Turbo codes (convolutifs)

Timisoara

18-20 mars, 2003

Michel Jézéquel,

Claude Berrou et Catherine Douillard ENST Bretagne

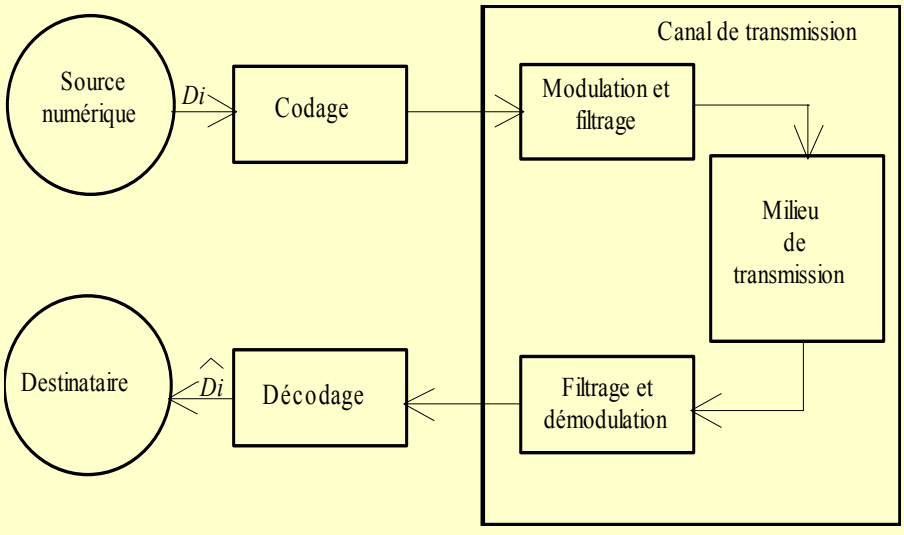
Michel.Jezequel@enst-bretagne.fr

Plan

Chapitre 1

Les codes correcteurs d'erreurs

Une chaîne de transmission



Codes correcteur d'erreurs Généralités

Différentes familles de codage

Codage de source : réduire le débit d'information
Codage de canal : ajouter de la redondance
Sécurité, cryptographie
Authentification, *watermarking*
CDMA
...

Codes correcteur d'erreurs Généralités

Séquence d'information à transmettre : $[d_1\ d_2\ \dots\ d_i\ \dots\ d_k]$

Addition d'une redondance linéaire : $[r_1\ r_2\ \dots\ r_j\ \dots\ r_{n-k}]$
(le code est dit systématique)

$$r_j = \sum_{i=1\dots k} p_{i,j} d_i \quad \text{mod } 2 \qquad (p_{i,j} = 0 \text{ ou } 1)$$

Forme matricielle

$$[r_1\ r_2\ \dots\ r_j\ \dots\ r_{n-k}] = [d_1\ d_2\ \dots\ d_i\ \dots\ d_k] \begin{pmatrix} p_{1,1} & \cdots & p_{1,n-k} \\ \vdots & & \vdots \\ p_{k,1} & \cdots & p_{k,n-k} \end{pmatrix}$$

Codes correcteur d'erreurs — Généralités

$[d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k][r_1\ r_2\ \ldots\ r_j\ \ldots\ r_{n-k}]$: mot de code

Rendement de codage : $R = k/n$

Code $(n\ ,\ k\ ,\ d_{min})$
de distance minimale d_{min}

Le code étant linéaire, la séquence « *tout zéro* »
peut être prise comme référence

La distance d_{min} est le nombre minimum de « 1 »
des mots de code $[d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k]\ [r_1\ r_2\ \ldots\ r_j\ \ldots\ r_{n-k}]$
si $[d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k] \neq$ « *tout zéro* »

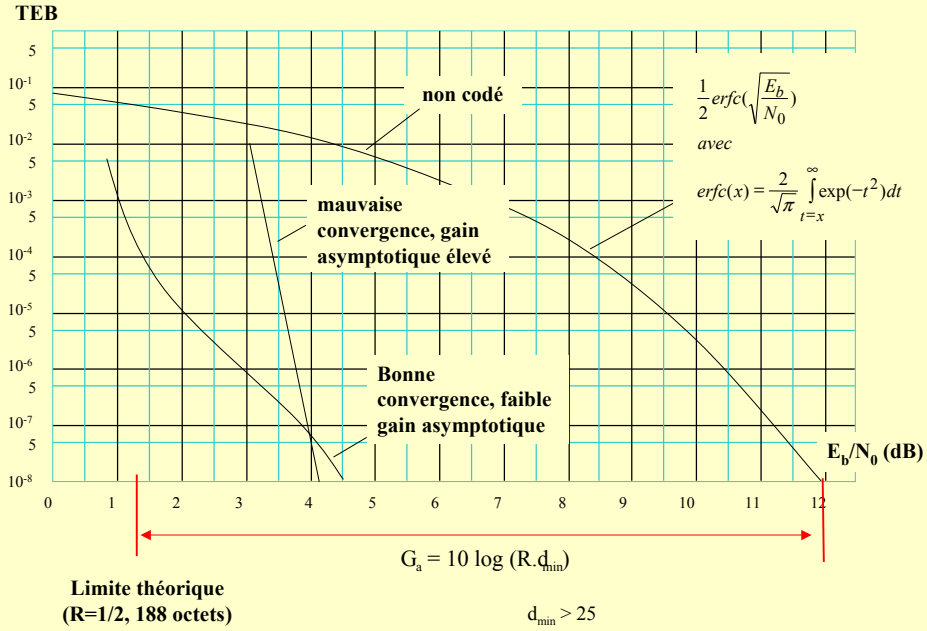
Codes correcteur d'erreurs — Généralités

Un bon code est un code avec une grande distance d_{min}
mais ce n'est pas tout :

- il doit pouvoir être décodé !
  (contre-exemple : les codes aléatoires)
- il doit exister un décodeur permettant d'utiliser
  la capacité de correction du code

Exemple : ↘

MDP4, canal gaussien, taux d'erreurs cible : 10^{-8}



Codes correcteur d'erreurs — Généralités

distinction traditionnelle mais inappropriée entre

Les codes en « bloc » : Hamming, Golay, BCH, Reed-Solomon, ...

Les codes « convolutifs »

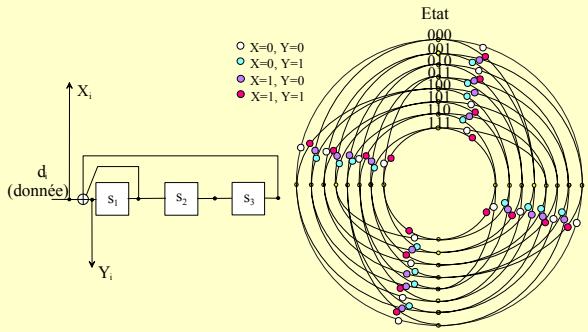
Une distinction plus naturelle prendrait en compte l'algorithme de décodage, en particulier :

- hard – dur (algébrique)
- soft – souple (probabiliste)

Exemple : ↘

Codes correcteur d'erreurs — Généralités

Comment transformer un code de Hamming parfait (8,4,4)



d_i	Hamming étendu	Y_i	d_i	Hamming étendu	Y_i
0000	0000	**0000**	1111	1111	**1111**
0001	0111	**1011**	1110	1000	**0100**
0010	1101	**0111**	1101	0010	**1000**
0011	1010	**1100**	1100	0101	**0011**
0100	1110	**1110**	1011	0001	**0001**
0101	1001	**0101**	1010	0110	**1010**
0110	0011	**1001**	1001	1100	**0110**
0111	0100	**0010**	1000	1011	**1101**

(autre exemple : le code de Golay étendu (24, 12, 8) peut être représenté par un treillis circulaire 16 états)

Codes correcteur d'erreurs — Généralités

Code aléatoire (Shannon)

Information					Redondance				
1	....	i	...	k	1	....	j	...	n- k
10000000	...	00000000000000			0010010111...0110000100				
01000000	...	00000000000000			1001110100...0011101001				
00100000	...	00000000000000			1111010001...0001111011				
.....									
00000000	...	00000000000001			1010101110...1010111101				
Σ									
10010110	...	10001100010101			0011101011...1110010100				

$d_{min} \approx (n-k)/4 = k.(1-R)/4R$ où $R = k/n$

Codes correcteur d'erreurs — Généralités

$d_{min} \approx (n-k)/4 = k.(1-R)/4R$ où $R = k/n$

Exemple : $k = 1504$ (MPEG), $R = 1/2$

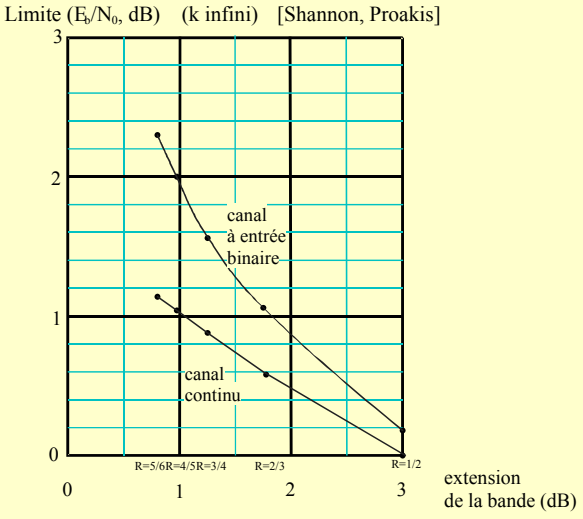
$\Longrightarrow$ $d_{min} \approx 375$!!

(un code convolutif 64 états
a une distance $d_{min} = 10$ pour $R = 1/2$!!)

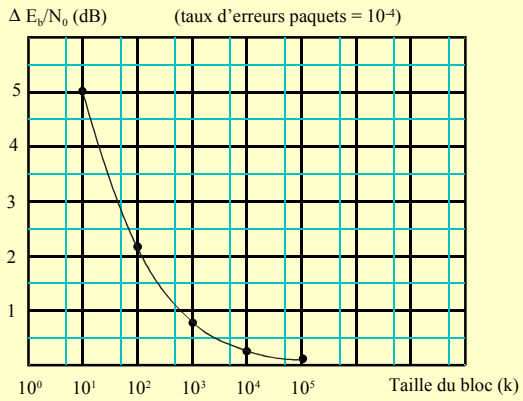
Mais non décodable en pratique ...

Codes correcteur d'erreurs — Généralités

(sur la base d'un code aléatoire)



Codes correcteur d'erreurs — Généralités



Correction à prendre en compte pour des bocks de taille *k* :
[S. Dolinar, D. Divsalar and F. Pollara, "Code performance as a function of block size", TMO progress report 42-133, JPL, NASA].

Calcul des limites

Un outil de calcul des limites théoriques a été développé par un étudiant en thèse il est disponible à l'adresse :

http://www-elec.enst-bretagne.fr/~emaury/Capa_UI.html

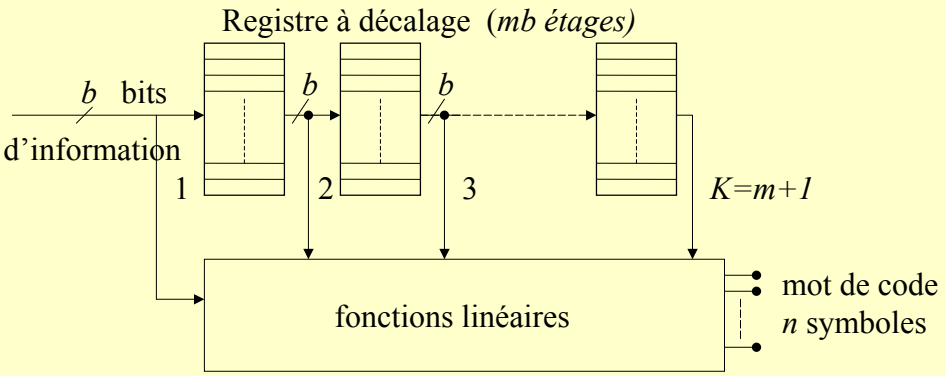
Chapitre 2

Codes convolutifs

Plan

- Codes convolutifs classiques
  - Représentations
  - Propriétés
- Codes convolutifs systématiques
- Codes convolutifs récursifs systématiques
- Rendement de codage et poinçonnage
- Fermeture de treillis

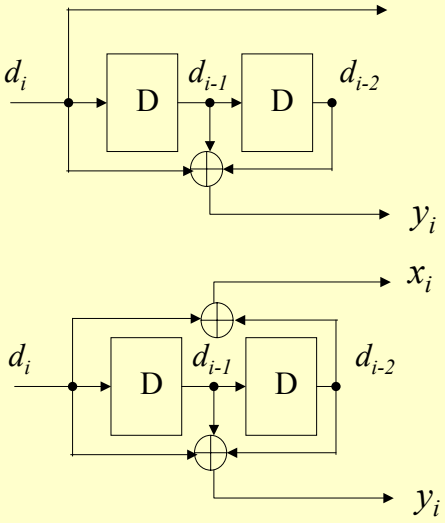
Codes convolutifs



K= longueur de contrainte

R=b/n (rendement de codage)

Code convolutif classique



Elias, 1954

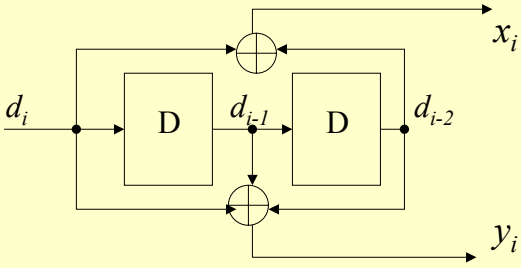
Mémoire du code : ν = 2

Longueur de contrainte : K = ν + 1 = 3

R = 1/2

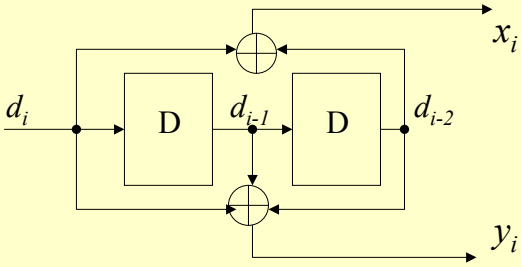
Forney, 1970

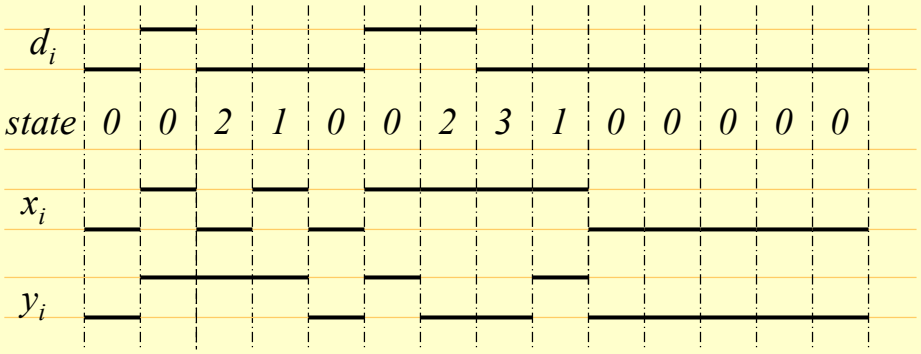
Exemple



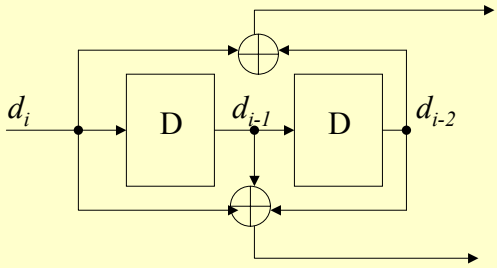
- K=3
- b=1
- n=2
- $R=b/n$=1/2

Exemple





Générateurs



$$g^1=[g_0^1,g_1^1,g_2^1]=[1,0,1] \longrightarrow G^1(D)=g_0^1+g_1^1D+g_2^1D^2$$

$$g^2=[g_0^2,g_1^2,g_2^2]=[1,1,1] \longrightarrow G^2(D)=g_0^2+g_1^2D+g_2^2D^2$$

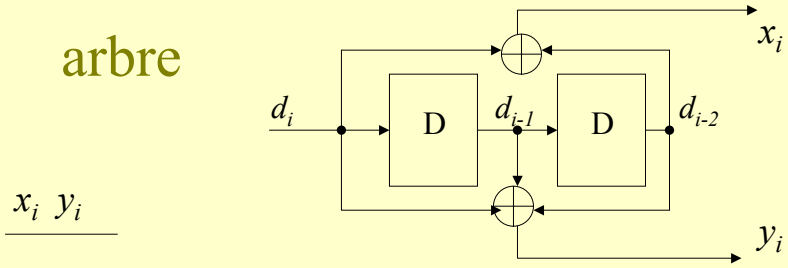
Générateurs sous la forme octale : (5,7)

Représentation des codes convolutifs

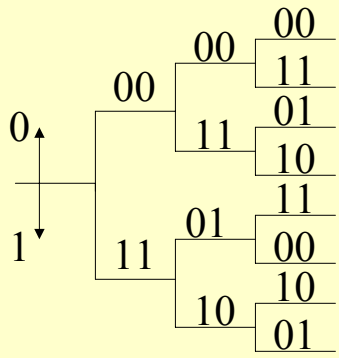
Trois formes :

