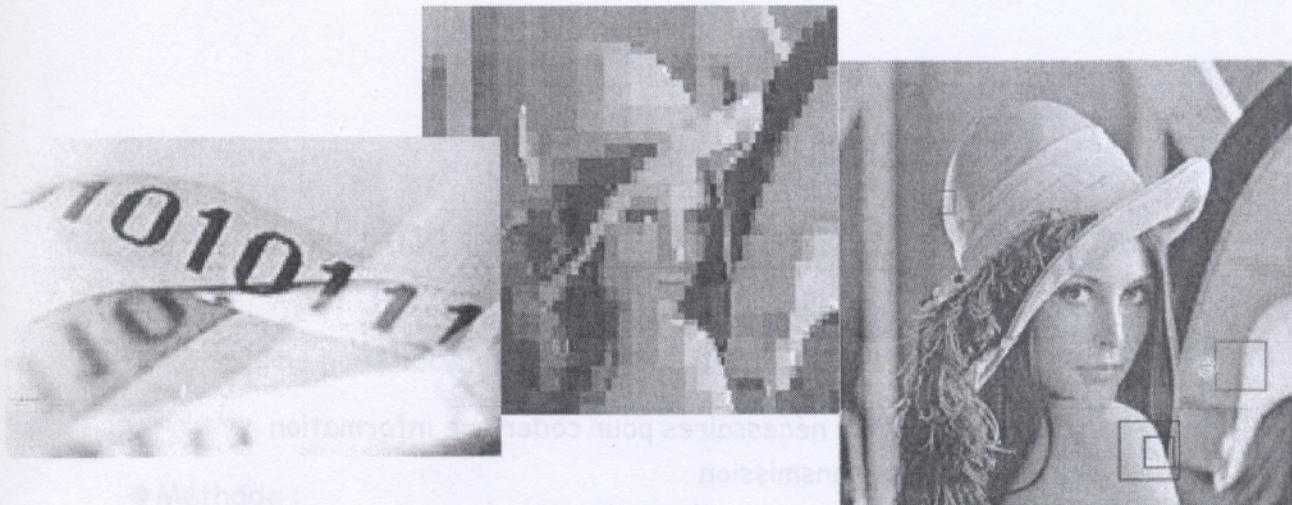
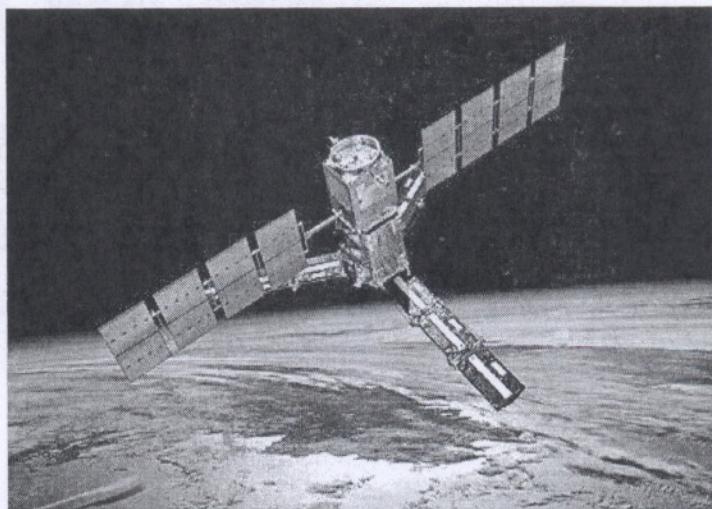
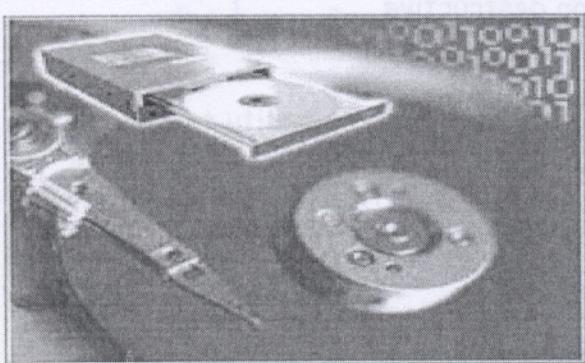


TELECOMMUNICATIONS NUMERIQUES

-Compression de données



-Correction d'erreurs



1^o Partie : Codage de source

◆ Cette étape de codage consiste essentiellement à « préparer » les données en vue de leur envoi sur un support de transmission dont la bande passante est forcément limitée.

Le traitement le plus courant consiste donc à limiter la quantité d'information à transmettre : compression de données.

◆ Compression sans perte d'information (entropique) : sources discrètes

◆ Compression avec pertes d'information : sources continues.

Domaine d'application : audio, vidéo, ...

Les points clefs de la compression de données

◆ Objectifs :

- Diminuer le nombre de bits nécessaires pour coder une information
- Limitations du canal de transmission
- Diminuer la taille nécessaire au stockage des données

◆ Faut-il préserver l'intégrité des données ?

Compression sans perte / Compression destructive

◆ Est-ce réalisable ?

- Il faut que le récepteur connaisse le procédé utilisé lors de l'encodage

(destruction du fichier)

◆ Compromis entre : taux de compression, distorsion introduite, calculs nécessaires pour l'encodage/décodage

→ Comment ça marche

↳ enlever la redondance

↳ ?

I Compression sans perte d'information

Construction du code dans le sens des bits de poids décroissant

◆ RLC

◆ Codage statistique : codage de Huffman (VRLC)

◆ Codage par dictionnaire (LZW) (.zip et .rar)

◆ Codage arithmétique (codage par intervalles)

Run Length coding (RLC)

Ce codage ne perd aucune information ; il est dit réversible.

◆ Méthode :

- Coder sur moins de bits les éléments d'apparition fréquente et sur plus de bits les éléments plus rares.
- Remplacer les suites de valeurs identiques par le nombre de fois où la valeur apparaît.

◆ Souvent utilisé en complément d'autres méthodes de compression :

Compression d'images codées avec peu de couleurs.

Run Length coding (RLC) : application

- ◆ On forme, en lisant l'image ligne par ligne, des couples {longueur de plage, intensité}. longueur de plage =

Intensité =

Méthode simple si l'image comporte de nombreuses « plages » de valeurs identiques : il est alors avantageux de coder la longueur (le nombre de symboles identiques) et la valeur de chaque plage.

C'est particulièrement vrai dans le cas de codage dans l'algorithme de type JPEG (images fixes).

