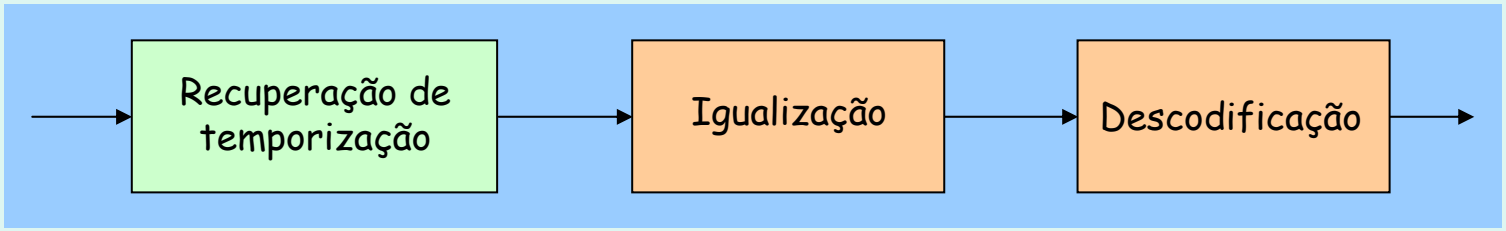


- Em série

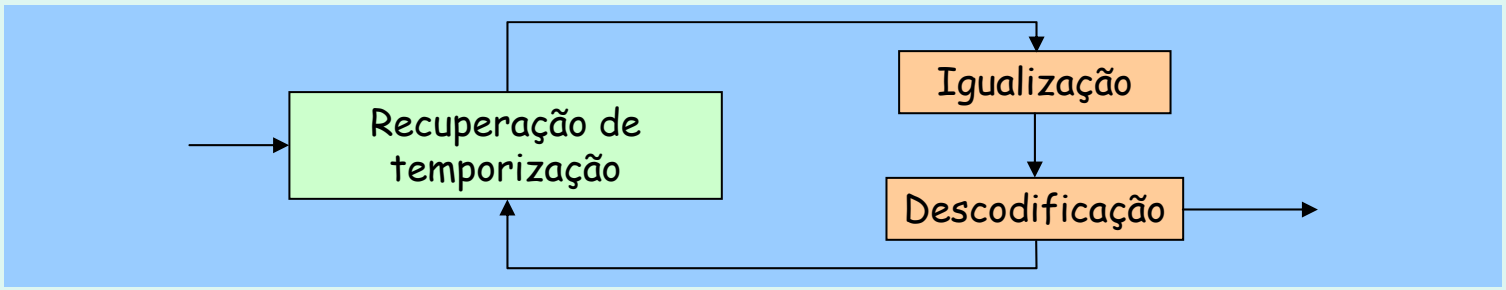


Recuperação de temporização turbo

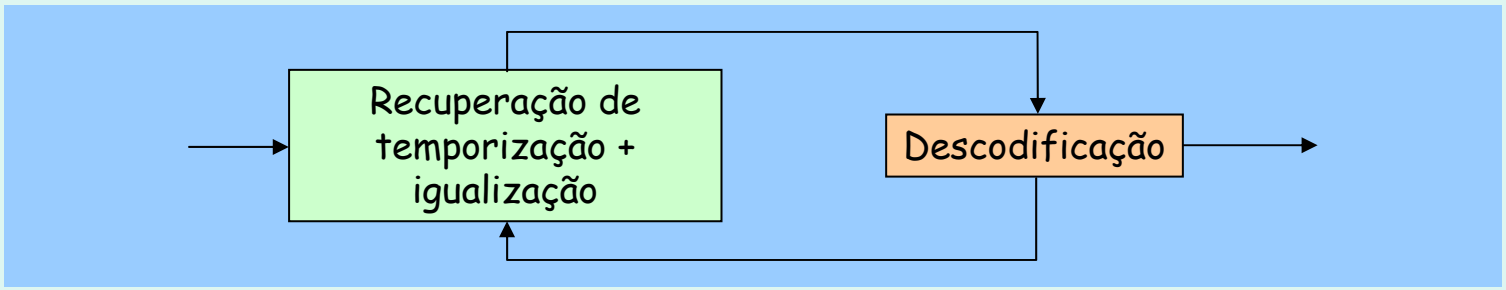
Abordagem convencional



Sincronização iterativa com recuperação e igualização separadas

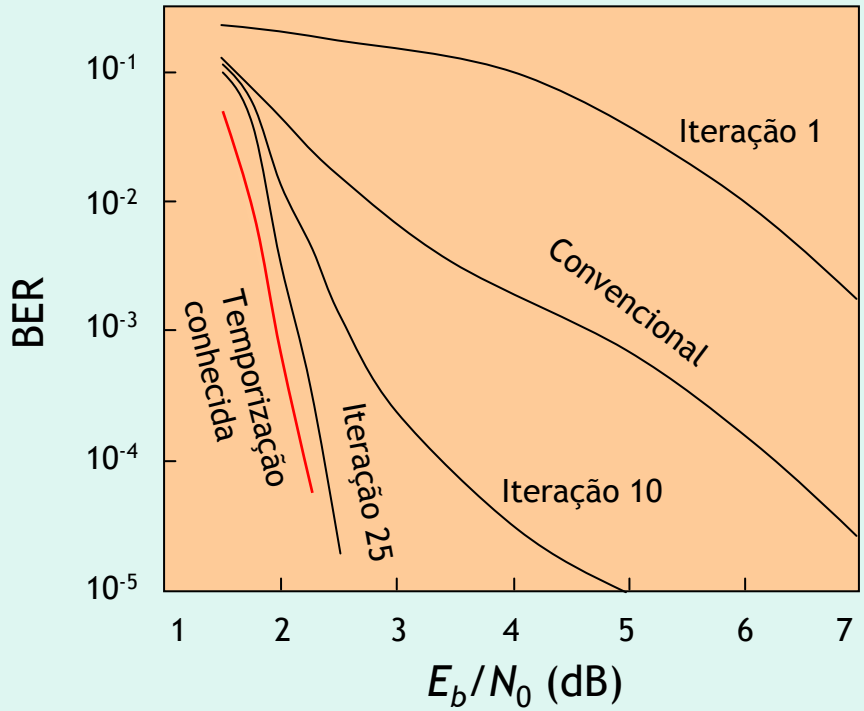


Sincronização iterativa com recuperação e igualização conjuntas



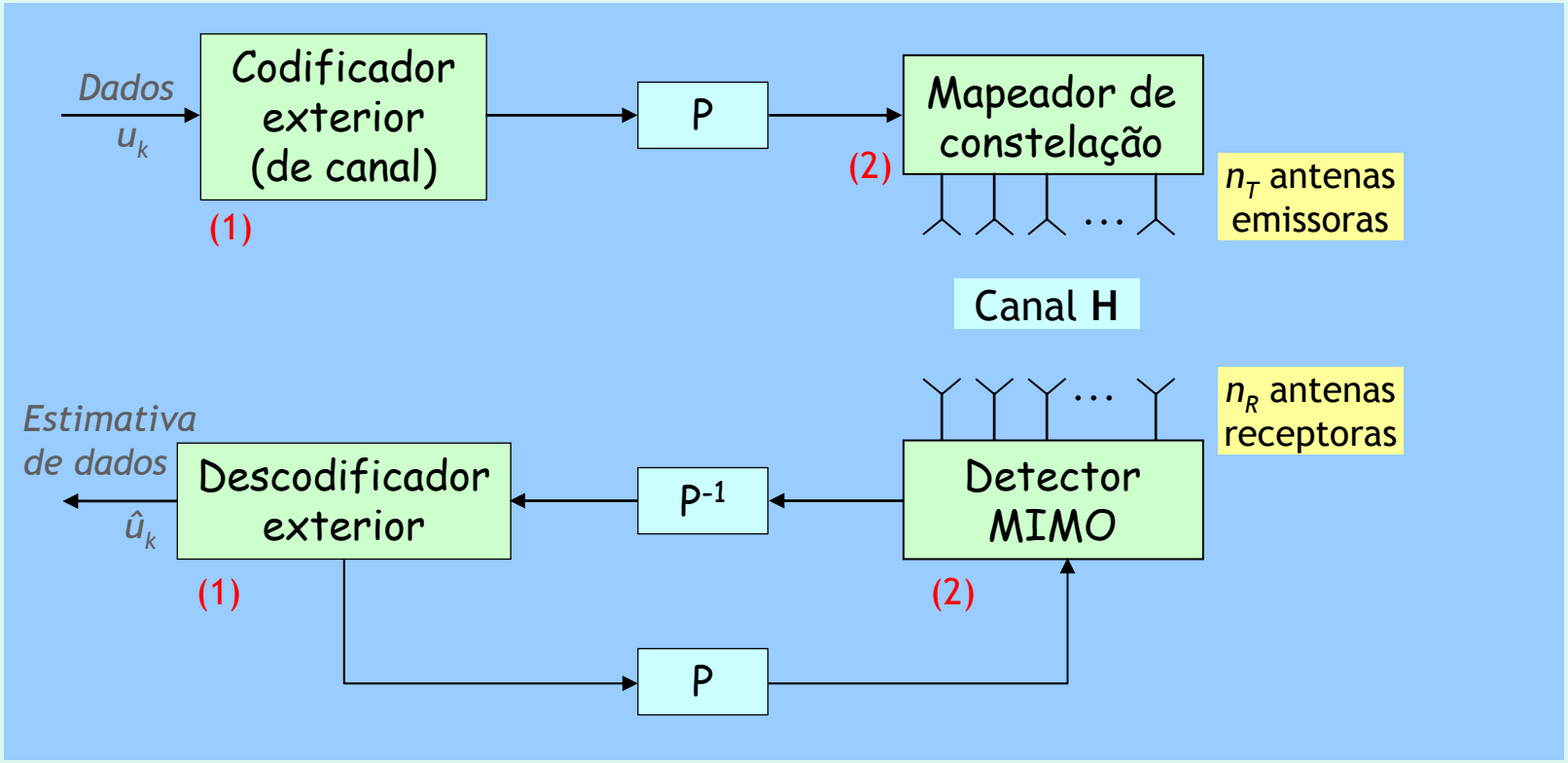