- arbre
- machine d'état
- treillis

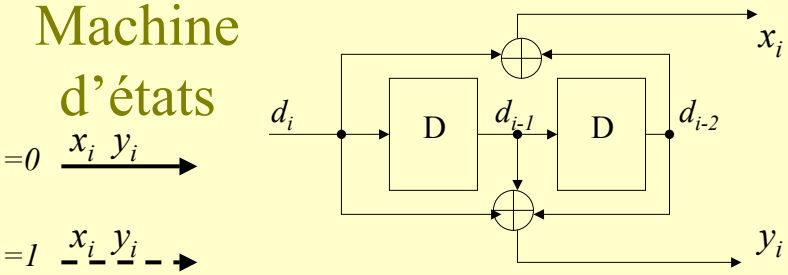
arbre



x_i y_i

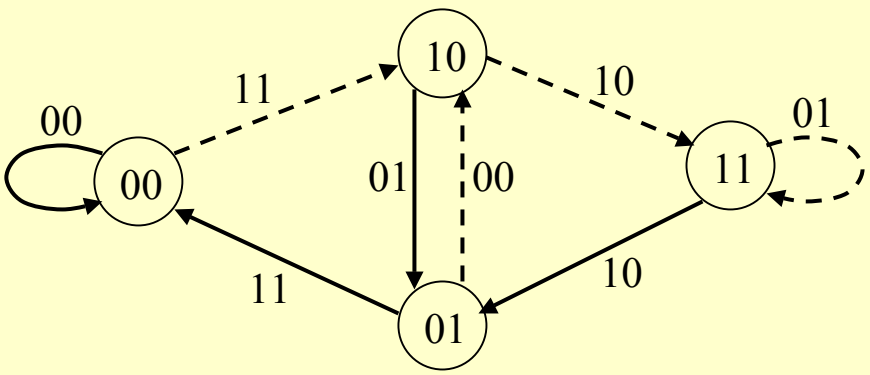


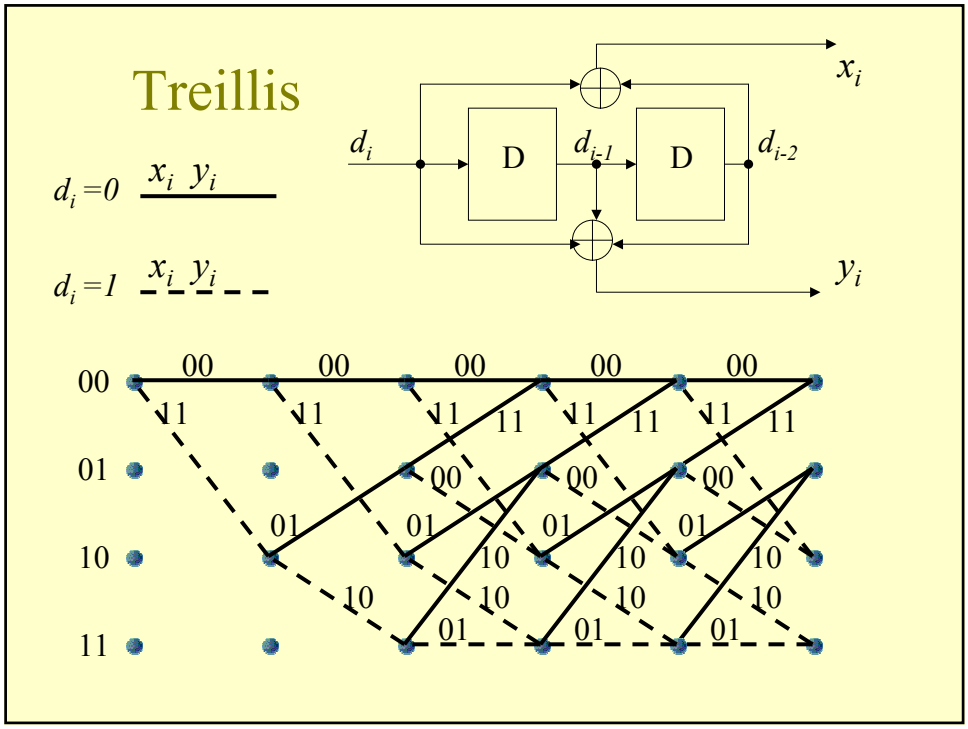
Machine d'états

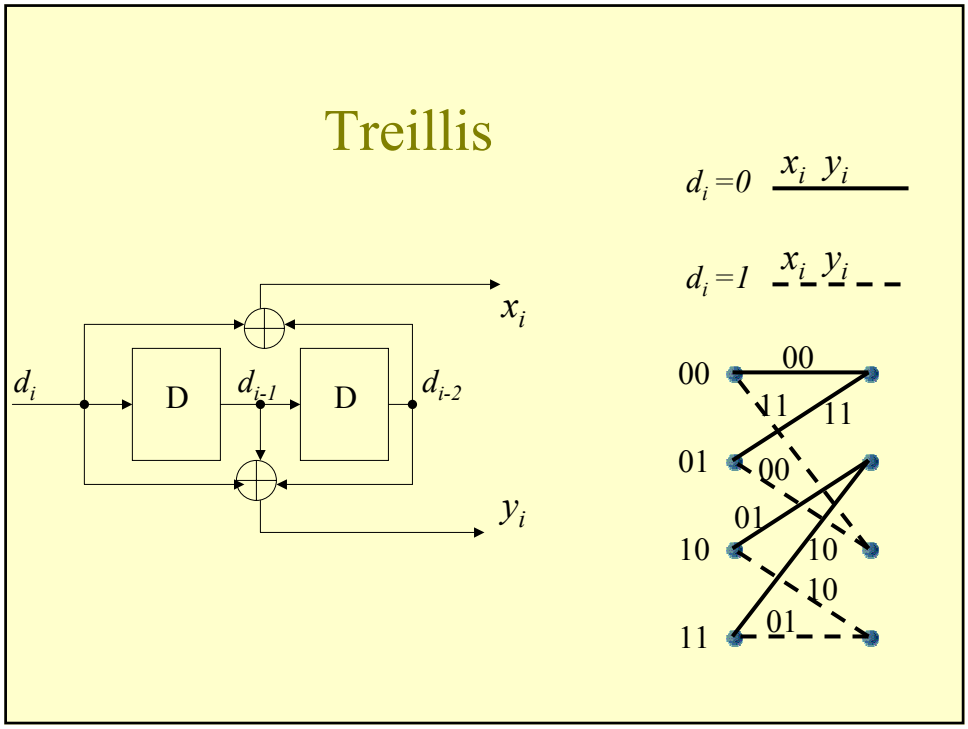


$d_i = 0$ x_i y_i

$d_i = 1$ x_i y_i

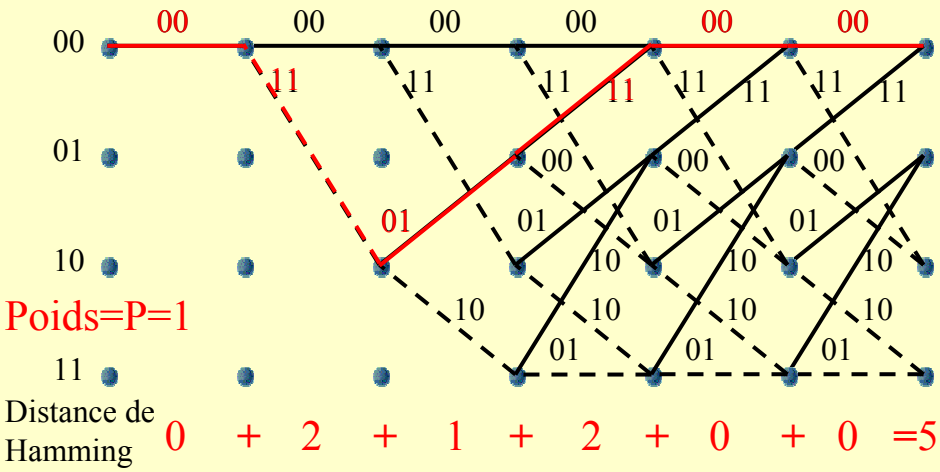



Treillis
$d_i=0$ x_i y_i
$d_i=1$ x_i y_i
d_i
D
d_{i-1}
D
d_{i-2}
x_i
y_i
00
01
10
11
00 00 00 00 00
11 11 11 11 11
00 00 00
01 01 01 01
10 10 10 10
10 10 10 10
01 01 01


Treillis
$d_i=0$ x_i y_i
$d_i=1$ x_i y_i
d_i
D
d_{i-1}
D
d_{i-2}
x_i
y_i
00
01
10
11
00
11 11
00
01 10
10
01

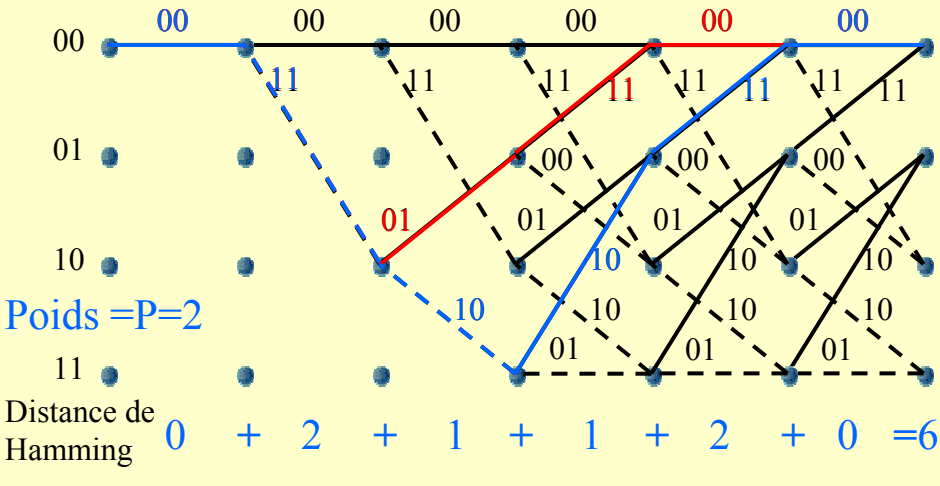
Distance

(K=3) (5,7) -> distance libre = d_{libre}=5



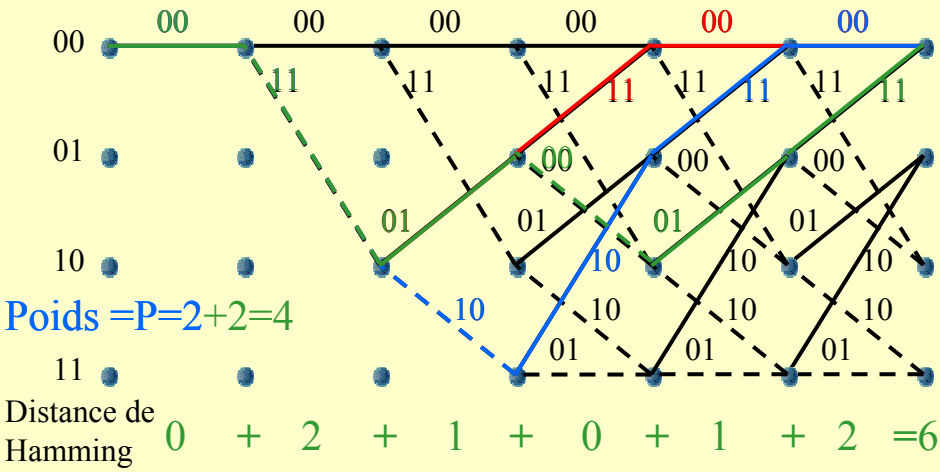
Distance

(K=3) (5,7) -> distance libre = d_{libre}=5



Distance

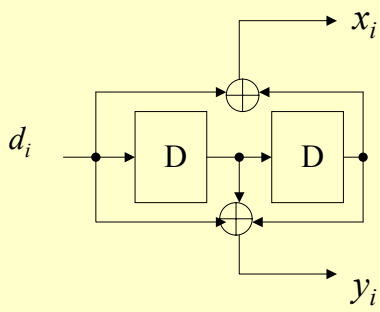
(K=3) (5,7) -> distance libre = d_{libre}=5



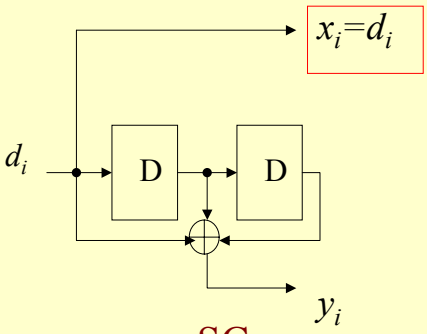
Distance

- Longueur de contrainte : K = 3
- Générateurs (5,7)
  - Distance libre minimale = 5
  - Spectre des distances:
    - Distance = 5 (1 cas), P=1
    - Distance = 6 (2 cas), P=4
    - Distance = 7 (4 cas), P=12
    - ….

Codes systématiques



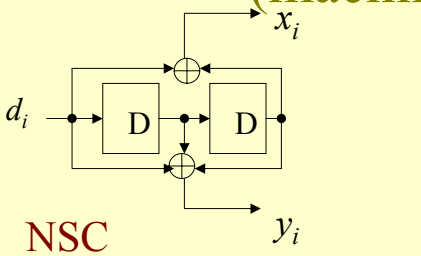
NSC
(Non Systematic Convolutional)



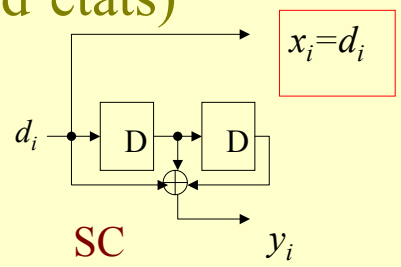
SC
(Systematic Convolutional)

Exemple : K=3, *b*=1, *n*=2, *R*=*b*/*n*=1/2

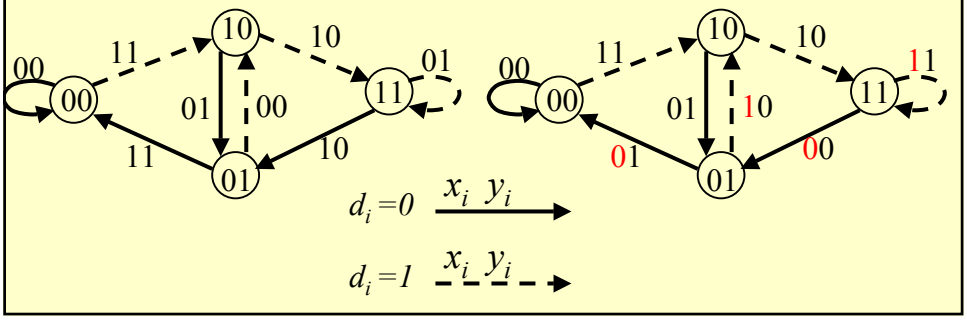
Codes systématiques (machine d'états)



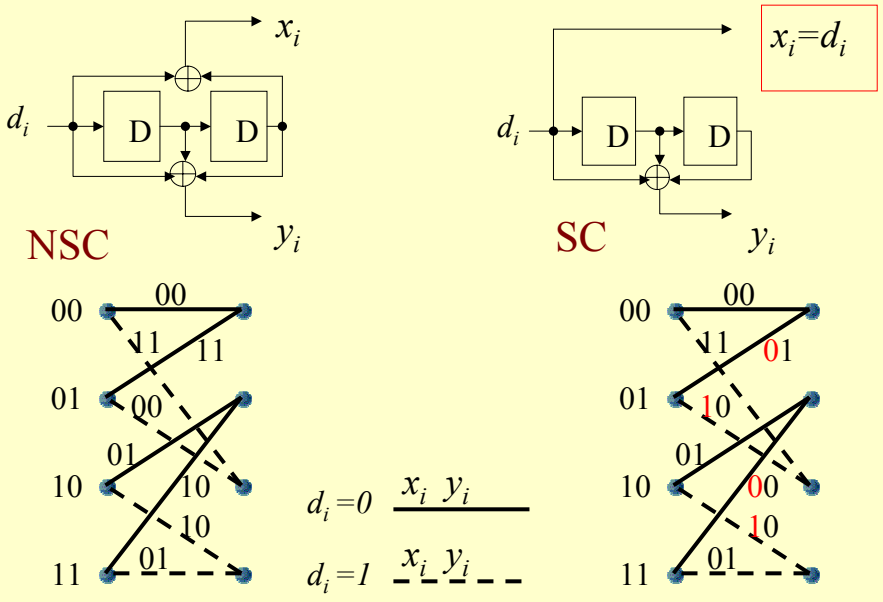
NSC



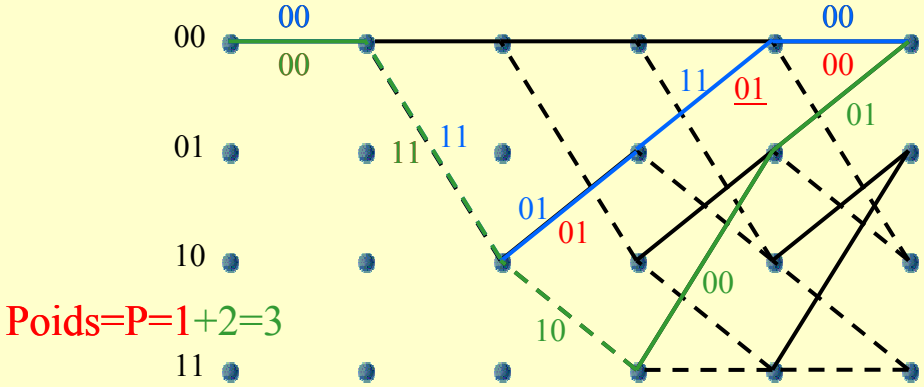
SC



Codes systématiques (treillis)



Codes systématiques(distance)



Distance libre minimale

NSC	0	+	2	+	1	+	2	+	0	=5
SC	0	+	2	+	1	+	1	+	0	=4
	0	+	2	+	1	+	0	+	1	=4

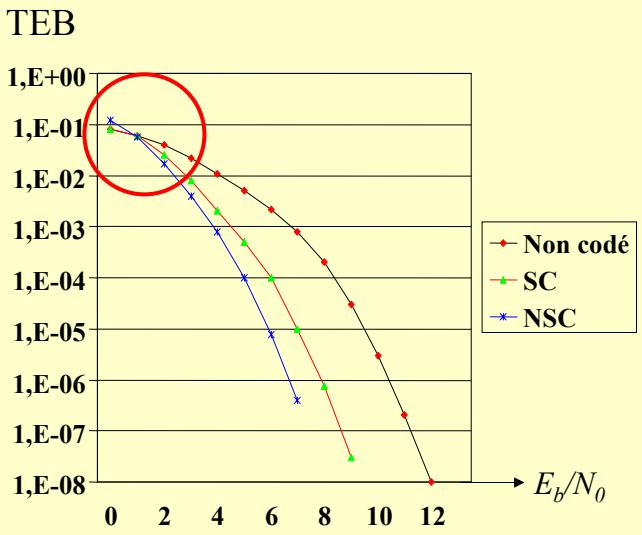
Spectre des distances

	n	P SC	n	P 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

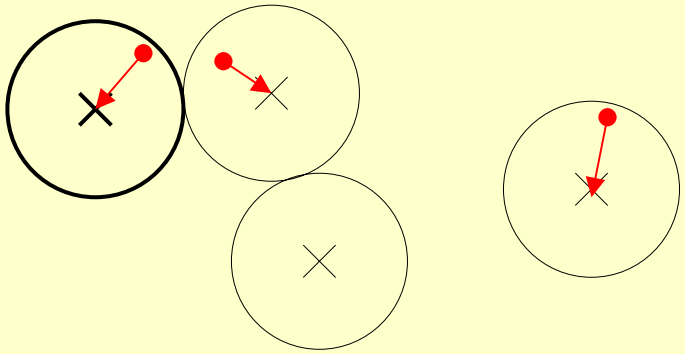
n = nombre de cas

P = poids

Codes systématiques (Taux d'Erreurs Binaire)



Interprétation des résultats



Codes convolutifs

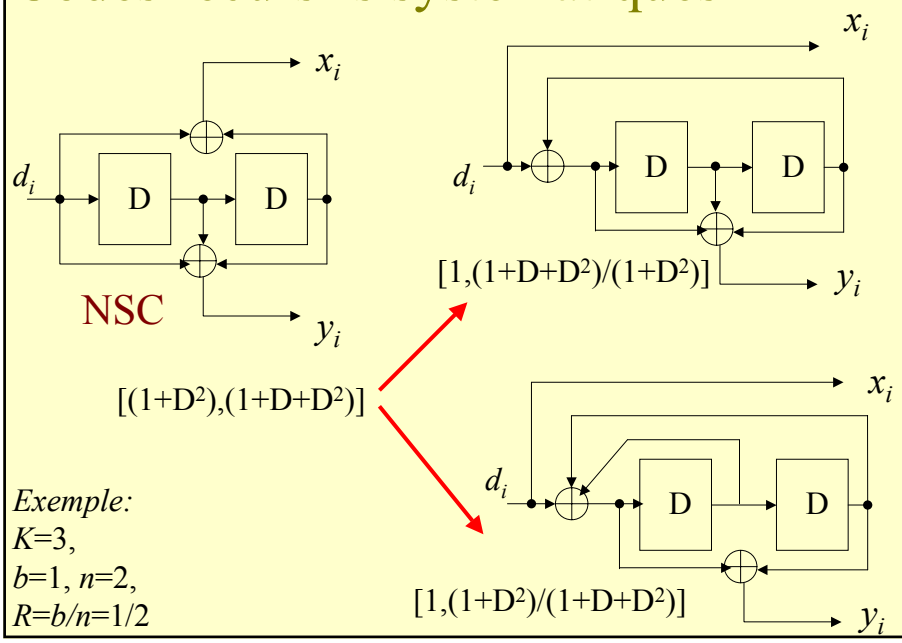
Existe-t-il un code convolutif combinant :

- les distances minimales des codes non systématiques
- le bon comportement à faible rapport signal/bruit

????