- ◆ Exemple : la suite suivante correspond à des niveaux de gris prélevés
11, 11, 15, 16, 16, 16, 16, 25, 25, 25, 31, 31, 31, 31, 31, 8)

→ codage par paire : (2, 11)(1, 15)(4, 16)(3, 25)(5, 31)(1, 8)

→ le nouveau codage de la ligne : 2 11 1 15 4 16 3 25 5 31 1 8

Codage de Huffman (1952) : Principe



David Huffman (1925-1999)

- ◆ Principe : Coder les octets rencontrés dans un ensemble de données source avec des valeurs de longueur binaire variable.

Ceux ont une occurrence très faible sur une longueur binaire supérieure à la moyenne, et coder les données très fréquentes sur une longueur binaire très courte.

- ◆ L'unité de traitement est ramenée au bit.

◆ Le codage de Huffman est une méthode de compression statistique de données qui permet de réduire la longueur du codage d'un alphabet.

On observe des réductions de taille de l'ordre de 20 à 90%.

Codage de Huffman : Application

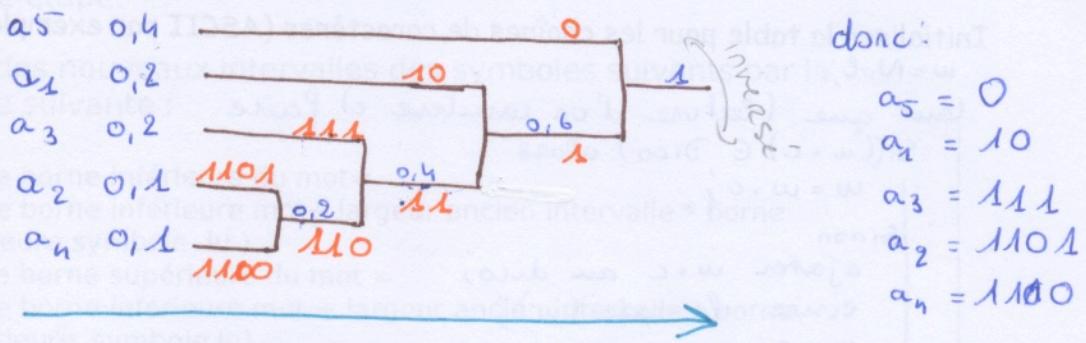
Construction du code dans le sens des bits de poids décroissant :

- ◆ 1- Classer les symboles par leur fréquences d'apparition
- ◆ 2- Regroupement des 2 moins fréquents et les coder par un n symbol (respecter l'ordre des fréquences)
- ◆ 3- Répéter 2 jusqu'à avoir qu'un symbole
- ◆ 4- Parcourir l'arbre jusqu'aux feuilles en attribuant 0 à la branche la + haute et 1 à la + basse

Arbre d'Huffman : le poids de chaque nœud est Σ des poids des fils

Codage de Huffman : Application

- ◆ Soit à coder la suite de symboles $\{a_1, a_2, a_3, a_4, a_5\}$ de probabilités respectives $\{0.2 ; 0.1; 0.2 ; 0.1 ; 0.4\}$.



On utilisera en moyenne $5/4$ bits / symboles = 1,25 bit / symb

Remarque: a_1 a la m fréq que a_3 mais ne sont pas codés sur le m nb de bits.

Codage par dictionnaire (LZW)

- ◆ Travaux de Lempel, Ziv et Welch dans les années 1980



- ◆ Compression adaptative : On ne peut pas prévoir à l'avance la taille du fichier compressé.

- ◆ Méthode :

- Stocker dans une table (ou dictionnaire) tous les mots rencontrés
- L'algorithme donne le numéro de la case où est stocké le mot
La taille du dictionnaire peut être fixe, ou agrandi au fur et mesure des nouveaux mots rencontrés. On compresse un fichier par bloc de 12K

- ◆ Essentiellement pour des fichiers de type texte ou programmes en binaires. Les images bitmap (format GIF) utilisent le codage LZW.

Codage par dictionnaire (LZW): Codage

Algorithme de compression :

Initialiser la table pour les chaînes de caractères (ASCII par exemple).

$w = N \cup \{ \}$

tant que (lecture d'un caractère c) faire

 si $(w + c) \in \text{Dico}$ alors

$w = w + c;$

 sinon

 ajouter $w + c$ au dico;

 écrire le code (w);

$w = c;$

 fin si

fin tant que;

écrire le code (w);

Codage arithmétique : Codage par intervalles

- ◆ Principe : Une séquence S de symboles est traduite par un intervalle [0;1], dont la longueur est égale à la probabilité p de S.
Pour le coder de façon optimale, il va lui attribuer ;

$$\log_2(nbr_symboles) = N \text{ bits tel que } 2^N = nbr_symboles.$$

Exemple : $nbr_symboles = 20$, $\log_2(20) = N$ soit $2^N = 20$ $N = 4,32 \text{ bits}$.

- ◆ On code une séquence de symboles dans sa globalité, pas les symboles individuellement.
- ◆ Codage utilisé dans la norme JPEG 2000, appareils photo numériques.
Conclusion

Codage arithmétique : algorithme d'encodage

- ◆ 1. Initialisation de l'intervalle de travail [0 ; 1[.
- ◆ 2. Le premier symbole est représenté par son intervalle de la première étape.
- ◆ 3. Calcul des nouveaux intervalles des symboles suivants par la méthode suivante :

- nouvelle borne inférieure du mot =
(ancienne borne inférieure mot + largeur ancien intervalle * borne inférieure symbole lu)
- nouvelle borne supérieure du mot =
(ancienne borne inférieure mot + largeur ancien intervalle * borne supérieure symbole lu)

Note : ancienne « borne inférieure » et « largeur ancien intervalle » correspondent à la ligne précédente dans le tableau :

Codage par dictionnaire (LZW) ; Décodage

Décodage : On reconstruit la table au fur et à mesure

lecture d'un caractère c ;
écrire c ;

$$w = c;$$

- Si w est dans la table de codage alors on écrit w et on met à jour la table en ajoutant c à la fin de w .
- Si w n'est pas dans la table de codage alors on écrit w et on ajoute c à la fin de w pour former un nouveau mot qui sera ajouté à la fin de la table.

- Évidemment pour des fichiers de très grande taille il faut prévoir une mémoire tampon.

Codage arithmétique : Codage par intervalles (TNT)

1980 IBM

◆ Comparaison Codage Huffman /Arithmétique

Exemple : Si les poids sont A 1, B 1, C 1 l'arbre d'Huffman sera le suivant

0 A 10 B 11 C

- A qui est pourtant aussi fréquent que B (ou C) est avantagé et on utilisera en moyenne 5/3 bits .

→ Huffman : 1,667 bits par caractère.

- Le codage arithmétique utilisera lui exactement $N = \log_2(3)$ bits par caractère .

→ Arithmétique : $2^N=3$ soit $N=1,585$ bits par caractère.

◆ On souhaite attribuer à chaque symbole, un nombre optimal de bits ; notion de codeur optimal.