Recuperação de temporização turbo

Desempenho em canal PR4

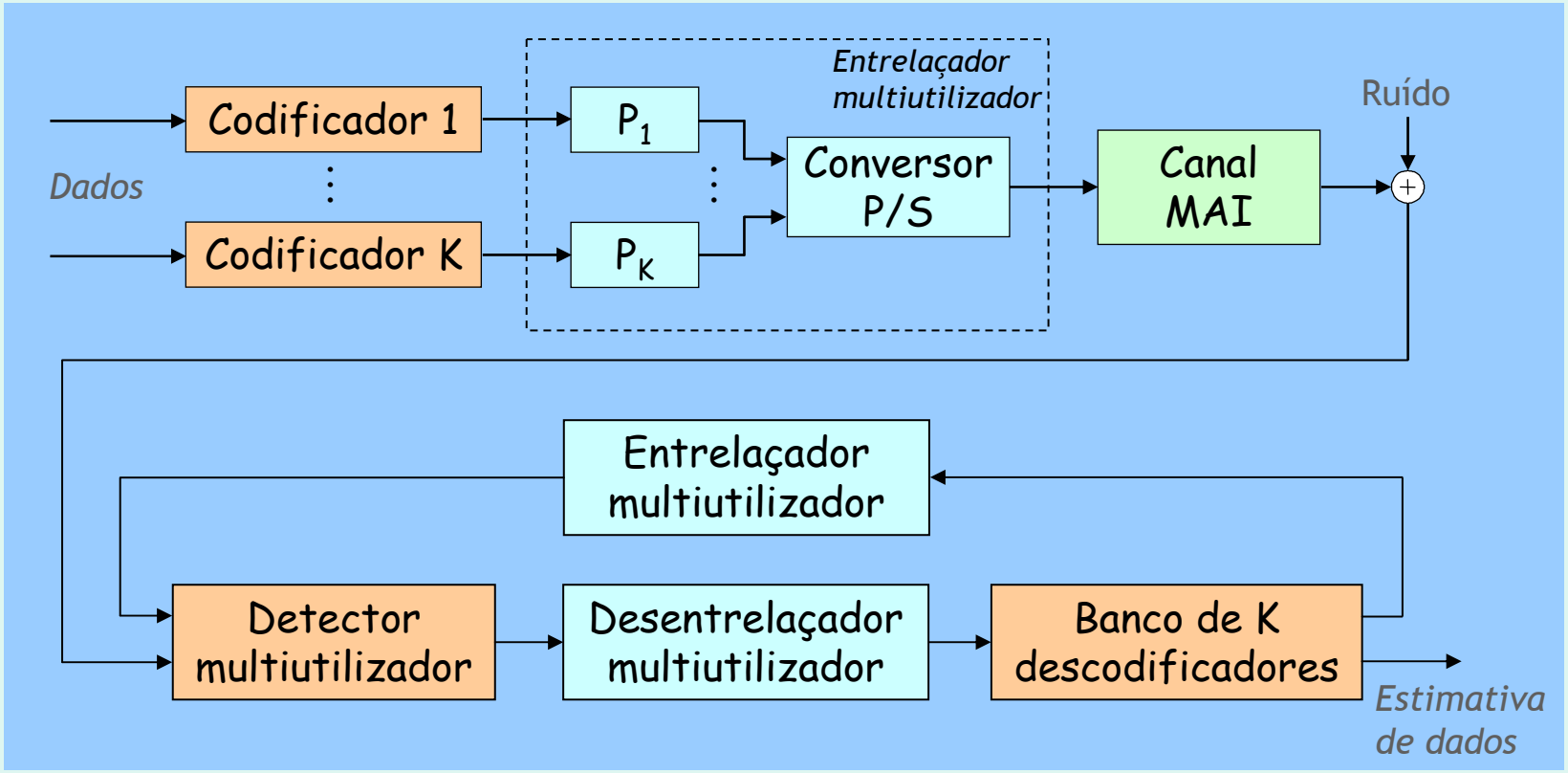


Fonte: Barry, J. R., Kavcic, A., LcLaughlin, S. W., Nayak, A., Zeng, W., "Iterative timing recovery", *IEEE Signal Processing Magazine*, Vol. 21, Jan. 2004, pp. 89-102

Sistemas MIMO (múltiplas antenas)



Detecção multi-utilizador turbo



- Canal MAI (Interferência de Acesso Múltiplo) pode ser encarado como um canal ISI de coeficientes variáveis no tempo
- Também pode ser encarado como codificador de taxa 1 no corpo dos reais

Uma “turbo” consequência

- A invenção dos turbo-códigos levou ao ressurgimento de um outro código que também se aproxima da capacidade:

LDPC

(Low-Density Parity-Check Code)

- Esteve adormecido mais de 30 anos...

Mas isso já é outra história!