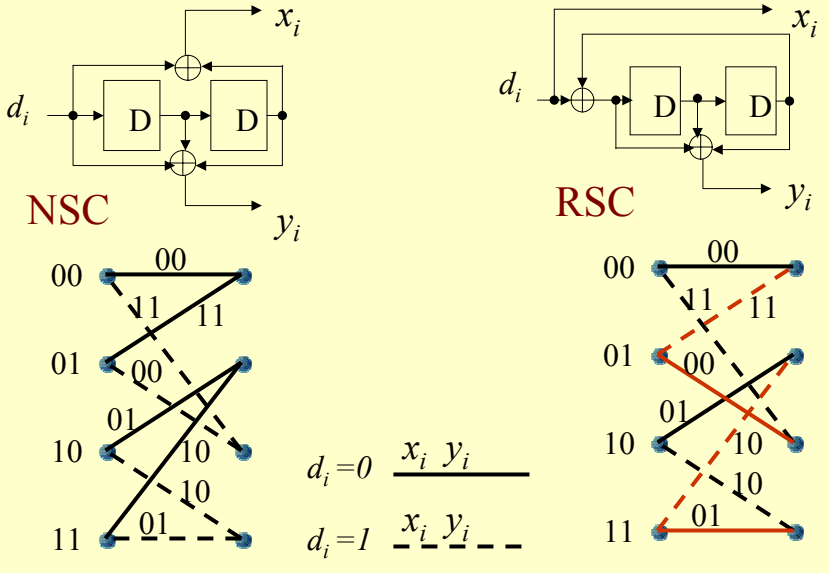
La réponse est : **OUI**

Codes récursifs systématiques

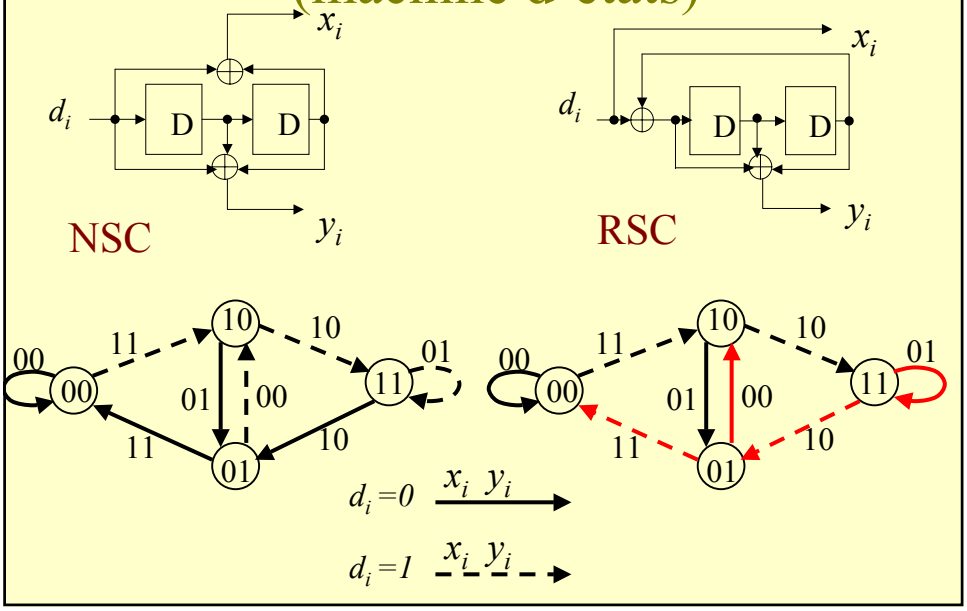


Exemple:
K=3,
b=1, *n*=2,
R=*b*/*n*=1/2

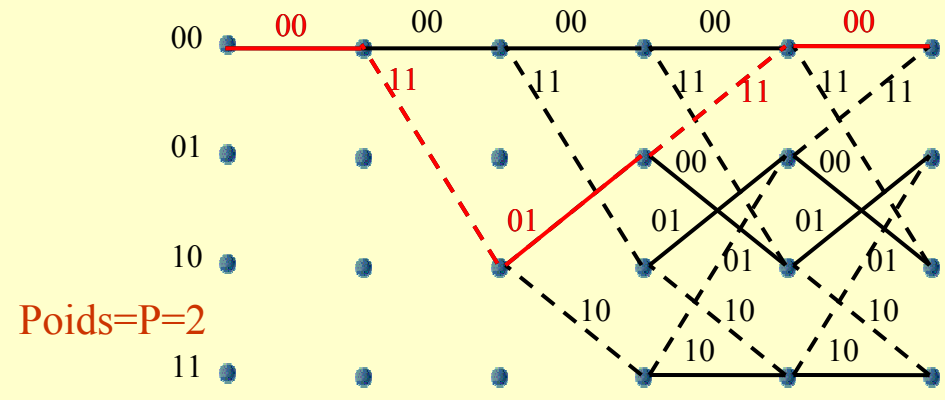
Codes récursifs systématiques (treillis)



Codes Récursifs systématiques (machine d'états)



Codes récursifs systématiques(distance)



Distance libre minimale RSC 0 + 2 + 1 + 2 + 0 =5

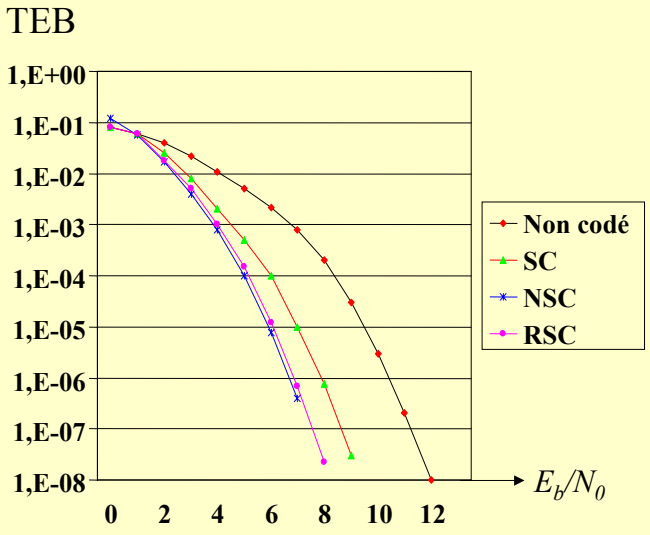
Spectre des distances

	n	P NSC	n	P RSC
d=5	1	1	1	2
d=6	2	4	2	6
d=7	4	12	4	14
d=8	8	32	8	32
d=9	16	80	16	72
d=10	32	192	32	160
d=11	64	448	64	352
d=12	128	1024	128	768
d=13	256	2304	256	1664
d=14	512	5120	512	3584
d=15	1024	11264	1024	7680

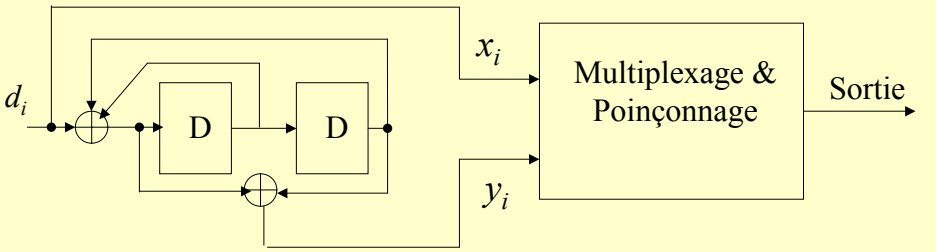
n = nombre de cas

P = poids

Codes récursifs systématiques (Taux d'Erreurs Binaires)

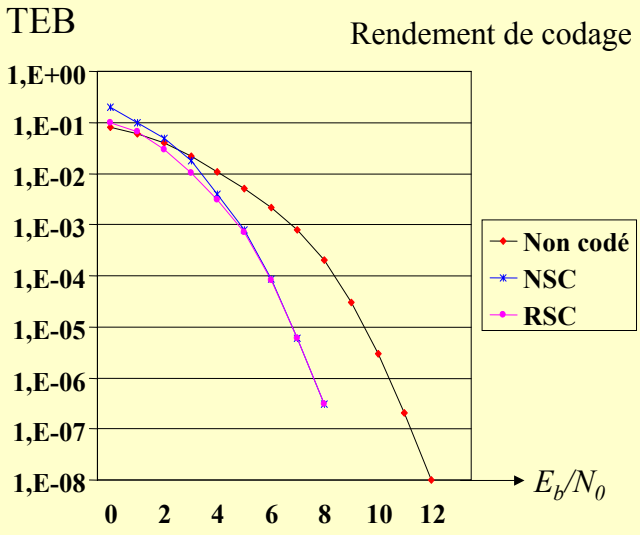


Rendement de codage et poinçonnage

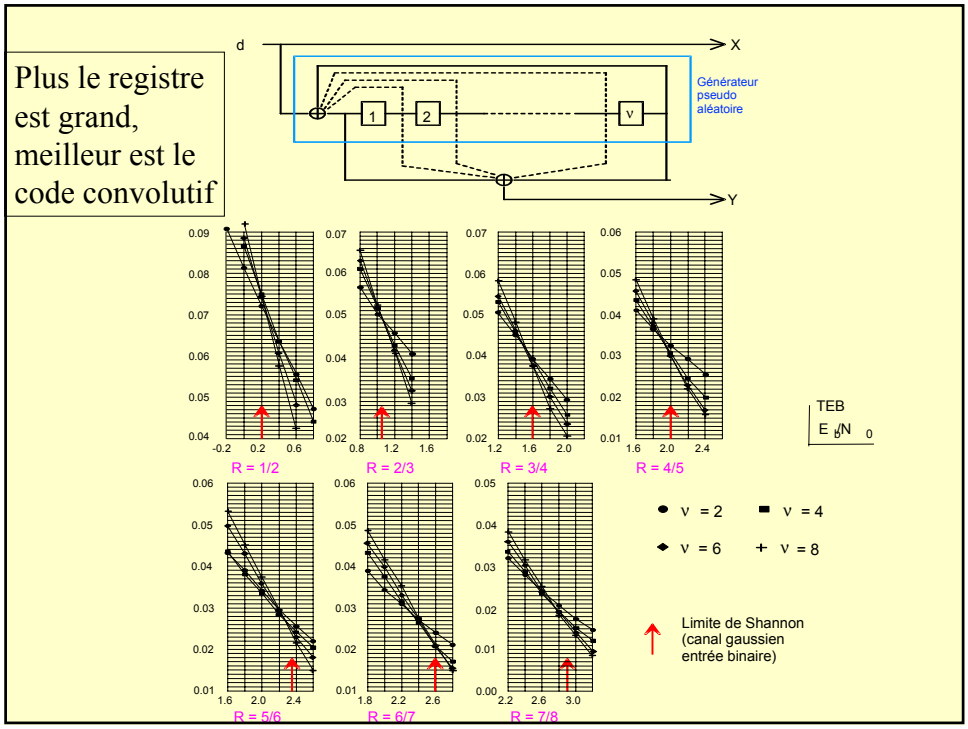


R=1/2	x_i	y_i	x_{i+1}	y_{i+1}	x_{i+2}	y_{i+2}	x_{i+3}	y_{i+3}	x_{i+4}	y_{i+4}
R=2/3	x_i	x_{i+1}	y_{i+1}	x_{i+2}	x_{i+3}	y_{i+3}	x_{i+4}	x_{i+5}	y_{i+5}	x_{i+6}
R=3/4	x_i	x_{i+1}	x_{i+2}	y_{i+2}	x_{i+3}	x_{i+4}	x_{i+5}	y_{i+5}	x_{i+6}	x_{i+7}
R=3/5	x_i	x_{i+1}	y_{i+1}	x_{i+2}	y_{i+2}	x_{i+3}	x_{i+4}	y_{i+4}	x_{i+5}	y_{i+5}

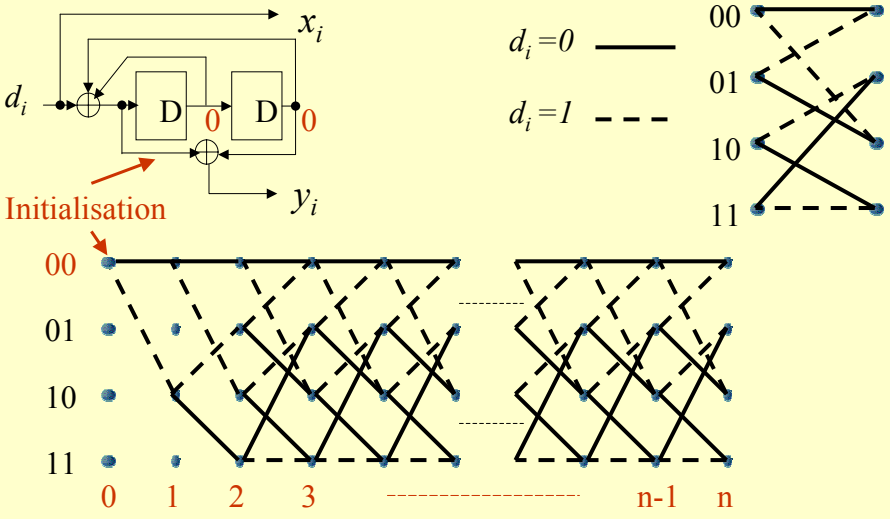
Codes poinçonnés (Taux d'Erreurs Binaire)



Plus le registre est grand, meilleur est le code convolutif

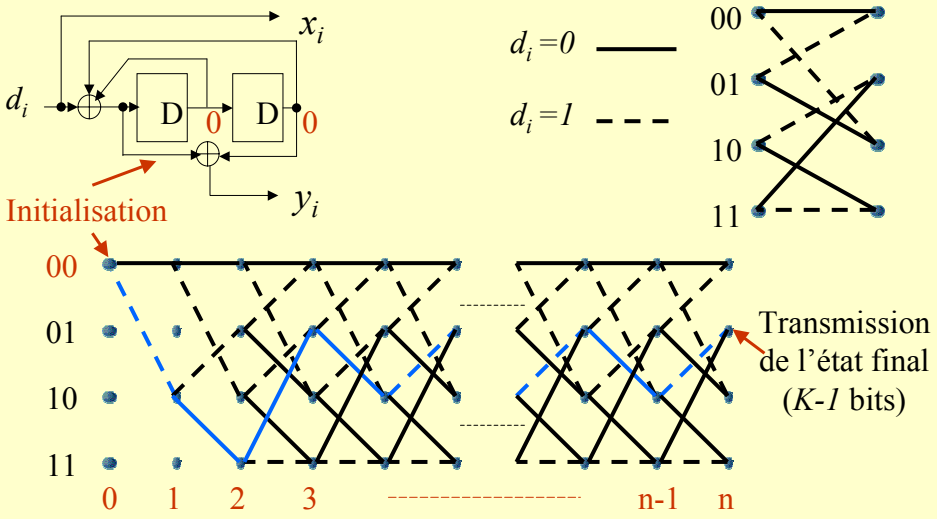


Fermeture de treillis



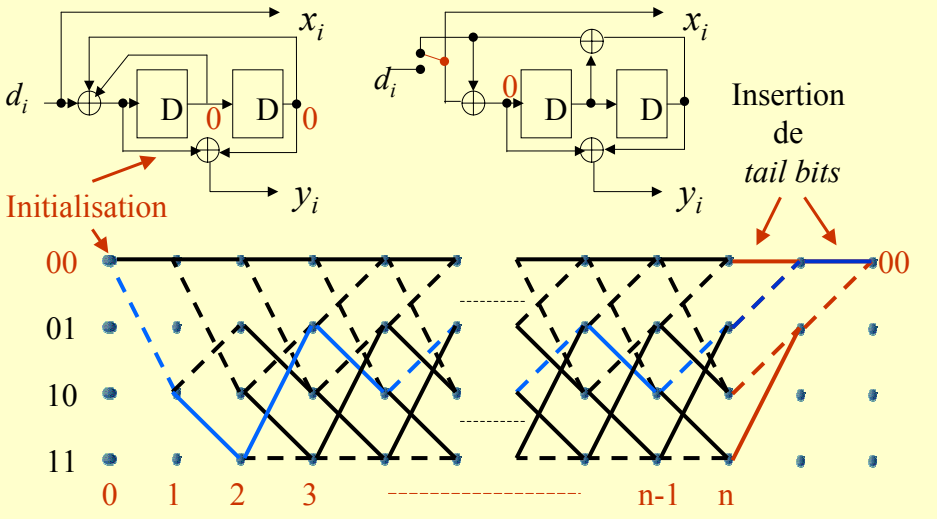
Bloc de k bits, $2k$ bits transmis → $R=1/2$

Fermeture de treillis



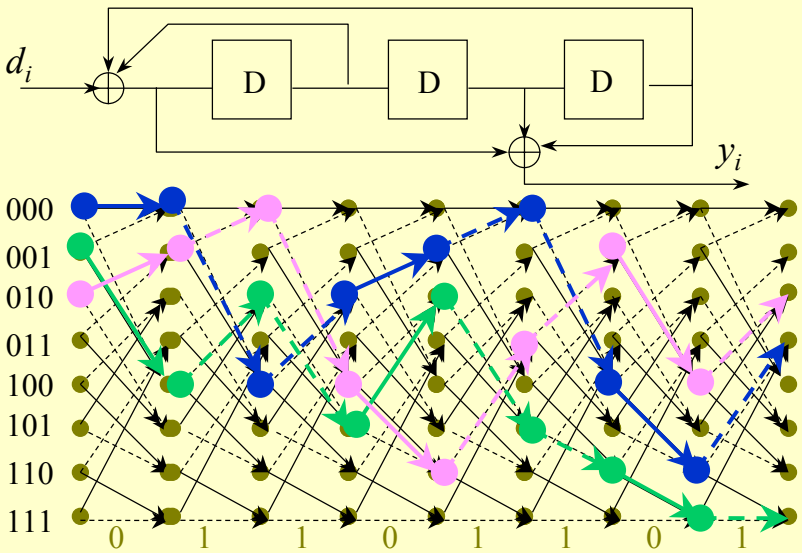
Bloc de *k* bits, *2k+K-1* bits transmis → *R=k/(2k+K-1)*

Fermeture de treillis

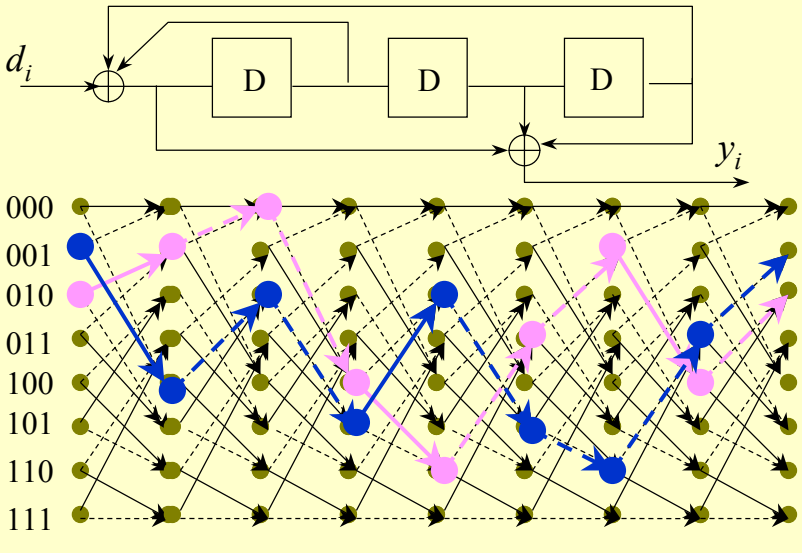


Bloc de *k* bits, *2k+2(K-1)* bits transmis → *R=k/2(k+K-1)*

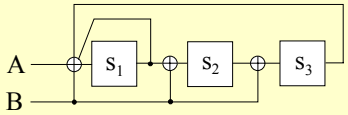
Codes convolutifs circulaires



Codes convolutifs circulaires



Calcul de l'état circulation



$$\mathbf{S}_i = \mathbf{G}.\mathbf{S}_{i-1} + \mathbf{X}_{i-1} \qquad \text{Mod 2} \qquad (1)$$

$$\mathbf{S}_i = \begin{bmatrix} s_{1,i} \\ s_{2,i} \\ s_{3,i} \end{bmatrix};\ \mathbf{X}_i = \begin{bmatrix} A_i + B_i \\ B_i \\ B_i \end{bmatrix};\ \mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

La période L du générateur est telle que $\mathbf{G}^L = \mathbf{I}$ (Identité)

En partant de (1), nous pouvons en déduire :

$\mathbf{S}_i = \mathbf{G}.\mathbf{S}_{i-1} + \mathbf{X}_{i-1}$

$\mathbf{S}_{i-1} = \mathbf{G}.\mathbf{S}_{i-2} + \mathbf{X}_{i-2}$

.....

$\mathbf{S}_1 = \mathbf{G}.\mathbf{S}_0 + \mathbf{X}_0$

Après codage d'une séquence de longueur k :

$$\mathbf{S}_k = \mathbf{G}^k.\mathbf{S}_0 + \sum_{p=1\ldots k} \mathbf{G}^{k-p}\mathbf{X}_{p-1} \qquad (2)$$

Si $\left\langle \mathbf{I} + \mathbf{G}^k \right\rangle$ est inversible, il existe un $\mathbf{S}_c$ tel que $\mathbf{S}_c = \mathbf{S}_k = \mathbf{S}_0$.
$\mathbf{S}_c$ est appelé état de circulation.

$$\mathbf{S}_c = \left\langle \mathbf{I} + \mathbf{G}^k \right\rangle^{-1} . \sum_{p=1\ldots k} \mathbf{G}^{k-p}\mathbf{X}_{p-1} \qquad (3)$$

En pratique :

Le codeur est initialisé à $\mathbf{S}_0 = \mathbf{0}$; la séquence de longueur k est codée donnant, d'après (2) :

$$\mathbf{S}_k{}^0 = \sum_{p=1\ldots k} \mathbf{G}^{k-p}\mathbf{X}_{p-1}$$

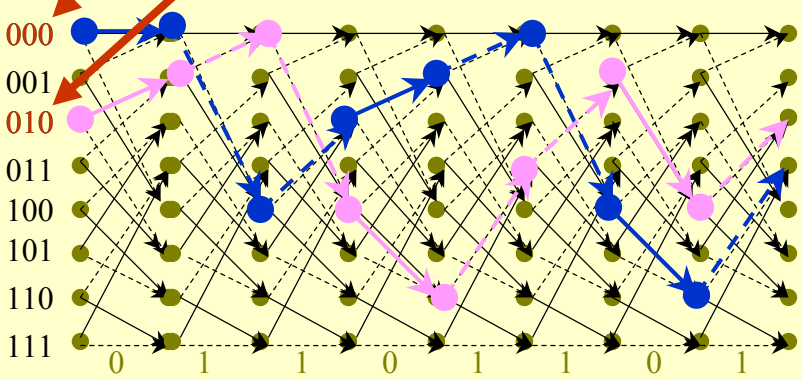
(3) devient :

$$\mathbf{S}_c = \left\langle \mathbf{I} + \mathbf{G}^k \right\rangle^{-1} . \mathbf{S}_k{}^0 \qquad (4)$$

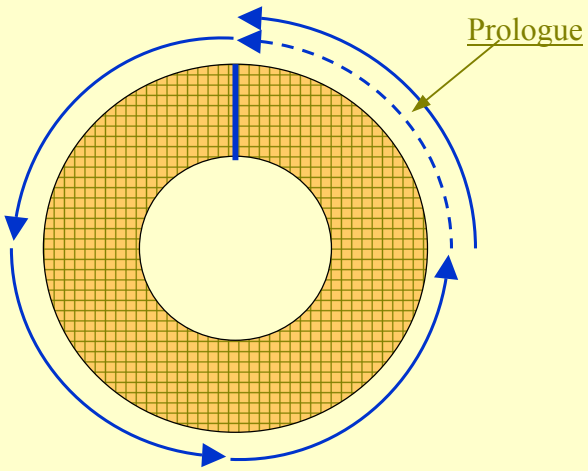
La correspondance entre $\mathbf{S}$ et $\mathbf{S}_k{}^0$ est mémorisée dans une petite fonction combinatoire (3 bits en entrée, 3 bits en sortie).