Codage, utilisé dans les nouvelles normes de compression d'images.

Codage arithmétique : algorithme d'encodage

- ◆ Codage du mot PIZZA avec l'alphabet {A,I,P,Z}:

	Freq	Prob.	Intervalle
A	1	2/10	[0 0,2]
I	1	2/10	[0,2 0,4]
P	1	2/10	[0,4 0,6]
Z	2	4/10	[0,6 1]

L'intervalle [0 1] est donc fractionné en 4 sous-intervalles ...

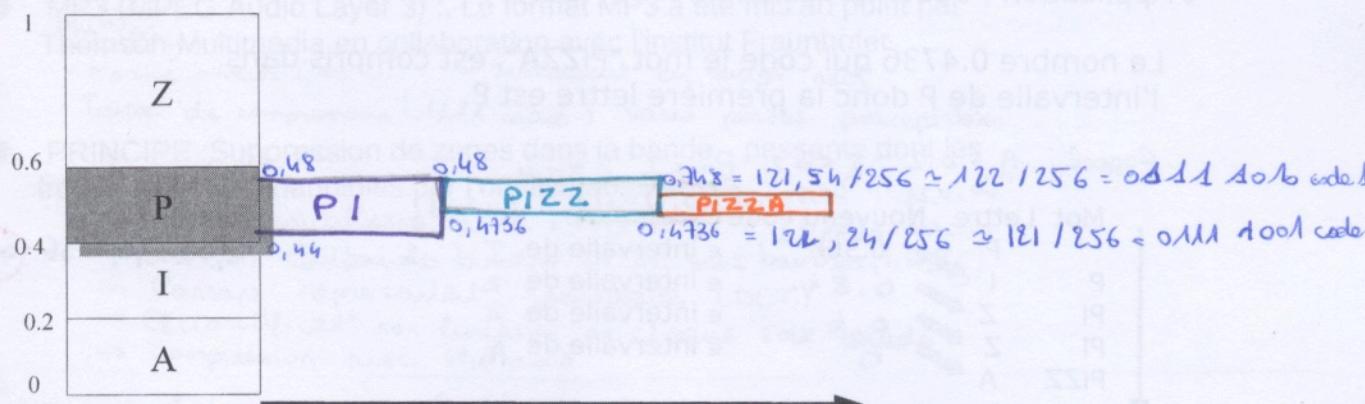
- ◆ Étapes du calcul des nouveaux intervalles :

Lettre	Borne Inférieure	Borne Supérieure
P	0.0	1.0
I	0.4	0.6 = recopie tel quel
=	0,4 + 2 * 0,2 = Binf(n-1)	$\Delta b_n(n-1) * B_{inf} Sym(n) = 0,4 + .2 * 0,4 = Binf(n-1) + \Delta b_n(n-1) * B_{sup}$
Z	0,464	0,48
Z	0,4736	0,48
A	0,4736	0,4748

- ◆ Tout nombre flottant entre et est le format compressé du mot "PIZZA".

Codage arithmétique : algorithme d'encodage

- ◆ Choix du code optimum:



Codage (sur 8 bits) de l'intervalle $[121/256 ; 121/256]$.

8 bits pour coder 5 symboles $\Rightarrow 1,6 \text{ bits/symbole}$.

Codage arithmétique : algorithme de décodage

Principe de la décompression :

◆ Postulat ; On doit connaître la table des symboles et leurs plages (entête du fichier) : A [0;0.2[, I [0.2;0.4[, P [0.4 ; 0.6[, Z [0.6;1[

◆ On répète les deux étapes suivantes jusqu'à l'obtention du mot :

- La prochaine lettre du mot est celle dont l'intervalle contient le nombre du mot actuel

- On modifie le nombre représentant le mot à l'aide du calcul :

$$\text{Code mot}_{(n)} = [\text{code mot}_{(n-1)} - \text{borne inférieure de la lettre}] / \text{intervalle lettre lue}$$

Codage arithmétique : algorithme de décodage

◆ Application ; décompression du mot PIZZA :

Le nombre 0.4736 qui code le mot "PIZZA", est compris dans l'intervalle de P donc la première lettre est P.

Rappel : A = 0,2 I = 0,2 P = 0,2 Z = 0,4
[0;0,2] [0,2;0,4] [0,4;0,6] [0,6;1]

Mot Lettre Nouveau code

P	I	0,368
P	Z	0,84
PI	Z	0,6
PI	Z	0
PIZZ	A	

ε intervalle de I [0,2;0,4]

ε

intervalle de Z

ε

intervalle de Z

ε

intervalle de A

ε

$$0,368 = 0,4736 - 0,4 / 0,6 - 0,4$$

PIZZA

→ le mot décompressé est bien "PIZZA"

Compression avec pertes d'informations

◆ Principe de la compression avec pertes

Suppression des informations les moins indispensables pour l'humain. (œil, oreille)

- Réduction du nombre de données.

- Taux de compression plus élevé.

Plus le taux de compression est élevé, plus le niveau de pertes est important et plus la qualité est dégradée.

◆ Principaux domaines d'application ; le son et les images.

Remarque : On arrive sans voir de différences à des compressions de 10 à 20.

Codage avec pertes en audio; MP3

◆ MP3 (MPEG Audio Layer 3) : Le format MP3 a été mis au point par Thomson Multimédia en collaboration avec l'institut Fraunhofer.

- Norme ouverte (au c.) - Traitement en temps réel

- Taux de compression (1/11 max) sans pertes perceptibles

◆ PRINCIPE : Suppression de zones dans la bande passante dont les fréquences sont inaudibles par l'oreille humaine.

→ Signal échantillonné à $f_e = 44,1 \text{ kHz} \rightarrow 26 \text{ ms} \text{ chacun}$

→ Division de la bande audio en 32 sous bandes (6 kHz)

→ Passage représentatif spectrale (DCT)

→ Quantification linéaire de chaque sous bande

→ Compression avec Huffman

→ Principe de compression Audio :

→ Masquage Temporel

Un élément d'un son est importante, - les autres sont entendu => masquage

→ Masquage Fréquences

Deux sons de fréquences très proches sont fusionnés en une seul fréquence

→ ADPCM : codage dynamique et tient compte de la grandeur des erreurs en modifiant le pas de quantification

→ wav

Codage avec pertes en imagerie

- ♦ Il y a deux modes de codage d'une image numérique :

- Codage avec pertes
- Codage sans pertes

Il existe de très nombreux formats de fichiers.

Ces formats diffèrent par les quantifications supportées (binaire, niveaux de gris, palette, 24 bits,...), par leur algorithmes de compression, par la volonté d'avoir un format propriétaire, ou au contraire ouvert.

- ♦ L'étude portera uniquement sur les images au format Bitmap .

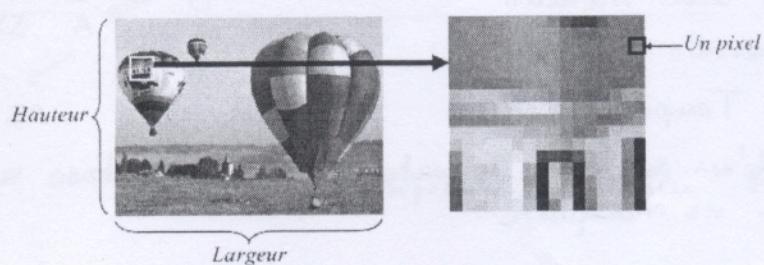
Codage avec pertes en imagerie

- ♦ Structure d'une image Bitmap :

Une image bitmap est un ensemble de points (=pixels), caractérisée par sa discréétisation et sa quantification.

- ♦ Deux grandes techniques de quantification :

- codage direct de la couleur dans le pixel.
- codage des couleurs dans une palette. Un pixel Largeur Hauteur



Codage avec pertes en imagerie ; JPEG

- ◆ JPEG signifie Joint Photographie Experts Group (Norme de 1987).
L'image est découpée en bloc de 8 x 8 points.

- ◆ Fonctionne en 4 étapes :

- Transformation et rééchantillonnage des couleurs : DCT

- Quantification de chaque bloc ; le compresseur divise chaque DCT / une "coeff de perte"

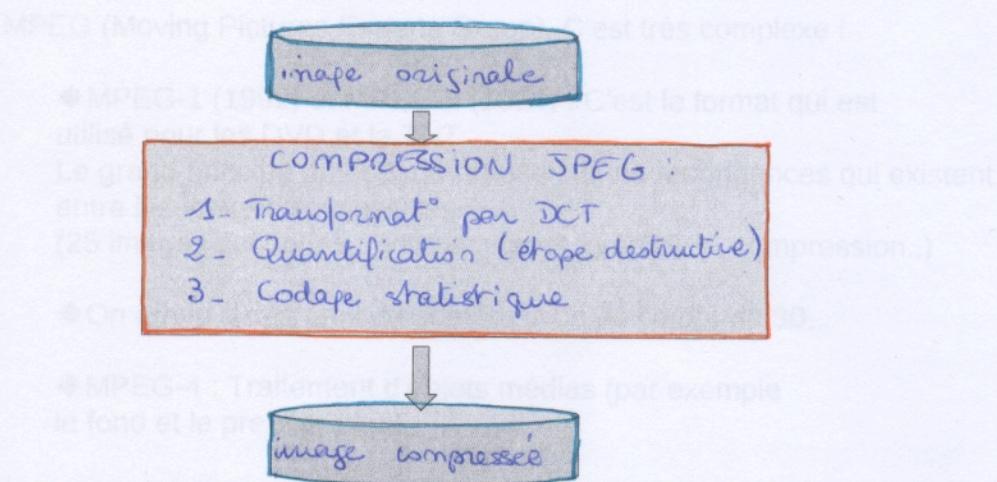
→ permet de contrôler la qualité de l'image

- Encodage : encodage et compression de Huffman.

blocs comme JPEG

Remarque : On arrive sans voir de différences à des compressions de l'ordre de 10 à 20.

Codage avec pertes en imagerie ; JPEG



Compression par ondelettes («Wavelets »)

- ◆ Évolution du format JPEG : JPEG 2000, utilisant une compression par ondelettes.
 - Qualité d'image plus élevée à fort taux de compression.
 - Taux de compression plus élevé que JPEG.
 - Possibilité de ne charger (décompresser) qu'une partie de l'image.
 - Niveaux de détail : possibilité de décompresser des versions plus petites de l'image.
 - Mode optionnel de compression sans perte.
- ◆ Méthode globale portant sur toute l'image pas d'apparition de blocs comme JPEG.

Codage avec pertes ; MPEG

MPEG (Moving Pictures Experts Group). C'est très complexe !

- ◆ MPEG-1 (1992) et MPEG-2 (1994) : C'est le format qui est utilisé pour les DVD et la TNT.
Le grand principe du codage repose sur les redondances qui existent entre les images successives.
(25 images secondes : entrelacement, prédition, compression..)
- ◆ On arrive à des taux de compression de l'ordre de 30.
- ◆ MPEG-4 : Traitement d'objets médias (par exemple le fond et le premier plan).

2^e Partie : Codage de voie

◆ Les codes linéaires : code de Hamming Binaire

◆ Les codes cycliques : CRC Cyclic Redundance Check

◆ Les codes convolutifs (récursifs ou non)

Codage de voie : Principe

◆ Objectif :

- Améliorer la qualité de la transmission de l'info,
- Sureté de fonctionnement ,
- Protéger des données (la cryptographie) ...

◆ Principe :

Introduire de la redondance (bit de parité) dans le train de bits qui constituent le message à transmettre , suivant une loi

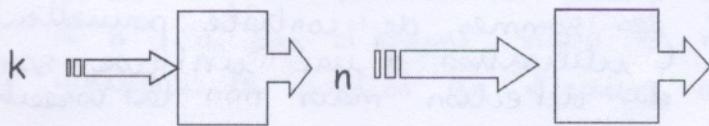
- loi de codage - .

Erreur = incohérence(s) dans le message reçu, donc détectable par le décodeur (qui connaît la loi de codage) . Possibilité d'en corriger certaines .

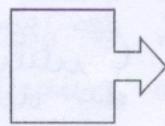
Codes linéaires : Principe

- ◆ A un bloc de k éléments binaires ('0' ou '1') d'information, (ou message m), on fait correspondre un bloc de n éléments binaires ($n > k$) ; (appelé code c).
- ◆ L'émission : Le code c est obtenu par la relation : $c = mG$
 G est appelée . matrice génératrice
- ◆ Reception : le message m est reconstitué après contrôle :
 $cH = 0$
 H est appelée matrice . de contrôle de parité

Emission : Encodage(k to n)



Reception : Contrôle de parité



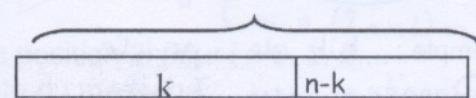
Codes linéaires ; Caractéristiques

- ◆ Soit c le code linéaire, issu du codeur de voie, noté $C(n,k)$,

n : mot de code composé de n bits

k bits d'information

$(n-k)$ bits de redondance



- ◆ Le taux de codage ou rendement (code rate) : $r = k/n < 1$

Ex de codage en redondance 0>000 , 1>111

le taux de codage=1/3

- ◆ Nbre d'erreurs (symboles) corrigibles $t = (n-k)/2$

Codes linéaires ; somme de contrôle

- ◆ La somme de contrôle (« checksum »), est un nombre qu'on ajoute à un message à transmettre afin de permettre au récepteur de vérifier que le message reçu est bien celui qui a été envoyé.
L'ajout d'une somme de contrôle à un message est une forme de contrôle par redondance.

ascii	H e L L O	H e l l a
checksum	48 65 6C 6C 6F F4 mod FF	48 65 6C 6C 61 E6 mod FF

- ◆ les codes utilisant les sommes de contrôle permettent de valider un message. L'utilisation d'une unique somme de contrôle permet la détection mais non la correction des erreurs.