Codes convolutifs circulaires

- Pré-codage :
  - Initialisation à l'état *tout zéro*
  - Codage de la séquence d'information
  - Détermination de l'état de circulation
- Initialisation à l'état de circulation
- Codage



Codes convolutifs circulaires
(comment décoder ?)



Fermeture de treillis

- “tail bits”
  - ☹ rendement de codage
  - ☺ facile à implémenter
- codes circulaires (“tail biting” NSC)
  - ☹ pré-codage
  - ☺ rendement de codage

Chapitre 3

Algorithmes de décodage à entrée et sortie pondérées ou SISO (Soft-In Soft-Out)

Codes convolutifs — algorithmes SISO

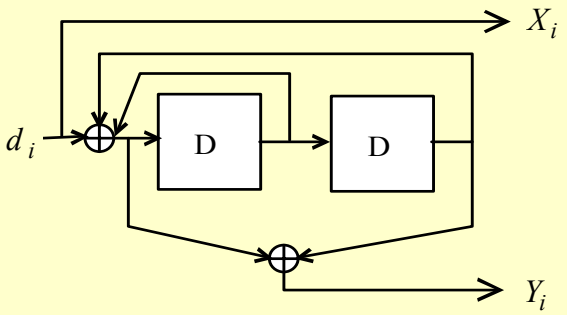
Le turbo-décodage utilise des décodeurs élémentaires à entrées et sorties pondérées.

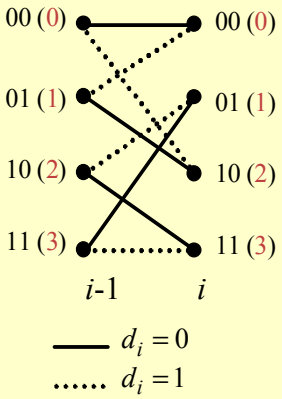
Deux familles d'algorithmes de décodage SISO pour les codes convolutifs :

- Les algorithmes basés sur l'algorithme de Viterbi
- L'algorithme MAP *(Maximum A Posteriori)* et ses approximations

Codes convolutifs algorithmes SISO

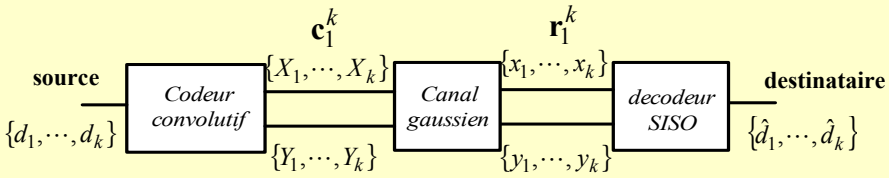
Illustration avec le code suivant :





Codes convolutifs algorithmes SISO

Modèle de la chaîne de transmission



Les symboles transmis X_i et Y_i, $i = 1\ldots k$ prennent leurs valeurs dans {-1,+1}

$\mathbf{c}_1^k$: séquence codée et modulée (MDP2)

$\mathbf{c}_1^k = \{c_1,\cdots,c_k\}$ avec $c_i = (X_i, Y_i)$

$\mathbf{r}_1^k$: séquence bruitée reçue

$\mathbf{r}_1^k = \{r_1,\cdots,r_k\}$ avec $r_i = (x_i, y_i)$, $\quad x_i = X_i + n_i^x$, $\; y_i = Y_i + n_i^y$

Codes convolutifs algorithmes SISO : SOVA

Application de l'algorithme de Viterbi

- L'algorithme de Viterbi est utilisé pour chercher le chemin à vraisemblance maximale dans le diagramme en treillis.
- Critère de décodage : maximiser $\Pr\{\mathbf{R}_1^k|\mathbf{C}_1^k\}$

$$\Pr\{\mathbf{R}_1^k|\mathbf{C}_1^k\} = \prod_{i=1}^{k} \Pr\{R_i|C_i\}$$

$$= \prod_{i=1}^{k} \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^2 \exp\left(-\frac{(x_i - X_i)^2}{2\sigma^2}\right) \exp\left(-\frac{(y_i - Y_i)^2}{2\sigma^2}\right)$$

- Maximiser $\Pr\{\mathbf{R}_1^k|\mathbf{C}_1^k\} \Leftrightarrow$ minimiser $\sum_{i=1}^{k} \left[(x_i - X_i)^2 + (y_i - Y_i)^2\right]$

Codes convolutifs algorithmes SISO : SOVA

Application de l'algorithme de Viterbi

- A chaque instant i, calcul de la *métrique de branche* $L_i^{\mathbf{x}}$ associée à chaque chemin **x** dans le treillis, c-a-d de la distance euclidienne entre le symbole reçu r_i et le symbole codé c_i associé au chemin **x**.

$$L_i^{\mathbf{x}} = (x_i - X_i)^2 + (y_i - Y_i)^2$$

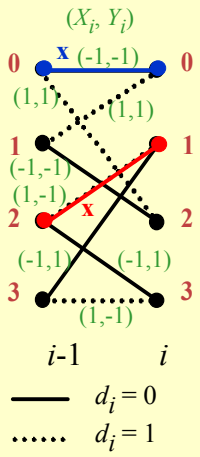
- Calcul (récursif) de la *métrique de chemin* $M_i^{\mathbf{x}}$ associée au chemin **x.**

$$M_i^{\mathbf{x}} = \sum_{j=1}^{i} L_j^{\mathbf{x}} = M_{i-1}^{\mathbf{x}} + L_i^{\mathbf{x}}$$

Codes convolutifs algorithmes SISO : SOVA

Application de l'algorithme de Viterbi

Exemple



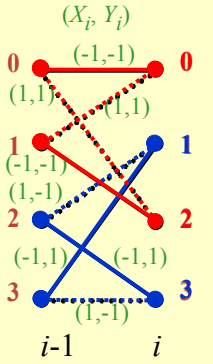
$$L_i^{\mathbf{x}} = L_i^{\mathbf{0,0}} = (x_i+1)^2 + (y_i+1)^2$$
$$M_i^{\mathbf{x}} = M_i^{\mathbf{0,0}} = (x_i+1)^2 + (y_i+1)^2 + M_{i-1}^{\mathbf{x}}$$

$$L_i^{\mathbf{x}} = L_i^{\mathbf{2,1}} = (x_i-1)^2 + (y_i+1)^2$$
$$M_i^{\mathbf{x}} = M_i^{\mathbf{2,1}} = (x_i-1)^2 + (y_i+1)^2 + M_{i-1}^{\mathbf{x}}$$

Codes convolutifs algorithmes SISO : SOVA

Application de l'algorithme de Viterbi

- Pour chaque nœud **s**, recherche du chemin à métrique minimale (le *survivant*) et mémorisation de la métrique associée $M_i^s(\mathbf{s})$ avec la valeur de d_i.

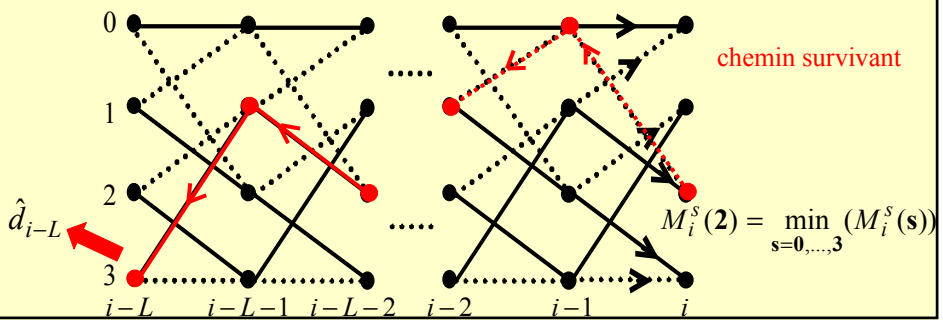


$$M_i^{\mathbf{0,0}} \Rightarrow M_i^s(\mathbf{0}) \quad M_i^{\mathbf{1,0}}$$
$$M_i^{\mathbf{3,1}} \quad M_i^{\mathbf{2,1}} \Rightarrow M_i^s(\mathbf{1})$$
$$M_i^{\mathbf{1,2}} \Rightarrow M_i^s(\mathbf{2}) \quad M_i^{\mathbf{0,2}}$$
$$M_i^{\mathbf{2,3}} \quad M_i^{\mathbf{3,3}} \Rightarrow M_i^s(\mathbf{3})$$

Codes convolutifs algorithmes SISO : SOVA

Application de l'algorithme de Viterbi

- A l'instant i, sélection du nœud à métrique minimale,
- Remontée le long du chemin survivant (pour profiter du futur).
- A l'instant i-L, la décision $\hat{d}_{i-L}$ est la valeur binaire stockée dans le nœud appartenant au chemin survivant.



Codes convolutifs algorithmes SISO : SOVA

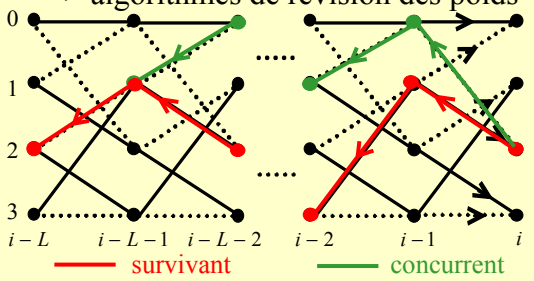
- L'algorithme de Viterbi ne peut pas être utilisé pour le décodage itératif !
  - Il produit des décisions binaires
  - Il doit être modifié pour produire des décisions pondérées => SOVA (Soft-Output Viterbi Algorithm)

Codes convolutifs algorithmes SISO : SOVA

- Une première estimation de la fiabilité ou du poids de la décision peut être donnée par la différence entre les métriques du chemin survivant et de l'autre chemin (le chemin concurrent):

$$w_i = \Delta M_i(\mathbf{s}) = M_i^c(\mathbf{s}) - M_i^s(\mathbf{s}) \geq 0$$

- Mais cette estimation doit être affinée!
  => algorithmes de révision des poids



Si $\Delta M_{i-1}(\mathbf{1}) >> 0$
et $\Delta M_i(\mathbf{2}) = 0$

=> le poids à l'instant *i*-1 doit être revu à la baisse

Codes convolutifs algorithmes SISO : MAP

L'algorithme MAP (*Maximum A Posteriori*)

- Encore appelé algorithme BCJR (Bahl, Cocke, Jelinek, Raviv), algorithme APP (*A Posteriori Probability*), ou algorithme Aller-Retour (*Backward-Forward*)
- But de l'algorithme : calcul du Logarithme de Rapport de Vraisemblance (LRV) relative à la donnée d_i

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

signe ⇒ décision binaire
valeur absolue ⇒ fiabilité

$\mathbf{r}_1^k$ désigne la séquence bruitée reçue

$\mathbf{r}_1^k = \{r_1, \cdots, r_k\}$ avec $r_i = (x_i, y_i)$

Codes convolutifs — algorithmes SISO : MAP

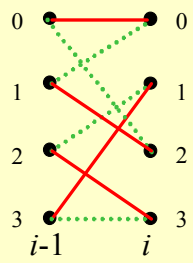
- Chaque APP (*A Posteriori* Probability) $\Pr\{d_i = j \,|\, \mathbf{r}_1^k\}$ est estimée par le biais des probabilités conjointes

$$\Pr\{\mathbf{S}_{i-1} = \mathbf{s}', \mathbf{S}_i = \mathbf{s}, \mathbf{r}_1^k\}$$

$\mathbf{S}_i$: état du codeur à l'instant i

$$\Pr\{d_i = j \,|\, \mathbf{r}_1^k\} = \frac{\sum_{(\mathbf{s}',\mathbf{s}) \in T_i^j} \Pr\{\mathbf{S}_{i-1} = \mathbf{s}', \mathbf{S}_i = \mathbf{s}, \mathbf{r}_1^k\}}{\Pr\{\mathbf{r}_1^k\}} \qquad j = 0,1$$

T_i^j : ensemble des transitions du treillis associées à $d_i = j$



$$\Rightarrow \quad \Lambda(d_i) = \ln \frac{\sum_{(\mathbf{s}',\mathbf{s}) \in T_i^1} \Pr\{\mathbf{S}_{i-1} = \mathbf{s}', \mathbf{S}_i = \mathbf{s}, \mathbf{r}_1^k\}}{\sum_{(\mathbf{s}',\mathbf{s}) \in T_i^0} \Pr\{\mathbf{S}_{i-1} = \mathbf{s}', \mathbf{S}_i = \mathbf{s}, \mathbf{r}_1^k\}}$$

Codes convolutifs — algorithmes SISO : MAP

- Principe de l'algorithme MAP : traiter séparément les données relatives aux instants compris entre 1 et i et celles relatives aux instants compris entre $i+1$ et k.

=> Introduction des probabilités « aller » (*forward*) $\alpha_i(\mathbf{s})$, $i = 1 \cdots k$ et « retour » (*backward*) $\beta_i(\mathbf{s})$, $i = 1 \cdots k$ relatives aux états du treillis

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

$$\beta_i(\mathbf{s}) = \Pr\{\mathbf{r}_{i+1}^k \,|\, \mathbf{S}_i = \mathbf{s}\}$$

=> et des probabilités relatives aux transitions du treillis

$$\gamma_i(\mathbf{s}', \mathbf{s}) = \Pr\{\mathbf{S}_i = \mathbf{s}, r_i \,|\, \mathbf{S}_{i-1} = \mathbf{s}'\}$$

Codes convolutifs algorithmes SISO : MAP

- On peut montrer que

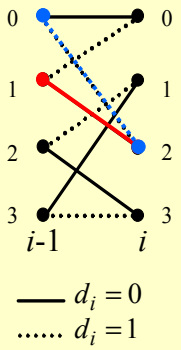
$$\Pr\{\mathbf{S}_{i-1}=\mathbf{s}',\mathbf{S}_i=\mathbf{s},\mathbf{r}_1^k\}=\alpha_{i-1}(\mathbf{s}')\gamma_i(\mathbf{s}',\mathbf{s})\beta_i(\mathbf{s})$$

$$\Rightarrow \quad \Lambda(d_i)=\ln\frac{\sum_{(\mathbf{s}',\mathbf{s})\in T_i^1}\alpha_{i-1}(\mathbf{s}')\gamma_i(\mathbf{s}',\mathbf{s})\beta_i(\mathbf{s})}{\sum_{(\mathbf{s}',\mathbf{s})\in T_i^0}\alpha_{i-1}(\mathbf{s}')\gamma_i(\mathbf{s}',\mathbf{s})\beta_i(\mathbf{s})}$$

Codes convolutifs algorithmes SISO : MAP

Exemple

* $\Pr\{\mathbf{S}_{i-1}=\mathbf{1},\mathbf{S}_i=\mathbf{2},\mathbf{r}_1^k\}=\alpha_{i-1}(\mathbf{1})\gamma_i(\mathbf{1},\mathbf{2})\beta_i(\mathbf{2})$
* $\Pr\{\mathbf{S}_{i-1}=\mathbf{0},\mathbf{S}_i=\mathbf{2},\mathbf{r}_1^k\}=\alpha_{i-1}(\mathbf{0})\gamma_i(\mathbf{0},\mathbf{2})\beta_i(\mathbf{2})$



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

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

Codes convolutifs — algorithmes SISO : MAP

- **Récursion aller** : calcul de $\alpha_i(\mathbf{s})$

$$\alpha_i(\mathbf{s}) = \sum_{\mathbf{s}'=0}^{2^\nu - 1} \alpha_{i-1}(\mathbf{s}')\gamma_i(\mathbf{s}',\mathbf{s}) \qquad (\nu = \text{mémoire du code})$$

Initialisation: si $\mathbf{s}_0$ est l'état initial du codeur

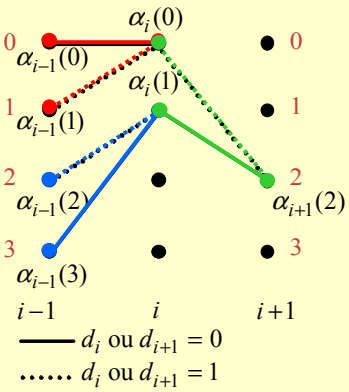
$$\alpha_0(\mathbf{s}_0) = 1$$
$$\alpha_0(\mathbf{s}) = 0 \quad \forall \mathbf{s} \neq \mathbf{s}_0$$

NB: pour éviter les problèmes de précision, $\alpha_i(\mathbf{s})$ peut être normalisé à chaque étape de calcul ($\Lambda(d_i)$est un rapport)

$$\alpha'_i(\mathbf{s}) = \frac{\alpha_i(\mathbf{s})}{\sum_{\mathbf{s}'=0}^{2^\nu - 1} \alpha_i(\mathbf{s}')}$$

Codes convolutifs — algorithmes SISO : MAP

Exemple:
(sans normalisation)



- $\alpha_i(\mathbf{0}) = \alpha_{i-1}(\mathbf{0})\gamma_i(\mathbf{0},\mathbf{0}) + \alpha_{i-1}(\mathbf{1})\gamma_i(\mathbf{1},\mathbf{0})$
- $\alpha_i(\mathbf{1}) = \alpha_{i-1}(\mathbf{3})\gamma_i(\mathbf{3},\mathbf{1}) + \alpha_{i-1}(\mathbf{2})\gamma_i(\mathbf{2},\mathbf{1})$
- $\alpha_{i+1}(\mathbf{2}) = \alpha_i(\mathbf{1})\gamma_{i+1}(\mathbf{1},\mathbf{2}) + \alpha_i(\mathbf{0})\gamma_{i+1}(\mathbf{0},\mathbf{2})$

Codes convolutifs algorithmes SISO : MAP

- **Récursion retour** : calcul de $\beta_i(\mathbf{s}')$

$$\beta_i(\mathbf{s}') = \sum_{\mathbf{s}=0}^{2^\nu-1} \beta_{i+1}(\mathbf{s})\gamma_{i+1}(\mathbf{s}',\mathbf{s})$$

Initialisation : si $\mathbf{s}_k$ est l'état final du codeur

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

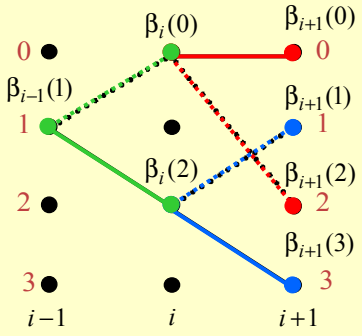
Si l'état final est inconnu :

$$\beta_k(\mathbf{s}) = \frac{1}{2^\nu} \quad \forall \mathbf{s}$$

Normalisation: $\beta'_i(\mathbf{s}) = \dfrac{\beta_i(\mathbf{s})}{\sum_{\mathbf{s}'=0}^{2^\nu-1} \beta_i(\mathbf{s}')}$

Codes convolutifs algorithmes SISO : MAP

Exemple:
(sans normalisation)



* $\beta_i(\mathbf{0}) = \beta_{i+1}(\mathbf{0})\gamma_{i+1}(\mathbf{0},\mathbf{0}) + \beta_{i+1}(\mathbf{2})\gamma_{i+1}(\mathbf{0},\mathbf{2})$

* $\beta_i(\mathbf{2}) = \beta_{i+1}(\mathbf{3})\gamma_{i+1}(\mathbf{2},\mathbf{3}) + \beta_{i+1}(\mathbf{1})\gamma_{i+1}(\mathbf{2},\mathbf{1})$

* $\beta_{i-1}(\mathbf{1}) = \beta_i(\mathbf{2})\gamma_i(\mathbf{1},\mathbf{2}) + \beta_i(\mathbf{0})\gamma_i(\mathbf{1},\mathbf{0})$

Codes convolutifs algorithmes SISO : MAP

- **Calcul des probabilités de branches** $\gamma_i(\mathbf{s}', \mathbf{s})$

S'il n'y a pas de transition entre **s'** et **s** dans le treillis.

$$\gamma_i(\mathbf{s}', \mathbf{s}) = 0$$

sinon

$$\gamma_i(\mathbf{s}', \mathbf{s}) = \Pr\{d_i = \mathrm{j}\} \Pr\{r_i \mid c_i\}$$

si $d_i = j$ est le bit d'information et c_i le symbole codé et modulé correspondant à la transition $\mathbf{s}' \rightarrow \mathbf{s}$ du treillis et r_i est le symbole reçu, à l'instant i .