Sommes de contrôle ; bit de parité

- ◆ Cas particulier répandu dans l'industrie.
Un Bit mis à **zéro** si la somme des autres bits est **paire**, et à **un** si elle est **impaire**, pour détecter des erreurs de transmission.

Exemple : **bit de parité'**

Données sur 7 bits

0000000
1010001
1101001
1111111

Parité paire

0 0000000
1 1010001
0 1101001
1 1111111

- ◆ Les messages envoyés sur huit bits ont toujours la parité zéro.
Si une erreur se produit, un zéro devient un un, ou l'inverse ; le récepteur sait qu'une altération a eu lieu.

- ◆ Dans un cas d'une transmission série de type RS-232, le bit de poids faible est transmis en premier (lsb) ↗ bit de poids fort (msb) et ensuite le bit de parité.

Codes linéaires ; Codes de Hamming



Richard Hamming (1915-1998)
A collaboré avec Shannon et au projet Manathan

- ◆ Principe ; Association de plusieurs sommes de contrôle.
- ◆ Utilisés en télécommunication pour compter le nombre de bits altérés dans la transmission d'un message d'une longueur donnée.
- ◆ Un code de Hamming permet la détection et la correction automatique d'une erreur si elle ne porte que sur un seul élément du message.
- ◆ Retrouver l'erreur dans un mot de Hamming

Si les bits de contrôle de parité C_2, C_1, C_0 ont tous la bonne valeur il n'y a pas d'erreurs ; sinon la valeur des bits de contrôle indique la position de l'erreur entre 1 et 7.

Codes linéaires ; Codes de Hamming

◆ Structure d'un code de Hamming

m bits du message à transmettre
n bits de contrôle de parité → longueur totale : $2n - 1$
 $(m = (2n - 1) - n)$

Les bits de contrôle de parité C_i sont en position $2i$ pour $i = 0, 1, 2, \dots$

Les bits du message D_j occupent le reste du message.

◆ Structure d'un code de Hamming(7:4)

Rang : 7 6 5 4 3 2 1
Type de donnée : D₃ D₂ D₁ C₂ D₀ C₁ C₀

Ce code a un coefficient d'efficacité de $4/7 = 57\%$
Il ne permet de retrouver et corriger qu'une erreur.

Codes linéaires ; Codes de Hamming

Code de Hamming(7:4) :

Il y a $2^{n-k} = 8$ configurations du syndromes :

- une configuration sans erreur => **syndrome nul**
- 7 configurations d'erreurs => **syndrome non nul**

Le récepteur va vérifier que le message reçu est bien celui qui a été envoyé.

Syndromes	000	001	010	100	110	101	011	111
Erreurs e		000001	000010	0001000	100000	010000	100000	

↓
Pas d'erreur ↓

Codes cycliques : (CRC)

◆ Ce type de code possède non seulement la capacité de détecter les erreurs, mais aussi de les corriger sous réserve d'altérations modérées.

◆ Représentation polynomiale : chacun des mots du code (c) est un multiple du polynôme générateur $g(x)$ par $m(x)$:

$$c(\alpha) = m(\alpha) g(\alpha)$$

$$\text{tel que } g(\alpha) h(\alpha) = \alpha^n + 1$$

- Degré de $m(\alpha)$: au plus $k-1$ (k éléments dans le msg)
- $c(\alpha)$: $n-1$ (n _____ code)



Degré de $g(x)$ = Degré $c(x)$ + Degré $m(x)$

$$\text{- Le message } m = [m_{(k-1)} \dots m_2, m_1, m_0] \iff m(x) = m_{(k-1)}x^{k-1} + \dots + m_2x^2, m_1x + m_0$$

$$\text{- Le polynôme générateur } g = [g_{(n-k)} \dots g_2, g_1, g_0] \iff g(x) = g_{(n-k)}x^{n-k} + \dots + g_2x^2 + g_1x + g_0$$

Codes cycliques : (CRC)

◆ Forme systématique : (pour la mise en œuvre pratique)

Les éléments binaires d'information ($m(x)$) sont séparés des éléments binaires de redondance ($r(x)$)

L'information est dans les degrés élevés du polynôme

◆ Le mot de code $c(x)$ d'un code polynomial (n,k) de polynôme générateur $g(x)$ (de degré $n-k$) et de diviseur de $x^n + 1$, associé au mot initial $m(x)$ est défini par :

$$c(x) = m(x) \cdot x^{n-k} + r(x)$$

- $r(x)$ reste de la division de $m(x)x^{n-k}$ par $g(x)$

- $g(x)$ polynôme générateur de code cyclique

Codes cycliques : Application (CRC5)

◆ Mise en œuvre du codeur CRC5 utilisé dans les protocoles de communication USB2 et en RFID sur des mots de 8 bits, avec comme polynôme générateur ; $x^5 + x^2 + 1$; $g=(100101)$.

Pour simplifier soit m le message à coder = (1001)
Convention d'écriture $m(msb \dots lsb)$

Il y aura donc $n-k = 5-4 = 1$ élément de redondance dans le code

$$m(x) = x^3 + 1 \text{ et } g(x) = (x^5 + x^2 + 1)$$

$$c(x) = m(x)g(x) = (x^3 + 1)(x^5 + x^2 + 1) = x^8 + x^3 + x^2 + 1$$

degré de $c = n-k = 8 \Rightarrow c = (100\ 001\ 101)$ msg noyé dans le code

◆ Mise sous forme systématique $c(x) = (x^3 + 1)x^5 + r(x)$

$$\text{Tel que } m(x)x^{n-k}/g(x) = x^8 + x^3/x^5 + x^2 + 1$$

$$\Rightarrow r(x) = x^3 \text{ et } k(x) = x^3$$

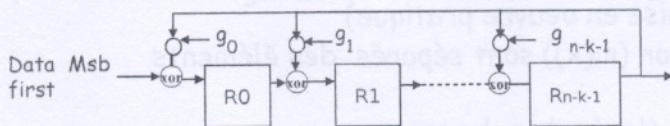
$$\text{d'où } c(x) \text{ systématique} = x^8 + x^5 + x^3$$

$$c \text{ systématique} = [100101000]$$

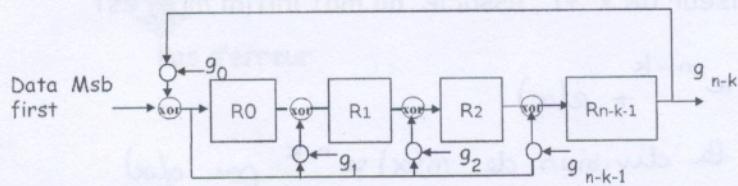
msg sur poids fort (gfsl)