Pour une transmission sur canal gaussien :

$$\Pr\{r_i \mid c_i\} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|r_i - c_i\|^2}{2\sigma^2} \right)$$

σ^2 : variance du bruit sur le canal

Codes convolutifs algorithmes SISO : MAP

- **Calcul des probabilités de branches** $\gamma_i(\mathbf{s}', \mathbf{s})$

Si $c_i = (X_i, Y_i)$, $r_i = (x_i, y_i)$ et $\Pr\{d_i = 0\} = \Pr\{d_i = 1\} = \frac{1}{2}$

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

$$\gamma_i(\mathbf{s}', \mathbf{s}) = K \exp\left(\frac{x_i}{\sigma^2}\right) \exp\left(\frac{y_i}{\sigma^2}\right) \quad (X_i = -1, Y_i = -1)$$

$$\gamma_i(\mathbf{s}', \mathbf{s}) = K \exp\left(\frac{x_i}{\sigma^2}\right) \exp\left(-\frac{y_i}{\sigma^2}\right) \quad (X_i = -1, Y_i = +1)$$

$$\gamma_i(\mathbf{s}', \mathbf{s}) = K \exp\left(-\frac{x_i}{\sigma^2}\right) \exp\left(\frac{y_i}{\sigma^2}\right) \quad (X_i = +1, Y_i = -1)$$

$$\gamma_i(\mathbf{s}', \mathbf{s}) = K \exp\left(-\frac{x_i}{\sigma^2}\right) \exp\left(-\frac{y_i}{\sigma^2}\right) \quad (X_i = +1, Y_i = +1)$$

N. B. $\Lambda(d_i)$ étant un rapport, K peut être omis dans les expressions de $\alpha_i(\mathbf{s})$ et $\beta_i(\mathbf{s})$.

Codes convolutifs — algorithmes SISO : Log-MAP et Sub-MAP

L'algorithme MAP dans le domaine logarithmique (Log-MAP)

- Multiplications => additions
- Les exponentielles dans les probas de branches s'éliminent
- Additions => ?

Solution 1: Log-MAP

$$\ln(e^a + e^b) = \max(a,b) + \ln(1 + e^{-|a-b|})$$

Le terme $\ln(1+e^{-|a-b|})$ est pré-calculé et stocké dans une table

☺ Performance ⇔ MAP

☹ Connaissance de σ requise

Codes convolutifs — algorithmes SISO : Sub-MAP

L'algorithme MAP dans le domaine logarithmique (Sub-MAP)

Solution 2: Sub-MAP (Max-Log-MAP)

$$\ln(e^a + e^b) \approx \max(a,b)$$

⇒ Algorithme « Dual Viterbi »

☹ Performance < MAP (qqes dixièmes de dB)

☺ Estimation de σ inutile

Codes convolutifs — algorithmes SISO : Sub-MAP

- Reformulation de $\Lambda(d_i)$

$$\Lambda(d_i) = \ln \sum_{(\mathbf{s}',\mathbf{s})\in T_i^1} \alpha_{i-1}(\mathbf{s}')\,\gamma_i(\mathbf{s}',\mathbf{s})\beta_i(\mathbf{s}) - \ln \sum_{(\mathbf{s}',\mathbf{s})\in T_i^0} \alpha_{i-1}(\mathbf{s}')\,\gamma_i(\mathbf{s}',\mathbf{s})\,\beta_i(\mathbf{s})$$

$$\Lambda(d_i) \approx \max_{(\mathbf{s}',\mathbf{s})\in T_i^1} \{\ln(\alpha_{i-1}(\mathbf{s}')\,\gamma_i(\mathbf{s}',\mathbf{s})\,\beta_i(\mathbf{s}))\} - \max_{(\mathbf{s}',\mathbf{s})\in T_i^0} \{\ln(\alpha_{i-1}(\mathbf{s}')\,\gamma_i(\mathbf{s}',\mathbf{s})\,\beta_i(\mathbf{s}))\}$$

Métriques de nœuds et de branches :

$M_i^F(\mathbf{s}) \overset{\Delta}{=} -\sigma^2 \ln \alpha_i(\mathbf{s})$ Métrique de nœud aller *(forward)*

$M_i^B(\mathbf{s}) \overset{\Delta}{=} -\sigma^2 \ln \beta_i(\mathbf{s})$ Métrique de nœud retour *(backward)*

$m_i(\mathbf{s}',\mathbf{s}) \overset{\Delta}{=} -\sigma^2 \ln \gamma_i(\mathbf{s}',\mathbf{s})$ Métrique de branche

Codes convolutifs — algorithmes SISO : Sub-MAP

L'algorithme Sub-MAP calcule $\Lambda'(d_i) \overset{\Delta}{=} \frac{\sigma^2}{2}\Lambda(d_i)$:

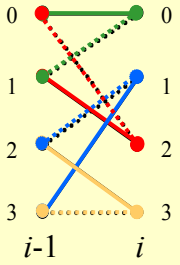
$$\Lambda'(d_i) \approx \frac{1}{2}\left[\min_{(\mathbf{s}',\mathbf{s})\in T_i^0} \left\{M_{i-1}^F(\mathbf{s}') + m_i(\mathbf{s}',\mathbf{s}) + M_i^B(\mathbf{s})\right\} - \min_{(\mathbf{s}',\mathbf{s})\in T_i^1} \left(M_{i-1}^F(\mathbf{s}') + m_i(\mathbf{s}',\mathbf{s}) + M_i^B(\mathbf{s})\right)\right]$$

$$\Lambda'(d_i) \approx \frac{1}{2}\left[\min_{(\mathbf{s}',\mathbf{s})\in T_i^0} \{M_{i-1}^F(\mathbf{s}') + x_i \pm y_i + M_i^B(\mathbf{s})\} - \min_{(\mathbf{s}',\mathbf{s})\in T_i^1} \{M_{i-1}^F(\mathbf{s}') - x_i \pm y_i + M_i^B(\mathbf{s})\}\right]$$

Codes convolutifs — algorithmes SISO : Sub-MAP

Exemple:

$$\Lambda'(d_i) = \frac{1}{2}\Big[\min\big(M_{i-1}^F(\mathbf{0}) + m_i(\mathbf{0},\mathbf{0}) + M_i^B(\mathbf{0}),\ M_{i-1}^F(\mathbf{3}) + m_i(\mathbf{3},\mathbf{1}) + M_i^B(\mathbf{1}),$$
$$M_{i-1}^F(\mathbf{1}) + m_i(\mathbf{1},\mathbf{2}) + M_i^B(\mathbf{2}),\ M_{i-1}^F(\mathbf{2}) + m_i(\mathbf{2},\mathbf{3}) + M_i^B(\mathbf{3})\big)$$
$$- \min\big(M_{i-1}^F(\mathbf{1}) + m_i(\mathbf{1},\mathbf{0}) + M_i^B(\mathbf{0}),\ M_{i-1}^F(\mathbf{2}) + m_i(\mathbf{2},\mathbf{1}) + M_i^B(\mathbf{1}),$$
$$M_{i-1}^F(\mathbf{0}) + m_i(\mathbf{0},\mathbf{2}) + M_i^B(\mathbf{2}),\ M_{i-1}^F(\mathbf{3}) + m_i(\mathbf{3},\mathbf{3}) + M_i^B(\mathbf{3})\big)\Big]$$



Codes convolutifs — algorithmes SISO : Sub-MAP

$M_i^F(\mathbf{s})$ et $M_i^B(\mathbf{s})$ sont calculés récursivement :

$$M_i^F(\mathbf{s}) \approx \min_{\mathbf{s}'=0\cdots 2^\nu-1}\{M_{i-1}^F(\mathbf{s}') + m_i(\mathbf{s}',\mathbf{s})\}$$

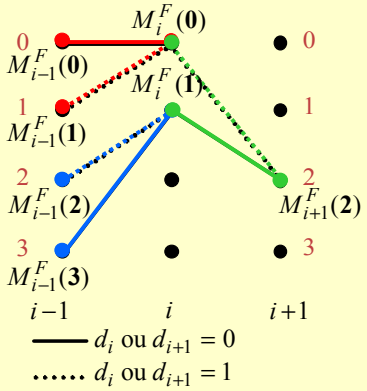
$$\boxed{M_i^F(\mathbf{s}) \approx \min_{\mathbf{s}'=0\cdots 2^\nu-1}\{M_{i-1}^F(\mathbf{s}') \pm x_i \pm y_i\}}$$

$$M_i^B(\mathbf{s}') \approx \min_{\mathbf{s}=0\cdots 2^\nu-1}\{M_{i+1}^B(\mathbf{s}) + m_{i+1}(\mathbf{s}',\mathbf{s})\}$$

$$\boxed{M_i^B(\mathbf{s}') \approx \min_{\mathbf{s}=0\cdots 2^\nu-1}\{M_{i+1}^B(\mathbf{s}) \pm x_{i+1} \pm y_{i+1}\}}$$

Codes convolutifs — algorithmes SISO : Sub-MAP

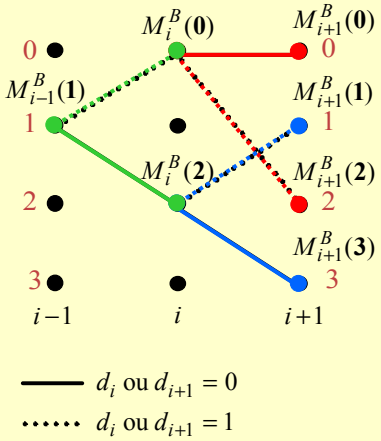
Exemple



- $M_i^F(\mathbf{0}) = \min\left(M_{i-1}^F(\mathbf{0}) + m_i(\mathbf{0},\mathbf{0}), M_{i-1}^F(\mathbf{1}) + m_i(\mathbf{1},\mathbf{0})\right)$
- $M_i^F(\mathbf{1}) = \min\left(M_{i-1}^F(\mathbf{3}) + m_i(\mathbf{3},\mathbf{1}), M_{i-1}^F(\mathbf{2}) + m_i(\mathbf{2},\mathbf{1})\right)$
- $M_{i+1}^F(\mathbf{2}) = \min\left(M_i^F(\mathbf{1}) + m_{i+1}(\mathbf{1},\mathbf{2}), M_i^F(\mathbf{0}) + m_{i+1}(\mathbf{0},\mathbf{2})\right)$

Codes convolutifs — algorithmes SISO : Sub-MAP

Exemple



- $M_i^B(\mathbf{0}) = \min\left(M_{i+1}^B(\mathbf{0}) + m_{i+1}(\mathbf{0},\mathbf{0}), M_{i+1}^B(\mathbf{2}) + m_{i+1}(\mathbf{0},\mathbf{2})\right)$
- $M_i^B(\mathbf{2}) = \min\left(M_{i+1}^B(\mathbf{1}) + m_{i+1}(\mathbf{2},\mathbf{1}), M_{i+1}^B(\mathbf{3}) + m_{i+1}(\mathbf{2},\mathbf{3})\right)$
- $M_{i-1}^B(\mathbf{1}) = \min\left(M_i^B(\mathbf{0}) + m_i(\mathbf{1},\mathbf{0}), M_i^B(\mathbf{2}) + m_i(\mathbf{1},\mathbf{2})\right)$

Codes convolutifs — algorithmes SISO : Sub-MAP

- Obtention de l'*information extrinsèque* pour le *décodage itératif* :

$$\Lambda'(d_i) \approx \frac{1}{2}\left[\min_{(\mathbf{s}',\mathbf{s})\in T_i^0}\{M_{i-1}^F(\mathbf{s}') + x_i \pm y_i + M_i^B(\mathbf{s})\} - \min_{(\mathbf{s}',\mathbf{s})\in T_i^1}\{M_{i-1}^F(\mathbf{s}') - x_i \pm y_i + M_i^B(\mathbf{s})\}\right]$$

$\Lambda'(d_i)$ peut s'écrire :

$$\Lambda'(d_i) \approx x_i + Z_i$$

Z_i est l'*information extrinsèque* produite par le décodeur

Codes convolutifs — algorithmes SISO : Biblio

[1] G. Battail, "Pondération des symboles décodés par l'algorithme de Viterbi", *Ann. Télécomm.,* vol. 42, N°1-2, pp. 31-38, Jan. 1987.

[2] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications", *in Proc. IEEE Globecom'89*, Dallas, Texas, Nov. 1989, pp. 4711-4717.

[3] C. Berrou, P. Adde, E. angui, and S. Faudeil, "A low complexity soft-output Viterbi decoder architecture", *in Proc. IEEE Int'l Conf. on Comm*., Geneva, Switzerland, 1993, pp.737-740.

[4] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284-287, 1974.

[5] P. Robertson, E. Villebrun, and P. Hoeher, " A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain", *in Proc. IEEE Int'l Conf. on Comm*., Seattle, WA, 1995, pp.1009-1013.

[6] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes", *IEEE Journal on Selected Areas in Comm.,* Vol. 16, N°2, Feb. 1998.

[7] B. Vucetic, J. Yuan, *Turbo Codes, Principles and Applications*, Kluwer Academic Publishers, 2000.

Chapitre 4

Turbo codes

Plan

- Mots croisés
- “turbo codes historique”
- Pourquoi de si bons résultats ?
- Différents schémas
  - Turbo Codes en bloc
  - Turbo Codes convolutifs

Mots croisés

Horizontal

I. Dommage
II. Lettre
III. Repas
IV. Minent
V. Enchâssement

Vertical

1. Gras
2. Envoyée
3. Indisposer
4. Intermédiaire
5. Gâteau

	1	2	3	4	5	
I	D	E	G	A	T	☺
II	G	A	M	M	A	☺
III	D	I	N	E	R	☺
IV	U	S	R	N	T	☹
V	D	A	R	I	G	☹
	☹	☹	☹	☹	☹	

	1	2	3	4	5	
I	D	E	G	A	T	☺
II	O	M	E	G	A	☺
III	D	I	N	E	R	☺
IV	U	S	E	N	T	☺
V	S	E	R	T	E	☺
	☺	☺	☺	☺	☺	

Turbo Codes

Inventés* en 1990, les Turbo Codes est une mise en oeuvre de la déclaration de Claude Shannon (1953) :

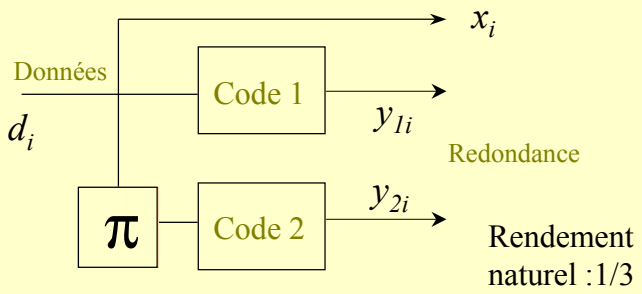
A scheme of coding and decoding can be found allowing correction of all transmission errors, if the information rate is inferior or equal to the channel capacity.

* première publication à ICC'93 – Genève - Suisses

Travaux précédents : Tanner, Gallager, Battail, Hagenauer & Hoegher, …

Codeur Turbo Codes

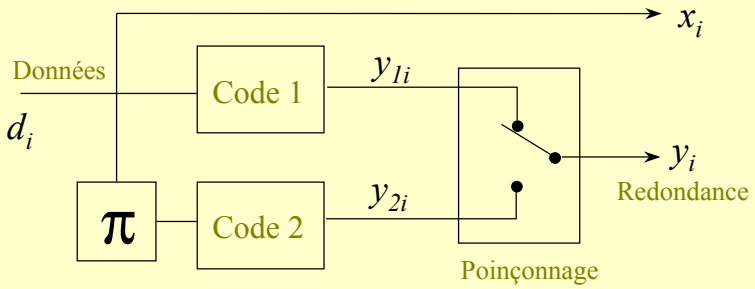
Un Turbo codeur est constitué d'au moins deux codeurs élémentaires de codes convolutifs systématiques (RSC) codes séparés par un entrelaceur



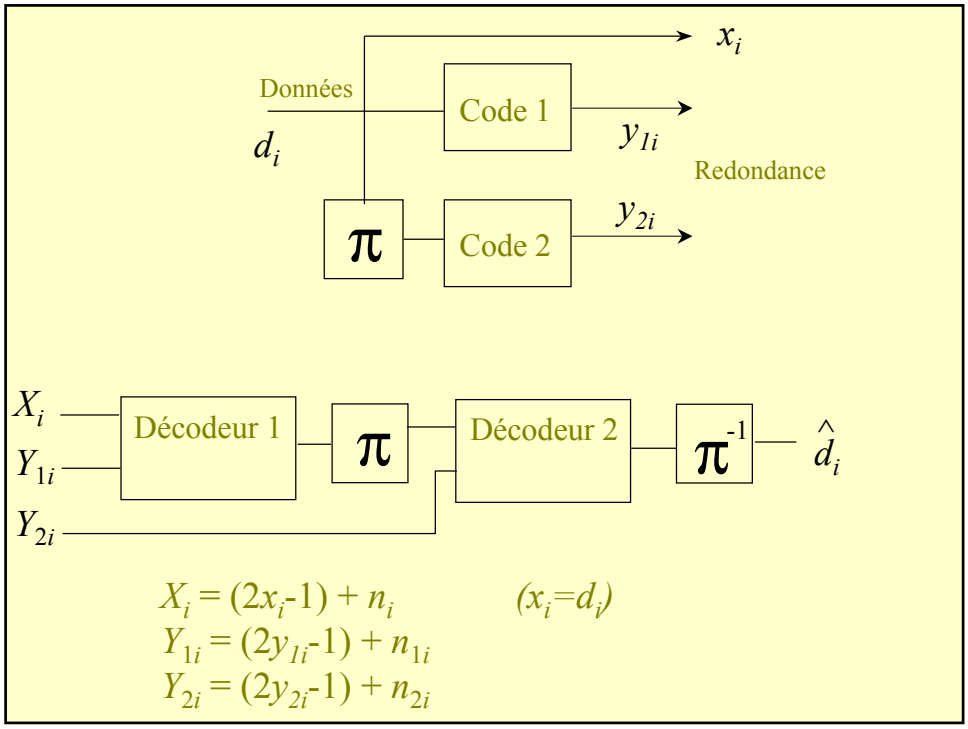
Concaténation parallèle de 2 codes RSC

Codeur Turbo Codes

Un Turbo codeur est constitué d'au moins deux codeurs élémentaires de codes convolutifs systématiques (RSC) codes séparés par un entrelaceur



Concaténation parallèle de 2 codes RSC (R=1/2)

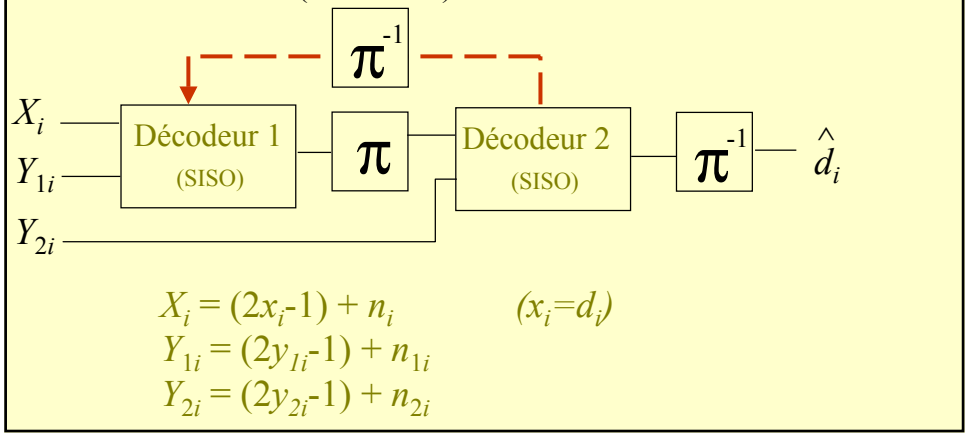


Turbo Codes

Décodeur

Construit sur la base de décodeurs élémentaires à entrées et sorties pondérées (*Soft Input Soft Output* SISO)

Processus itératif (effet turbo)



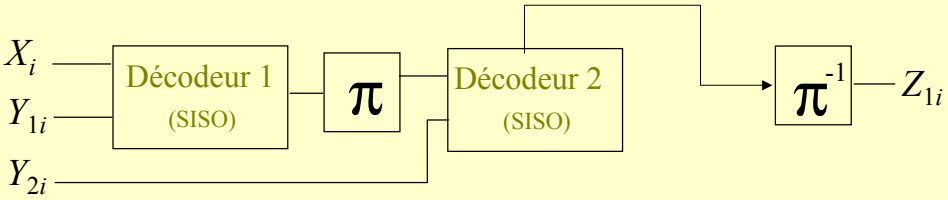
Turbo Codes

Décodeur

Construit sur la base de décodeurs élémentaires à entrées et sorties pondérées (*Soft Input Soft Output* SISO)

Processus itératif (effet turbo)

première itération



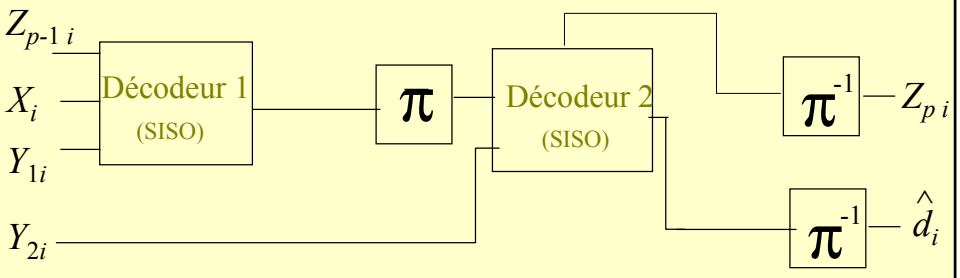
$X_i = (2d_i\text{-}1) + n_i$
$Y_{1i} = (2x_i\text{-}1) + n_{1i}$
$Y_{2i} = (2y_i\text{-}1) + n_{2i}$

Z_{1i} : *information extrinsèque*

Turbo Codes

Décodeur

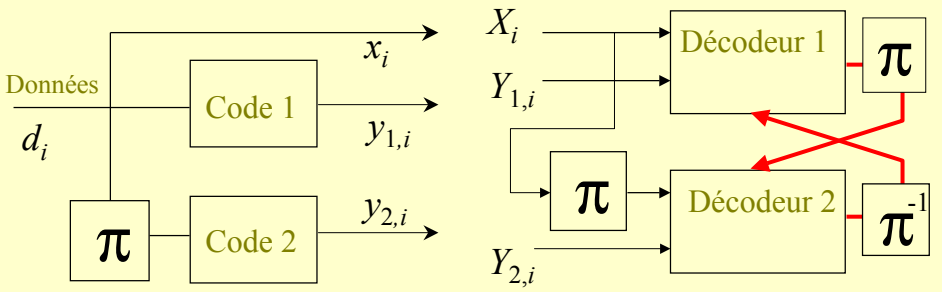
Itération p



$Z_{p\ i}$: information extrinsèque associée à la donnée d_i

$Z_{p\ i}$ et X_i sont altérées par des bruits non corrélés (effet de la diversité)

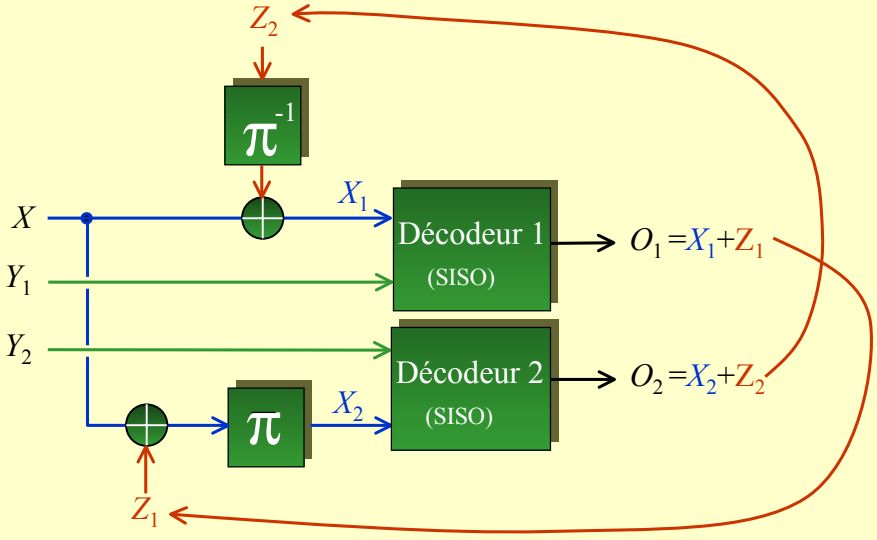
Comment faire pour que les deux décodeurs travaillent conjointement, c'est à dire que le décodeur 1 profite de Y_2 et que le décodeur 2 profite de Y_1 ?



« Message passing »
« turbo »

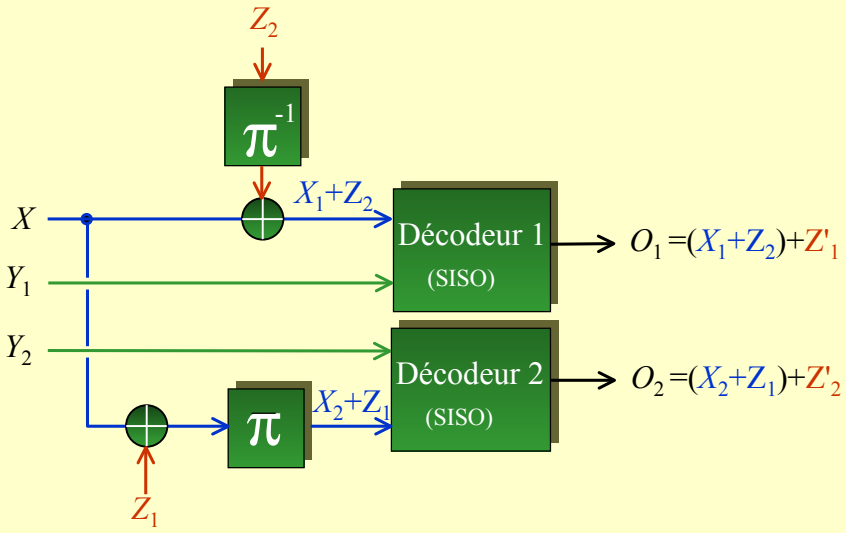
Turbo décodage — Information extrinsèque

- Critère de décodage : les deux décodeurs doivent converger vers la même solution (décision et probabilité associée)



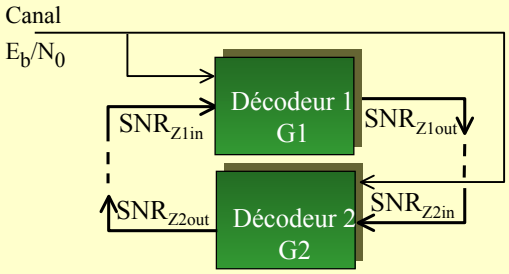
Turbo décodage — Information extrinsèque

- Critère de décodage : les deux décodeurs doivent converger vers la même solution (décision et probabilité associée)



Turbo décodage — Le principe turbo

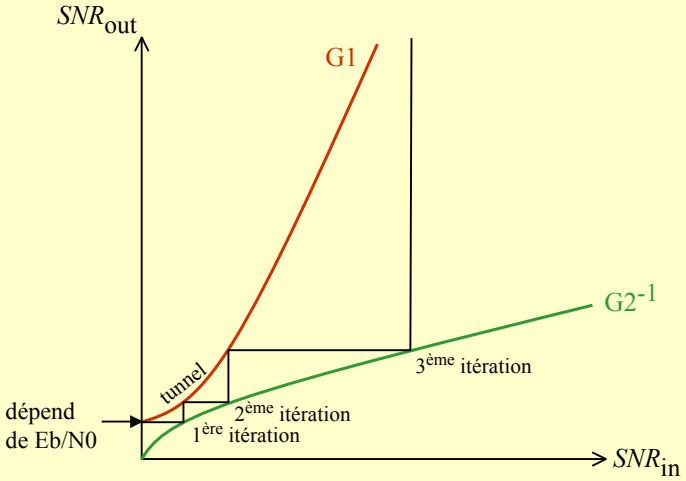
Itérations et convergence d'un turbo décodeur



- L'information Extrinsèque est approximée à une variable Gaussienne de moyenne $\overline{\lambda}$ et de variance σ_λ^2
- $SNR \overset{\Delta}{=} \overline{\lambda}^2 / \sigma_\lambda^2$
- Chaque décodeur est considéré au travers d'une fonction de transfert non-linéaire de rapport signal à bruit (SNR)
  SNR_{Z1out}=G1 (SNR_{Z1in}, E_b/N_0)
  SNR_{Z2out}=G2 (SNR_{Z2in}, E_b/N_0)

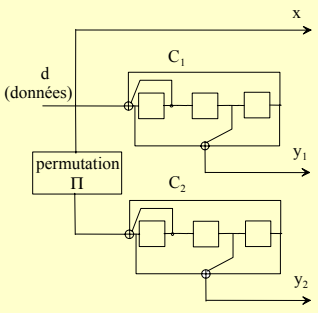
Turbo décodage — Le principe turbo

Itérations et convergence d'un turbo décodeur

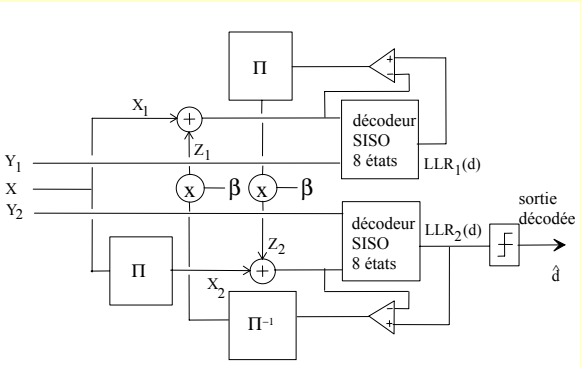


D. Divsalar, S. Dolinar, F. Pollara, "Low Complexity Turbo-like Codes", *Proc. 2nd International Symposium on Turbo Codes and Related Topics,* Brest, France sept. 2000, pp. 73-80

Turbo codeur



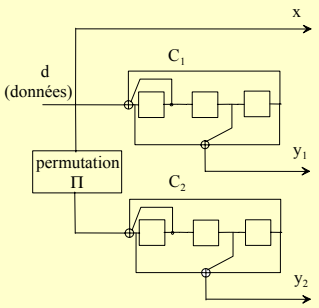
Turbo décodeur : une structure à contre-réaction



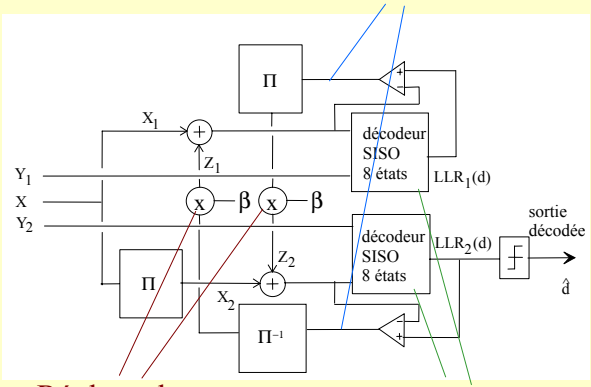
Adaptée pour un circuit analogique, mais pas pour un circuit numérique !!

⇒ Processus itératif

Turbo codeur



Turbo décodeur



Réglage du gain de boucle

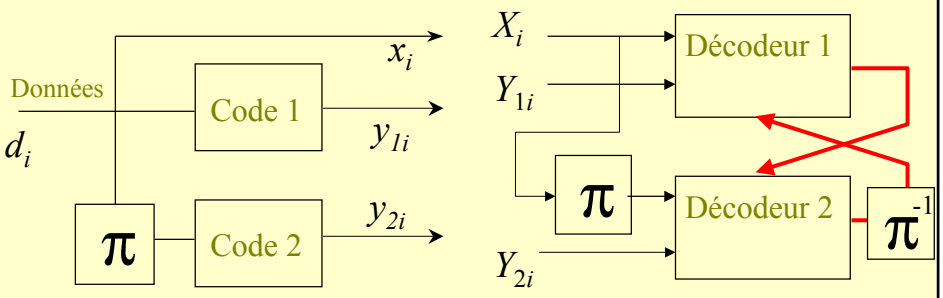
Décodeurs à entrées et sorties pondérées

Pourquoi de si bons résultats ?

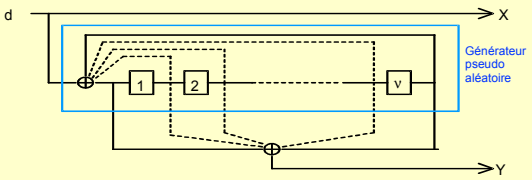
Turbo codes : similaires à des RSC de grande longueur

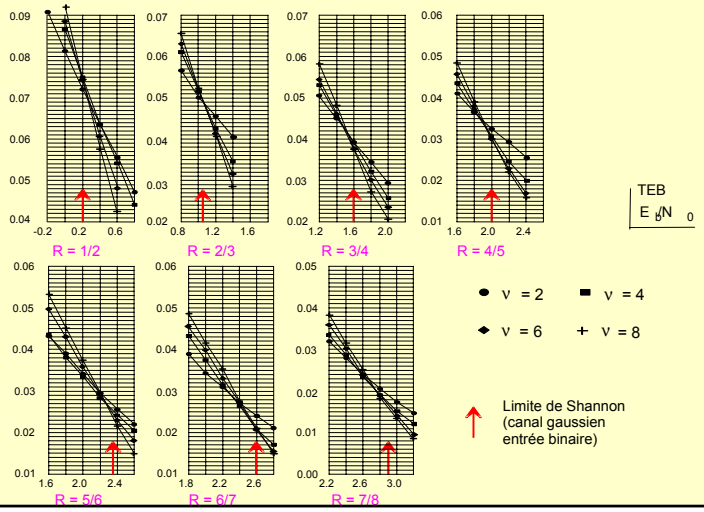
Il peut être comparé à un code convolutif ayant une grande longueur de contrainte,

qu'il est possible de décoder.

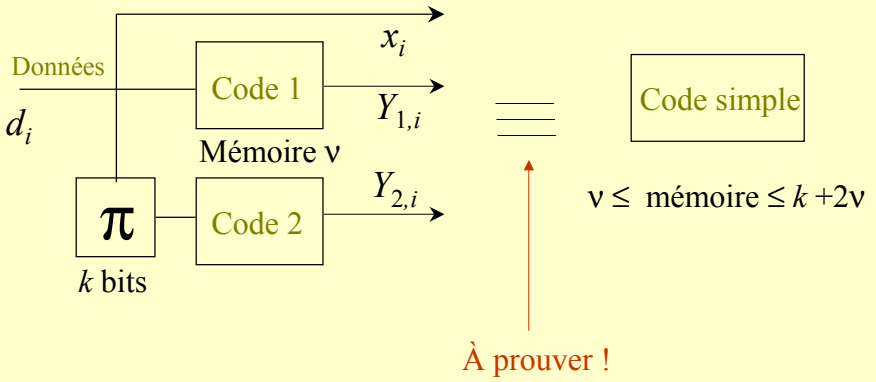


Plus le registre est grand, meilleur est le code convolutif



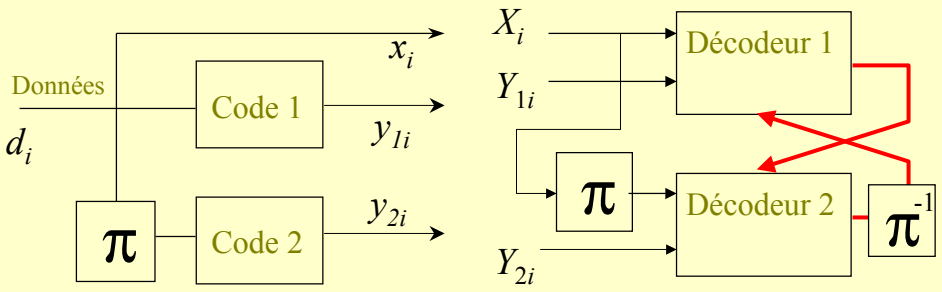


Turbo codes : similaires à des RSC de grande longueur

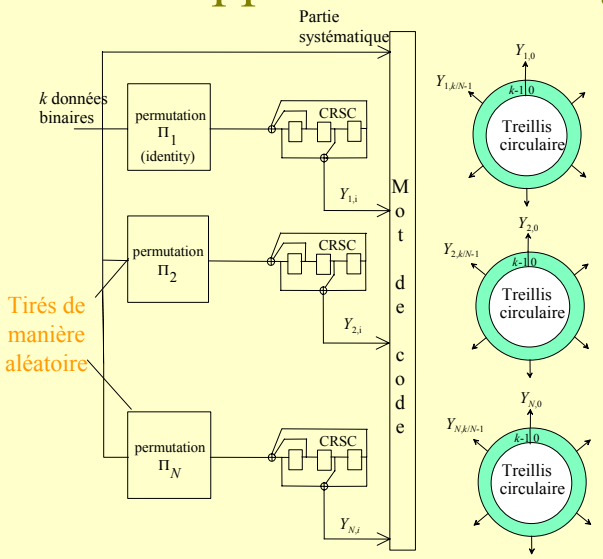


Pourquoi de si bons résultats ?

Lorsqu'un paquet d'erreurs affecte "code 1", les erreurs sont disséminées pour le "code 2" et vice versa.



Turbo codes : une approche du codage aléatoire



Concaténation parallèle de N codes CRSC:

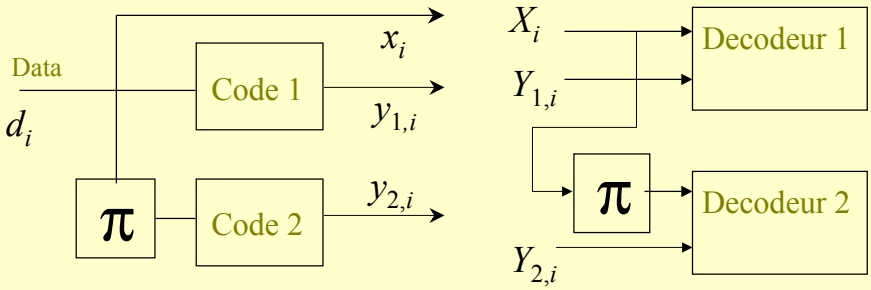
Codage quasi-aléatoire (pour $N \geq 4$)

Et c'est décodable !!

Heureusement, $N = 2$ est suffisant:

turbo codes

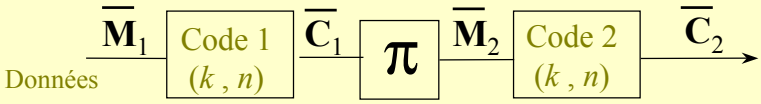
Le rôle fondamental de π :
minimiser la probabilité que les 2 décodeurs défaillent
(un problème très complexe combinant algèbre, géométrie, ...)