Codes cycliques : Application (CRC5)

- ◆ Schéma de principe d'un diviseur polynomial



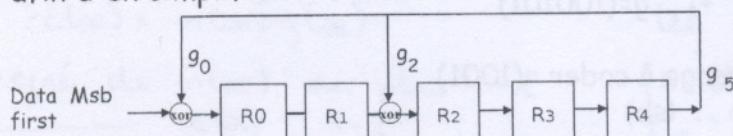
- ◆ Structure générique d'un codeur série



$$g(x) = g_5x^5 + g_4x^4 + g_3x^3 + g_2x^2 + g_1x^1 + g_0x^0$$

Codes cycliques : Application (CRC5)

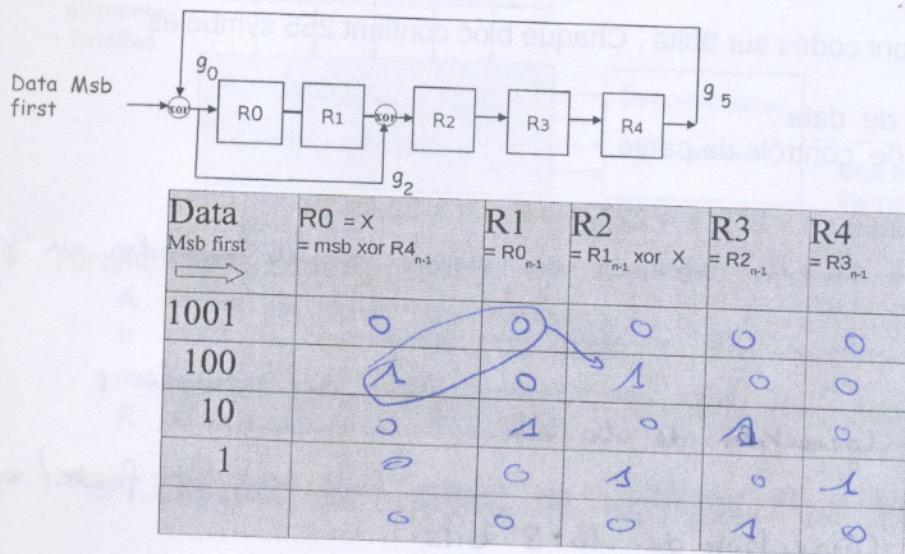
- ◆ Mise en œuvre du diviseur polynomial appliquée au code CRC5 avec comme polynôme générateur $x^5 + x^2 + 1$, et un mot de 4 bits afin d'en simplifier l'étude .



Data	R ₀ = msb xor R ₄ _{n-1}	R ₁ = R ₀ _{n-1}	R ₂ = R ₁ _{n-1} xor R ₄ _{n-1}	R ₃ = R ₂ _{n-1}	R ₄ = R ₃ _{n-1}
1001	0	0	0	0	0
100	1	0	0	0	0
10	0	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
0	0	1	0	0	1
0	1	0	1	0	0
0	0	1	0	1	0
0	0	0	1	0	0
0	0	0	0	1	0

Codes cycliques : Application (CRC5)

- ◆ Mise en œuvre du codeur CRC5 série avec comme polynôme générateur $x^5 + x^2 + 1$, et un mot de 4 bits afin de simplifier l'étude.



Codes cycliques : code de Reed-Solomon



Gustave Solomon (1930-1996)
Irvin Reed (1923/2012)

Mis au point dans les années 60

- ◆ Les codes de Reed Solomon sont des codes à éléments non binaires

- Chaque élément (symbole) q du code est une puissance de deux : $q = 2^m$.

- ◆ Principe : Le message est divisé en blocs ; k coefficients du polynôme, puis les $2t$ informations de redondance sont calculées.

Code : n symboles q _aires

K voff de $m(\alpha)$ (m bits chacun) & t symboles de redondance

- ◆ Paramètres du code RS(n, k, t) :

Longueur d'un code RS $n = 2^n - 1$ $n = \text{nbre Total de symb.}$
 $k = n - 2t$ $k = \text{nbre de symb du msg}$

Codes cycliques : code de Reed-Solomon

- ◆ Le plus populaire ; RS(255,223), qui est utilisé notamment pour coder les trames internet.

Les symboles sont codés sur 8bits . Chaque bloc contient 255 symboles (octet) :

- 223 octets de data .
- 32 octets de contrôle de parité.

- ◆ Code RS(255,223) : $n = 255$, $k = 223$, $s = 8$ $2t = 32$, $t = 16$.

Une enem = 16 bits séparés en enem ou 16 symbols en exten par bloc .

Il permet de corriger 16 erreurs par blocs :

→ Pire des cas: 16 bits en erreur dans des symbols + correction de 16 bits

→ Cas idéal : 16 symbols en erreur (un seul bit faute/ symbol)
correction de 16×8 bits

⇒ gain de 10dB fourni par les codes RS (en dB)

Codes convolutifs

Le codeur qui engendre un code convolutif comporte un **effet de mémoire**

Chaque bloc de n éléments (binaire) en sortie du codeur dépend :

- du bloc de k éléments présents en entrée du codeur
- des $(m-1)$ blocs de k éléments présents précédemment .

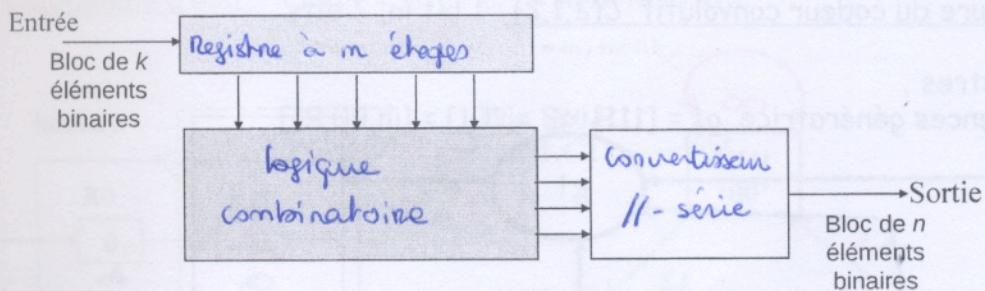
- ◆ Il existe deux types de structure ; une récursive ou non récursive. On se limitera ici, à l'étude non récursive.