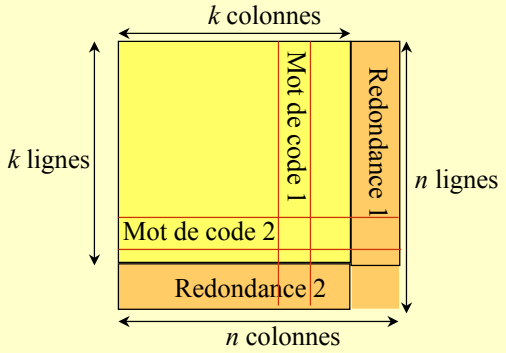


$X_i = (2x_i\text{-}1) + n_i \qquad (x_i = d_i)$
$Y_{1,i} = (2y_{1i}\text{-}1) + n_{1i}$
$Y_{2,i} = (2y_{2i}\text{-}1) + n_{2i}$

Différents schémas de turbo codes

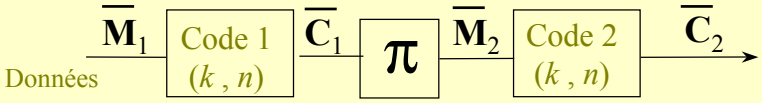
Turbo code en bloc





concaténation série de deux codes en bloc

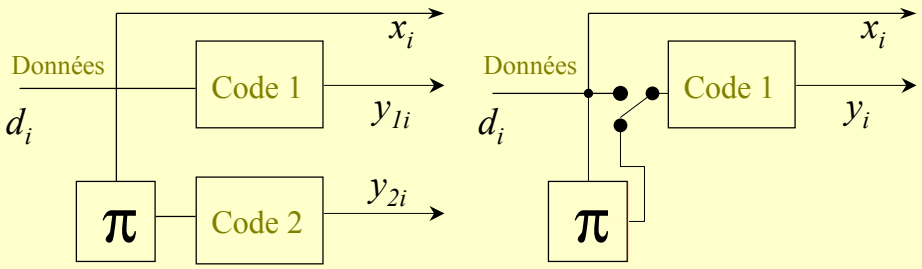
Turbo code en bloc



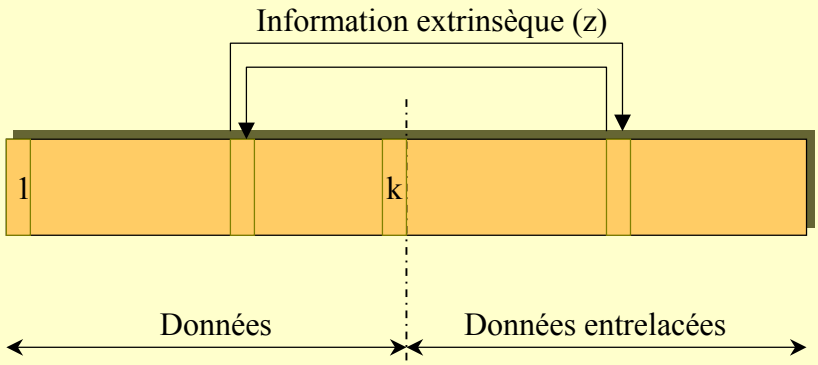
Propriétés (sans poinçonnage) :

- $n = n_1.n_2$
- $k = k_1.k_2$
- $R = R_1.R_2$
- $d_{min} = d_{min1}.d_{min2}$

Turbo codes convolutifs auto-concaténés

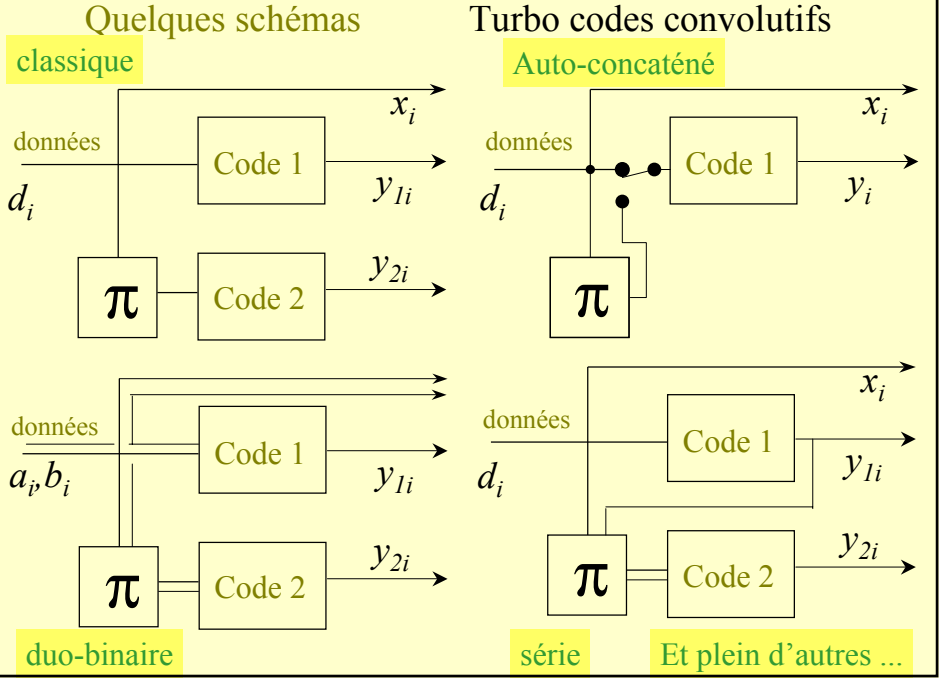


Principe de décodage



Quelques schémas

Turbo codes convolutifs

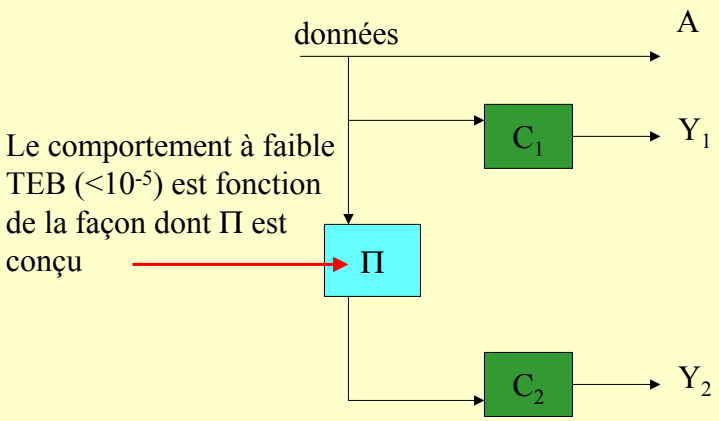


Chapitre 5

La permutation

Permutation

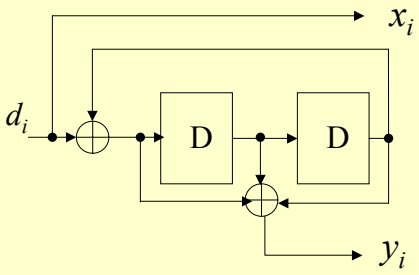
(Permutation ≡ entrelacement)

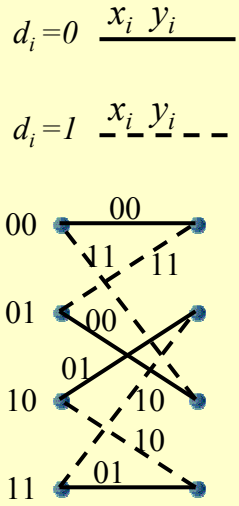


Le rôle fondamental de la permutation : si une séquence directe est RTZ, il faut minimiser la probabilité que la séquence permutée soit aussi RTZ (et vice-versa).

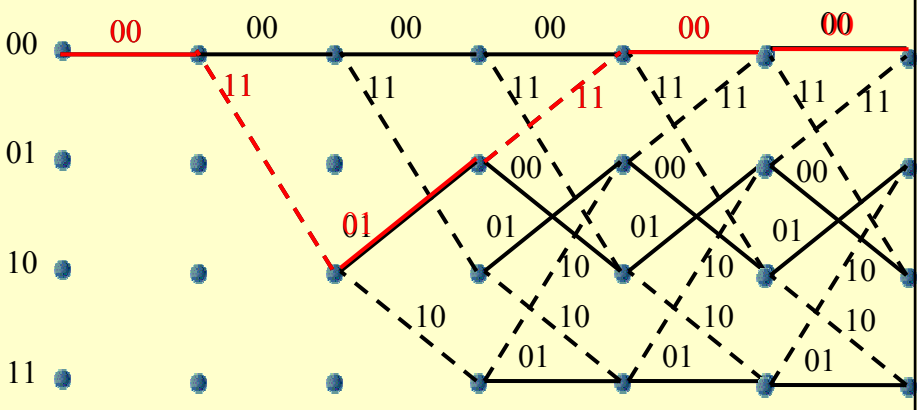
Séquences RTZ

Illustration avec le code suivant :

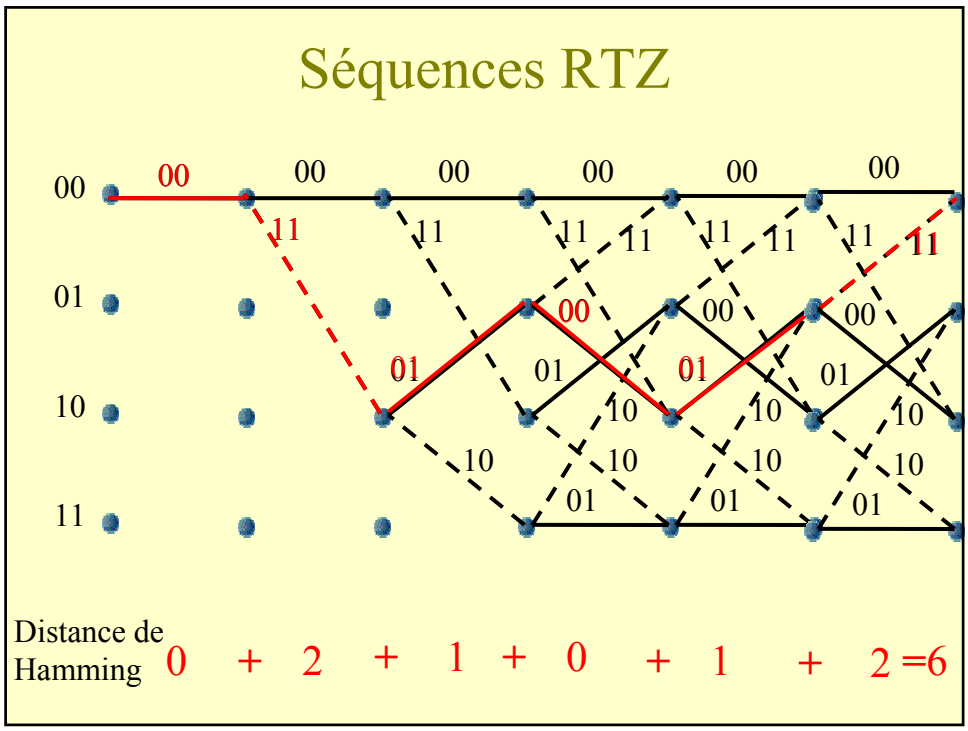


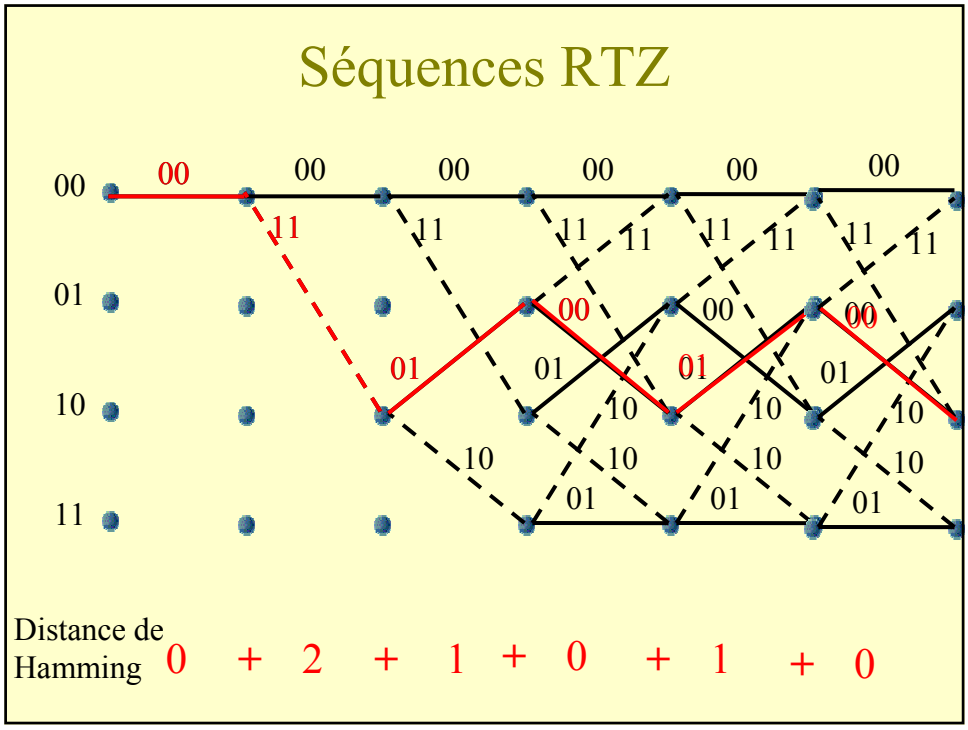


Séquences RTZ

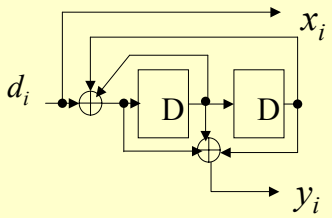


Distance de Hamming 0 + 2 + 1 + 2 + 0 + 0 =5


Séquences RTZ
00
01
10
11
Distance de Hamming
0 + 2 + 1 + 0 + 1 + 2 =6

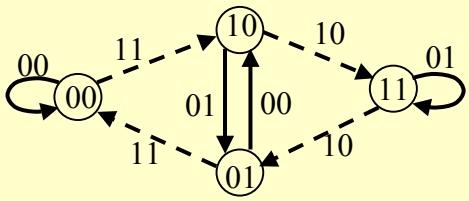

Séquences RTZ
00
01
10
11
Distance de Hamming
0 + 2 + 1 + 0 + 1 + 0

Séquences RTZ



$d_i = 0$ x_i y_i

$d_i = 1$ x_i y_i



Permutation Régulière

Ordre naturel : adresse 0

M colonnes

0010110010...0110001
.....
0100100100...1001010
0011110100...1011011
0101000110...0101001

N lignes

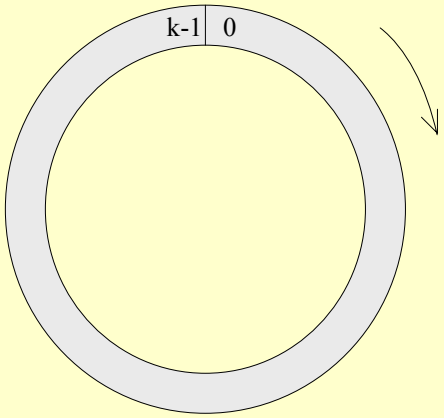
Écriture en ligne

Lecture en colonne

Ordre naturel : adresse k-1

Condition : k = M.N

Permutation Régulière



Adresse *naturelle* : i
$0 \le i \le k-1$

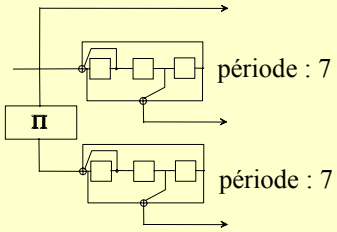
Adresse *permutée* : j
$0 \le j \le k-1$

$j = P.i \quad \text{mod. } k$

(P et k premiers entre eux)

Permutation Régulière

La permutation régulière est adaptée aux poids 2

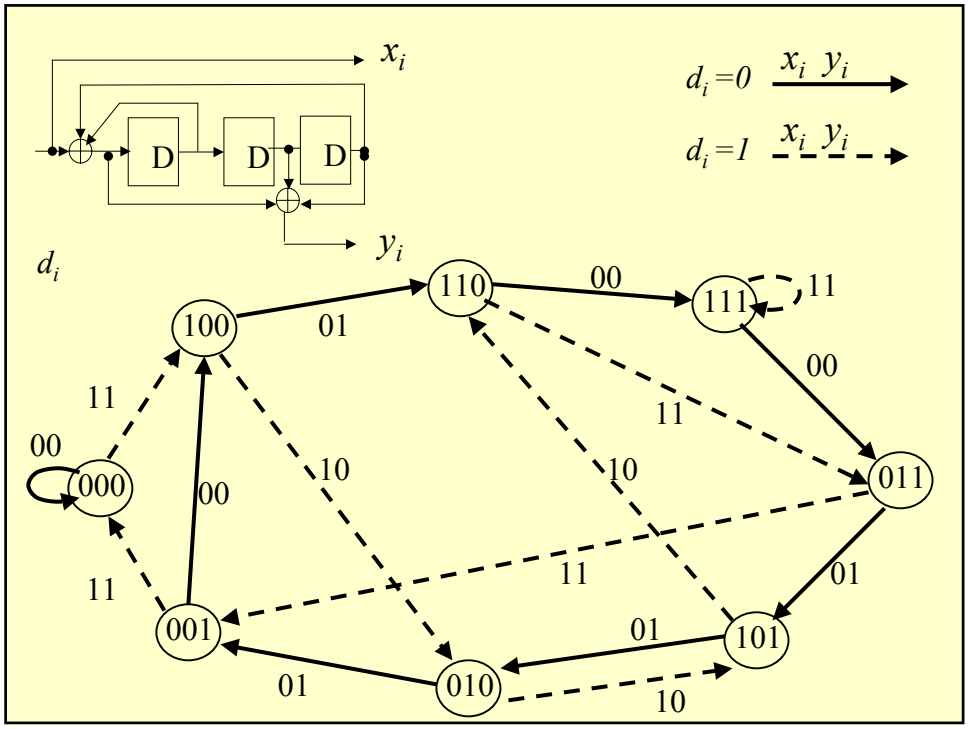


M colonnes

0100000010...0000000
.....
0000000000...0000000
0000000000...0000000
0000000000...0000000

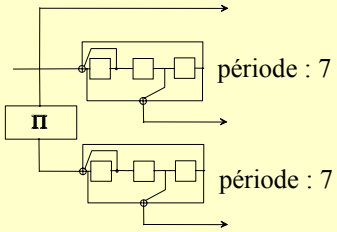
N lignes

$k \to \infty \Rightarrow d(w=2) \to \infty$



Permutation Régulière

La permutation régulière est adaptée aux poids 3



M colonnes

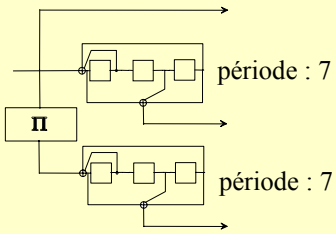
0110100000...0000000
.....
0000000000...0000000
0000000000...0000000
0000000000...0000000

N lignes

$k \rightarrow \infty \Rightarrow d(w=3) \rightarrow \infty$

Permutation Régulière

La permutation régulière n'est pas adaptée aux poids 4



M colonnes

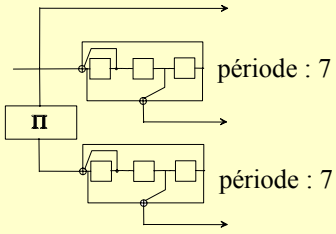
01000000010...0000000
0000000000...0000000
0000000000...0000000
0000000000...0000000
0000000000...0000000
0000000000...0000000
0000000000...0000000
01000000010...0000000
.....

N lignes

$k \rightarrow \infty \Rightarrow d(w=4)$ est limité

Permutation Régulière

La permutation régulière n'est pas adaptée pour différents poids



M colonnes

01101000000...0000000
01101000000...0000000
0000000000...0000000
01101000000...0000000
0000000000...0000000
0000000000...0000000
0000000000...0000000
0000000000...0000000
.....

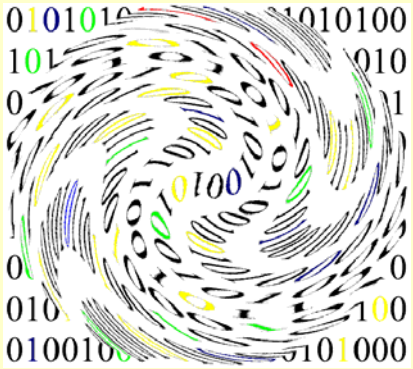
N lignes

Il faut donc introduire du désordre,
mais pas de n'importe quelle manière !!

Une bonne permutation doit assurer :
(1) Un maximum de dispersion des données
(2) Un maximum de désordre dans la séquence permutée

(ces deux conditions sont contradictoires)

Permutation Aléatoire



Permutation Aléatoire

Permutation aléatoire selon S. Benedetto et G. Montorsi

Tous les entrelaceurs sont considérés, de manière statistique, en y incluant les pires

Quels sont les pires ?

01000000010...0000000
.....
0000000000...0000000
0000000000...0000000
0000000000...0000000

ce couple a une probabilité de 1/k de rester inchangé après permutation

Permutation Aléatoire

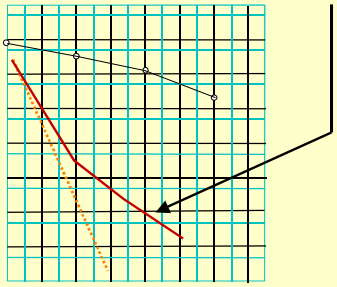
Permutation aléatoire selon S. Benedetto et G. Montorsi

Cette approche conduit naturellement à une borne qui a l'allure suivante :

$$P_e \approx \frac{1}{k}.erfc\left(\sqrt{\frac{R.E_b.d_{\min}}{N_0}}\right)$$

01000000010...0000000
.....
0000000000...0000000
0000000000...0000000
0000000000...0000000

Donnant une évaluation très pessimiste !