- ◆ Chaque sortie est le produit (de convolution) entre l'entrée du codeur et la réponse du codeur, définie par des séquence génératrices.

- ◆ Ce sont les codes les plus utilisés dans les communications fixes et mobiles.

Codes convolutifs

1955



- ◆ Paramètres du code $C(n,k,m)$:

$$n = \text{nbre de bits en sortie}$$

$$k = \text{nbre de bits en entrée}$$

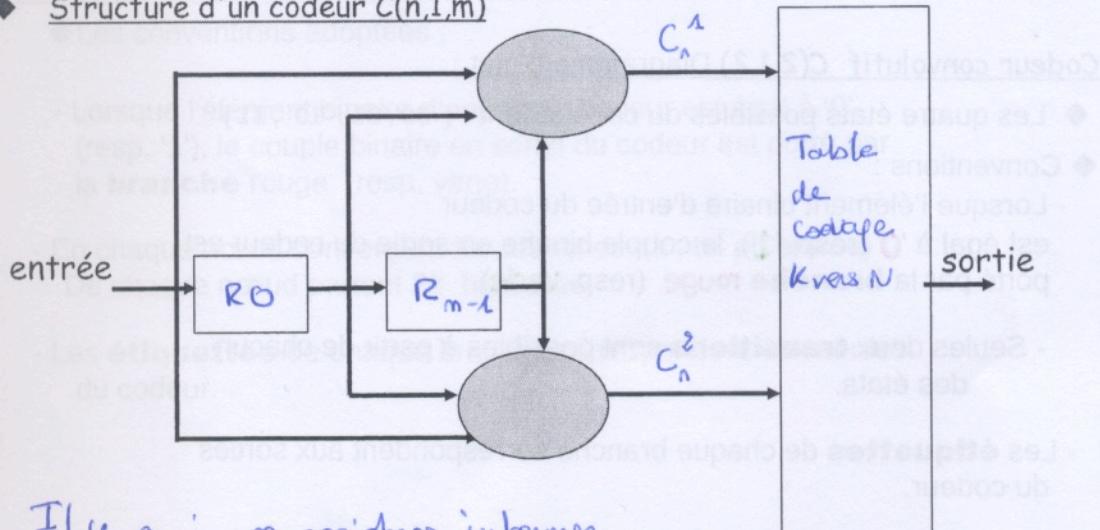
$$m = \text{nbre de registres}$$

$$R = \text{rendement} = k/n$$

la longueur de contrainte du code est : $m \cdot k$

Codes convolutifs: Structure génératrice

- ◆ Structure d'un codeur $C(n,1,m)$



Il y a :

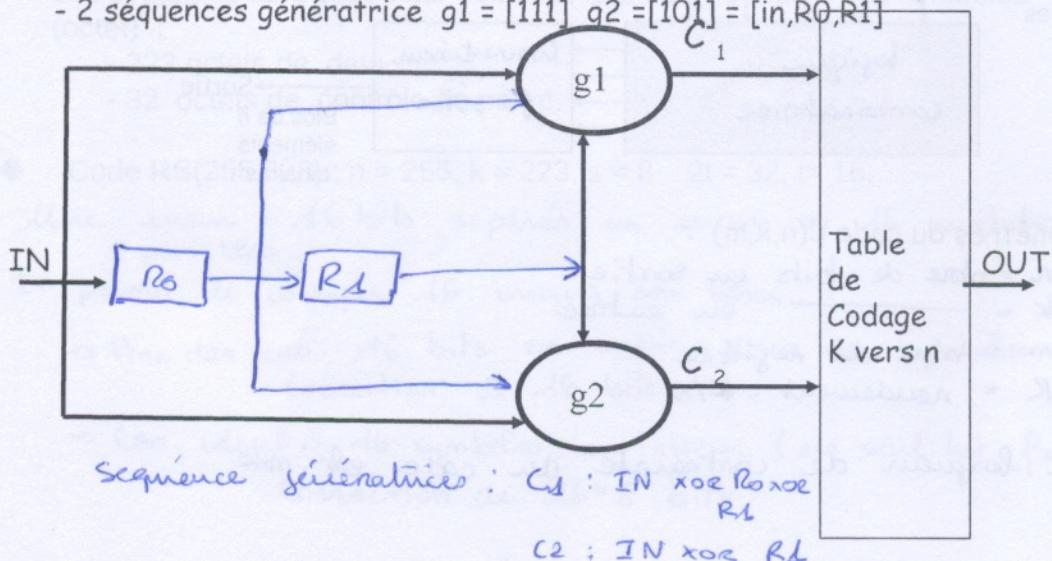
- m registres internes
- n séquences génératrices C_n^i

Codes convolutifs: Structure appliquée

- ◆ Structure du codeur convolutif $C(2,1,2)$: 1 bit in, 2 bits out

- 2 Registres ,

- 2 séquences génératrice $g_1 = [111]$ $g_2 = [101] = [in, R_0, R_1]$



Codes convolutifs: Représentations graphiques :

Le codeur qui engendre un code convolutif comporte les étages de mémoire

Codeur convolutif $C(2,1,2)$ Diagramme d'état :

- ◆ Les quatre états possibles du codeur sont : { '00', '01', '10', '11' }

◆ Conventions :

- Lorsque l'élément binaire d'entrée du codeur est égal à '0' (resp. '1'), le couple binaire en sortie du codeur est porté par la **branche rouge** (resp. **verte**).

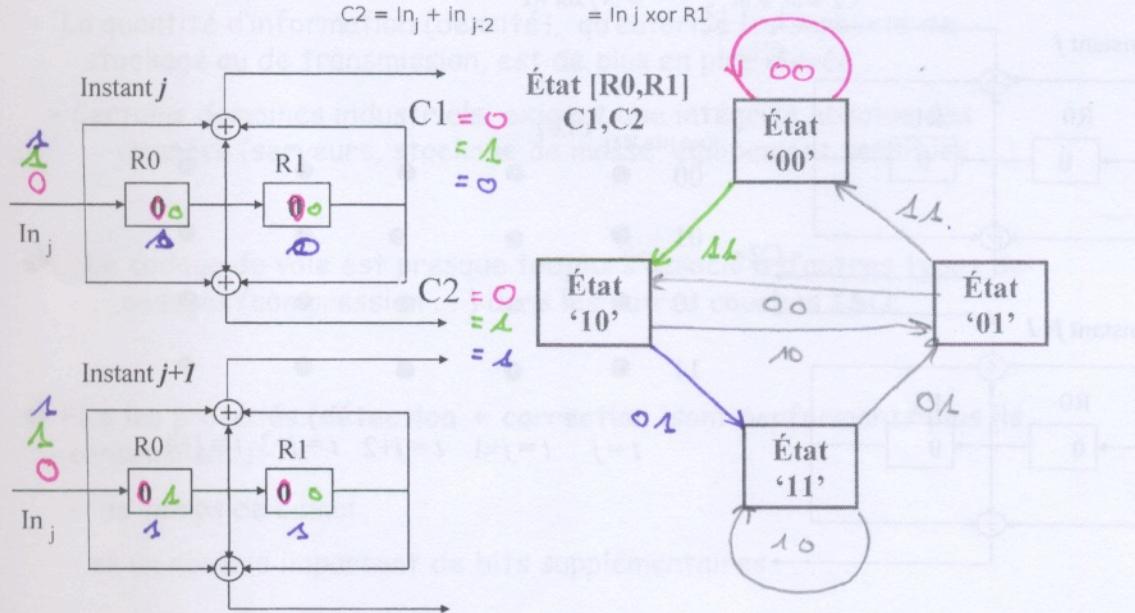
- Seules deux **transitions** sont possibles à partir de chacun des états.