Permutation pseudo-aléatoire (= désordre contrôlé)

Chacun a ses recettes

Par exemple, l'entrelacement
du Consultative Committee for
Space Data Systems (CCSDS)

CCSDS 101.0-B-4: Telemetry Channel Coding.
Blue Book. Issue 4. May 1999.

voir http://ftp.ccsds.org/all_books.html#telemetry

Permutation CCSDS

non-uniformité de degré 8

$m = s - 1 \mod 2$

$i = 4\frac{s\text{-}1}{k}$

$j = \frac{s\text{-}1}{2} - \frac{ik}{8}$

$t = 19i + 1 \mod 4$

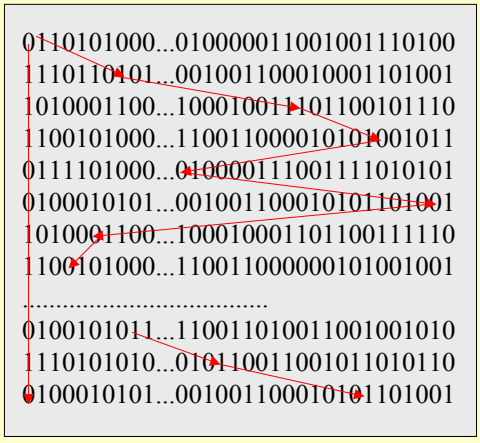
$q = t \mod 8 + 1$

$c = p_q j + 21m \mod \frac{k}{8}$

$\Pi(s) = 2(t + 4c + 1) - m$

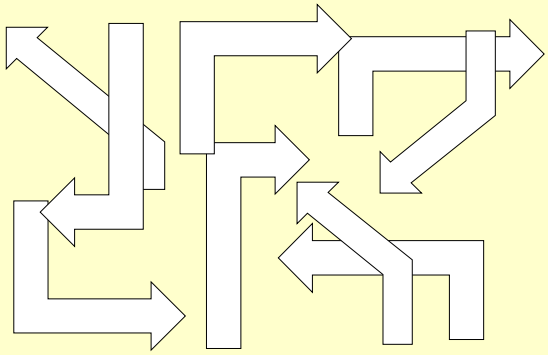
$p_1 = 31; p_2 = 37; p_3 = 43; p_4 = 47;$

$p_5 = 53; p_6 = 59; p_7 = 61; p_8 = 67;$



Permutation

En conclusion :

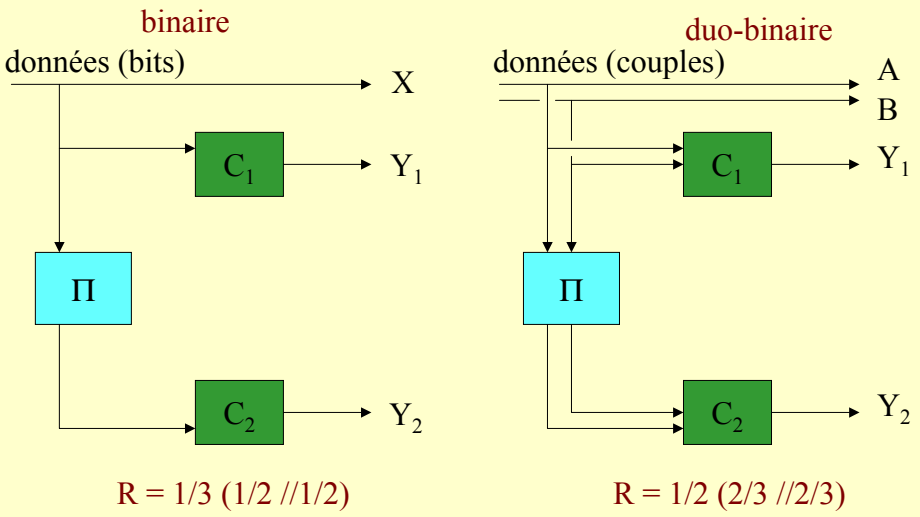


- Pas de permutation idéale pour le moment (existe-t-elle ?)
- Il n'est pas nécessaire de l'avoir

Chapitre 6

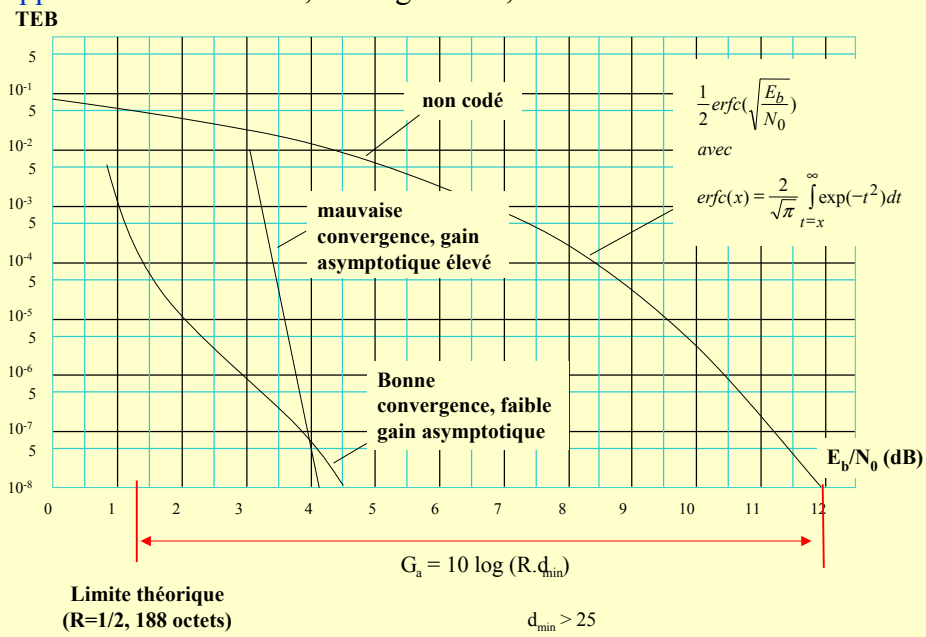
turbo codes duo-binaires

Turbo codes duo-binaires



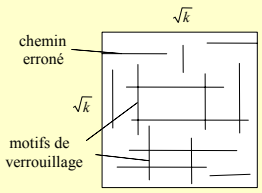
Rappel

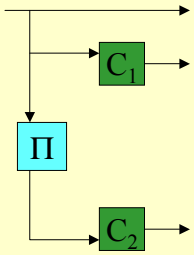
MDP4, canal gaussien, taux d'erreurs cible : 10^{-8}



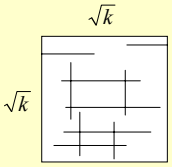
Turbo codes duo-binaires

Contribution des codes duo-binaires sur la convergence

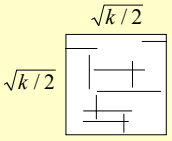


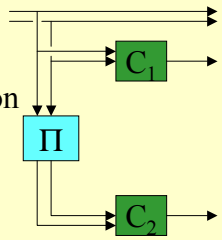


binaire



duo-binaire :
Diminue la corrélation
Durant le décodage

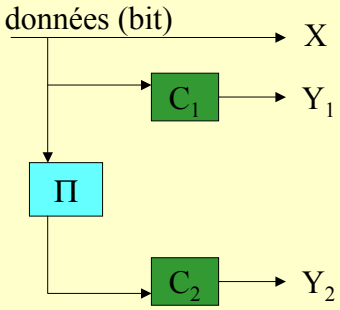




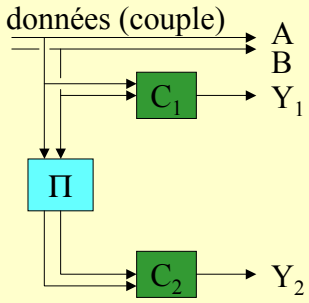
Turbo codes duo-binaires

Contribution des codes duo-binaires sur les distances minimales

Rappel : Le rôle fondamental de la permutation : si une séquence directe est RTZ, il faut minimiser la probabilité que la séquence permutée soit aussi RTZ (et vice-versa).



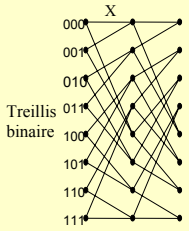
Π : permutation inter-symbole

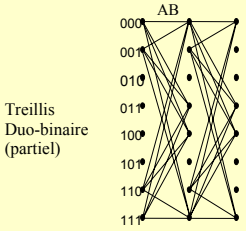


Π : permutation inter-symbole + intra-symbole

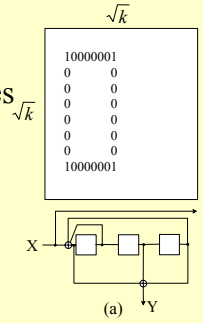
Turbo codes duo-binaires

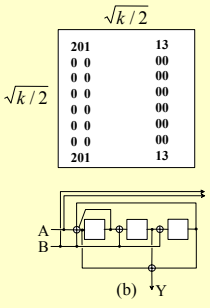
Contribution des codes duo-binaires sur les distances minimales

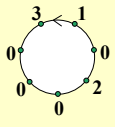




exemples de motifs à faibles distances avec une permutation régulière







00 : **0**
01 : **1**
10 : **2**
11 : **3**

Périodicités du Code duo-binaire

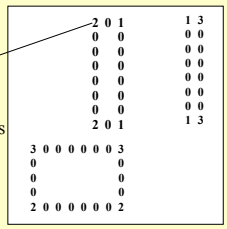
Turbo codes duo-binaires

Contribution des codes duo-binaires sur les distances minimales

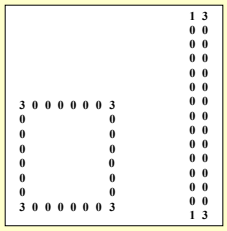
Permutation intra-symbole

(sans introduction de désordre aléatoire !)

(A,B) devient (B,A) avant le codage vertical, une fois de temps en temps



Ces motifs ne sont plus possibles si les couples sont Inversés périodiquement



Motifs d'erreurs possibles avec une grande distance

Grâce à cette technique, les distances minimales sont généralement plus grandes que celles des concaténations séries (codes produits) !!!!

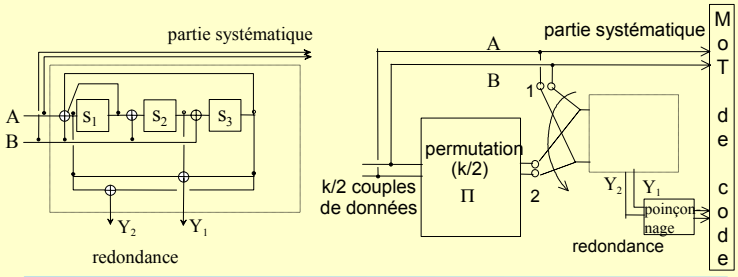
Turbo codes duo-binaires

Autres avantages des codes duo-binaires

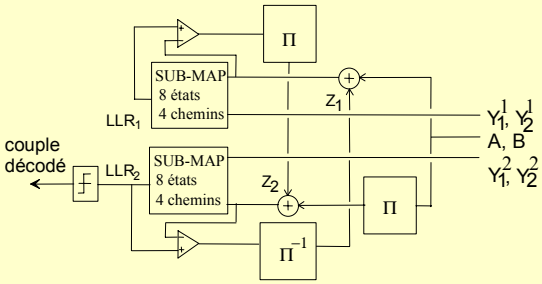
- Latence divisée par 2 (codage et décodage)
- Débit de traitement des données doublé (parallélisme intrisèque)
- Plus faible dégradation MAP → SUB-MAP
- Meilleur *mapping* pour constellations à grand nombre d'états

Turbo codes duo-binaires

Standard DBV-RCS (+RCT)



Complexité
≈ 20.000 portes
par bit.itération
@ horloge système
+ mémoire



Turbo codes duo-binaires

Standard DVB-RCS (+RCT)

Permutation générique Π

Niveau 1 (intra-couple)

if j=0 mod 2 $(A,B)=(B,A)$ [inversion du couple]

Niveau 2 (inter-couple)

if j=0 mod 4 $P=0$
if j=1 mod 4 $P=P_1+N/2$
if j=2 mod 4 $P=P_2$
if j=3 mod 4 $P=P_3+N/2$

$i=\pi(j)=P_0.j+P+1 \mod N$

P_0, P_1, P_2 et P_3 sont des entiers,
pour chaque N (nombre de couples).

Turbo codes duo-binaires

Standard DVB-RCS (+RCT)

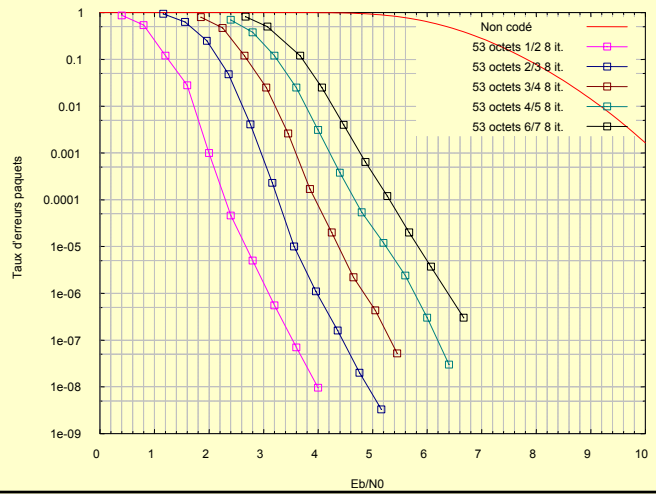
Taille des blocs : 12, 16, 53, 55, 57, 106, 107, 108, 110, 188, 212, 214, 216 octets

Rendements : 1/3, 1/2, 2/3, 3/4, 4/5, 6/7

Turbo codes duo-binaires

Standard DVB-RCS (+RCT)

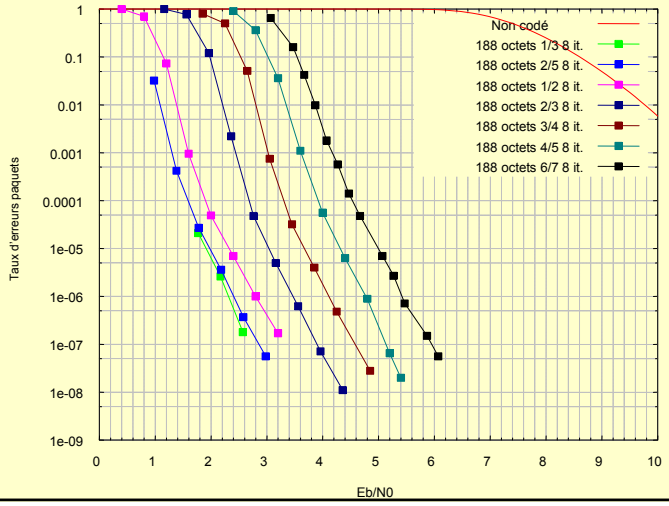
Exemple de performance : TC1000 (TurboConcept)
(Mesures, 53 octets, SUB-MAP, 4 bits de quantification, 8 itérations)



Turbo codes duo-binaires

Standard DVB-RCS (+RCT)

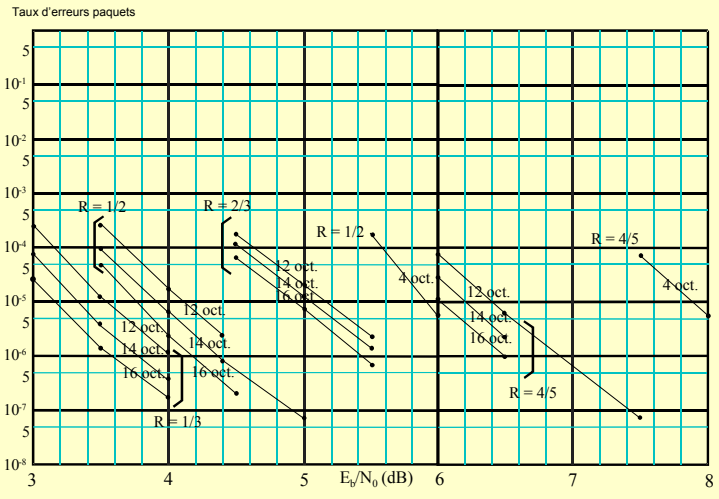
Exemple de performances : TC1000 (TurboConcept)
(Mesures, 188 octets, SUB-MAP, 4 bits de quantification, 8 itérations)



Duo-binaire turbo codes

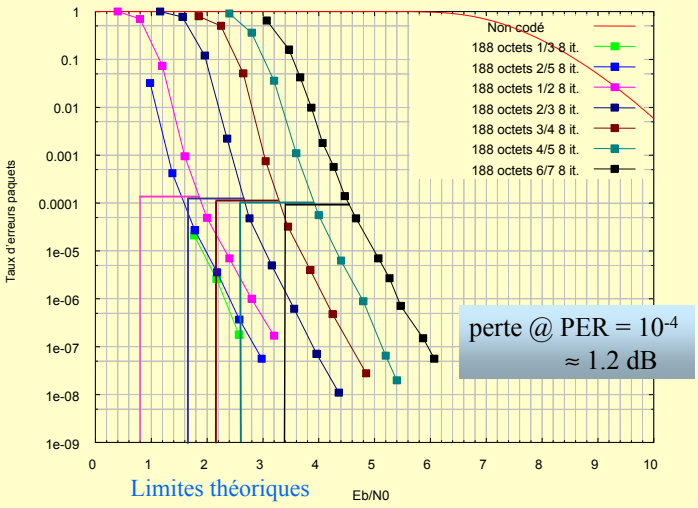
Standard DVB-RCS (+RCT)

Exemple de performances : mini-slots
(simulation, SUB-MAP, 4 bits de quantification, 6 itérations)



Chapitre 7

Conclusions et perspectives



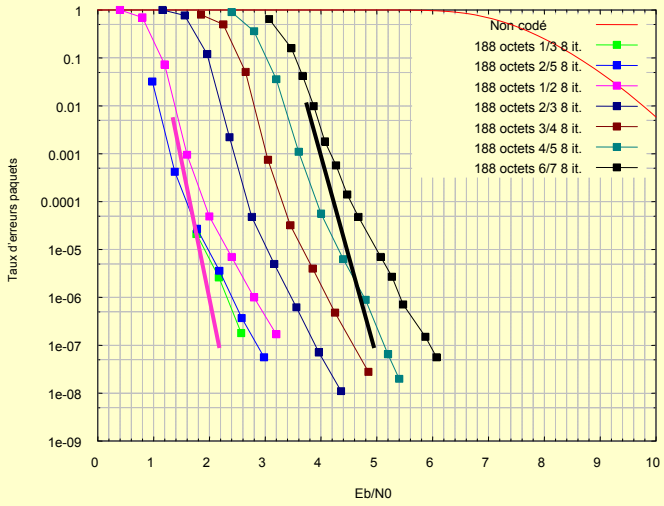
0.3 dB (turbo + it.) + 0.3 dB (SUB-MAP) + 0.15 dB (quantification) + 0.05 dB (matériel) + 0.4 dB (?)

Conclusions et perspectives — Codage

Génération suivante (2002)

1 dB gain @ TEP = 10^{-7}

pour une complexité double



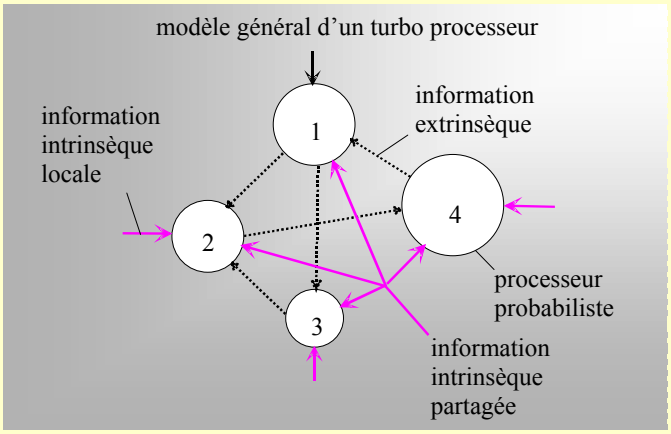
Standards actuels

Application	turbo code	terminaison	polynômes	rendements
CCSDS	binaire, 16 états	tail bits	23, 33, 25, 37	1/6, 1/4, 1/3, 1/2
IMT-2000	binaire, 8 états	tail bits	15, 13, 17	1/4, 1/3, 1/2
DVB-RCS	duo-binaire, 8 états	circulaire	15, 13	de 1/3 à 6/7
DVB-RCT	duo-binaire, 8 états	circulaire	15, 13	1/2, 3/4
Inmarsat (mini M)	binaire, 16 états	no	23, 35	1/2
Skyplex	duo-binaire, 8 états	circulaire	15, 13	4/5, 6/7

Conclusions et perspectives

turbo-X

- Turbo (dé)modulation
- Turbo égalisation
- Turbo détection (CDMA)
- Turbo synchronisation
- Turbo (dé)codage de source



Applications

turbo-égalisation,
turbo-synchro,
turbo-crypto,
...

Disques durs ?
CD-ROM ?

DVB-S ?

DSL ?

CCSDS

UMTS
CDMA2000

DVB-RCS

patents

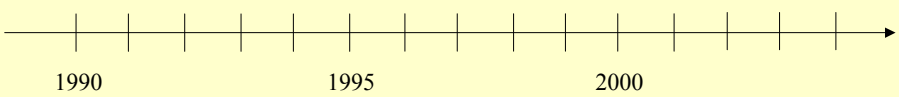
M4

DVB-RCT

invention

publication

HiperAccess



Information générale à propos des Turbo codes

- C. Heegard, S. B. Wicker, *Turbo Coding*, Kluwer Academic Publishers, 1999
- Branka Vucetic, Jinhong Yuan, *Turbo Codes, Principles and Applications*, Kluwer Academic Publishers, 2000
- C. Schlegel, *Trellis coding*, IEEE Press, 1997
- B.J. Frey, *Graphical Models for Machine Learning and Digital Communication*, MIT Press, 1998
- R. Johannesson , K. Sh. Zignagirov, *Fundamentals of Convolutional Coding*, IEEE Press, 1999
- Hanzo, Lajos, *Adaptative wireless transceivers*, John Wiley & sons, 2002
- Hanzo, Lajos, *Turbo coding, turbo equalisation and space-time coding for transmission over fading channels*, John Wiley & sons, 2002

- sites **WEB** : http://www-turbo.enst-bretagne.fr/2emesymposium/presenta/turbosit.htm
  est un bon point de départ
- base de données **IEEE Xplore** : mot clé : *turbo*, 1974 réponses le 13 mars, 2003