- Les **étiquettes** de chaque branche correspondent aux sorties du codeur.

Codes convolutifs: Représentations graphiques :

Diagramme d'état: $C_1 = In_j + In_{j-1} + in_{j-2} = In_j \text{ xor } R_0 \text{ xor } R_1$

$$C_2 = In_j + in_{j-2} = In_j \text{ xor } R_1$$



Codes convolutifs: Représentations graphiques :

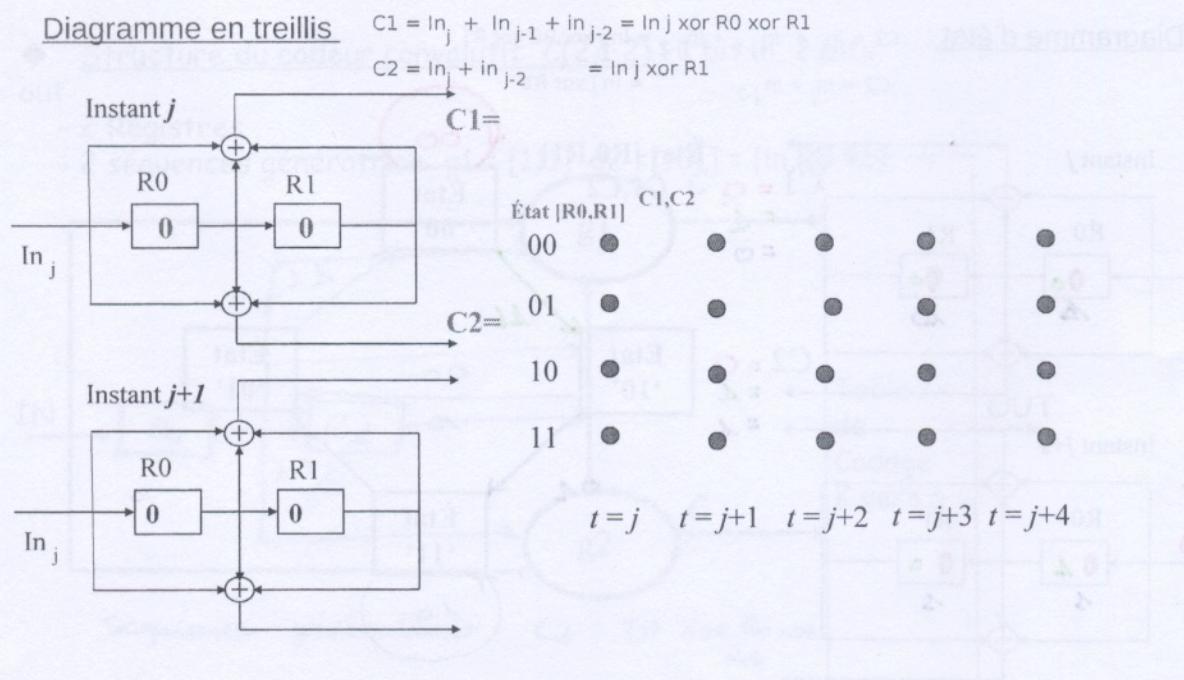
Codeur convolutif $C(2,1,2)$ Diagramme en treillis:

◆ Les conventions adoptées :

- Lorsque l'élément binaire d'entrée du codeur est égal à '0' (resp. '1'), le couple binaire en sortie du codeur est porté par la **branche** rouge (resp. verte).
- En chaque nœud convergent $2k$ branches. (k ; bit en entrée)
De chaque nœud partent $2k$ branches.
- Les **étiquettes** de chaque branche correspondent aux sorties du codeur.

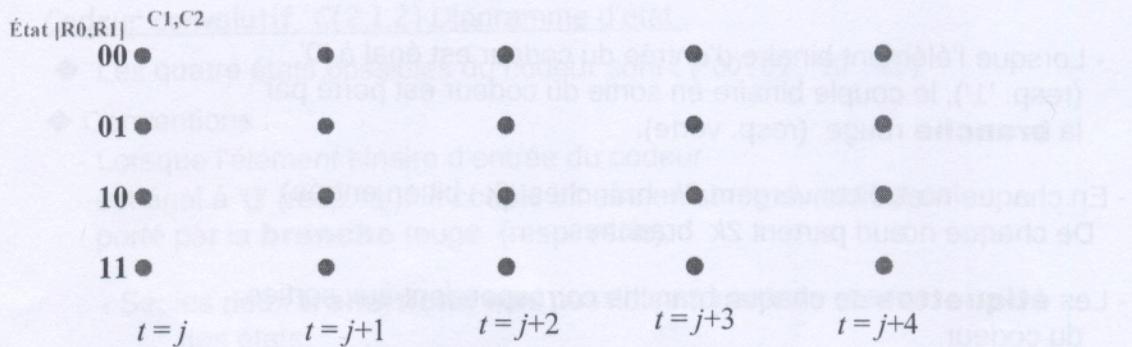
Codes convolutifs: Représentations graphiques :

Diagramme en treillis



Codes convolutifs: Représentations graphiques :

Diagramme en treillis : encodage élémentaire



Conclusion Codage de voie

- ◆ La détection des erreurs, est d'autant plus essentielle que :
 - La quantité d'information (densité), qu'autorise les supports de stockage ou de transmission, est de plus en plus élevée ..
 - Certains domaines industriels, exigent une intégrité absolue des données (serveurs, stockage de masse, équipement sensibles ..)
- ◆ Le codage de voie est presque toujours associé à d'autres types de codages (compression ..) dans les autres couches ISO.
- ◆ Plus les procédés (**détection + correction**) sont performants plus ils consomment ;
 - du temps de calcul
 - et un nombre important de bits supplémentaires