Codes détecteurs et correcteurs

B. Rouzeyre

Bibliographie

- Codes correcteurs, principes et exemples
  - J. Badrikian, Technosup, Ellipses (cours et exercices)
- Codes correcteurs : Théorie et applications
  - A. Poli, Li. Huguet, Masson (cours)

- **Codes détecteurs et/ou correcteurs**
  - **Codes en blocs linéaires**

    Parité,

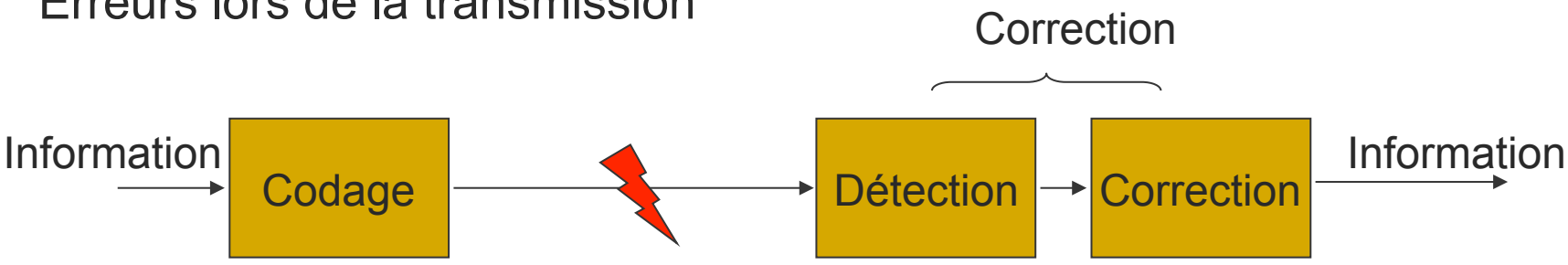
    Code de Hamming,

    Codes cycliques

    CRC/FCS, code BCH, Reed-Salomon

Introduction

- Erreurs lors de la transmission



- Information découpée en blocs de **k** bits (mots d'information)
- Blocs codés de longueur **n** supérieure à **k**
- Longueur constante n
- Chaque mot de code ne dépend que du mot d'information
- (n-k) : redondance de code
- ρ = k/n : rendement du code
- La fonction de codage doit être injective
  - 2 informations distinctes donnent 2 codes distincts
  - certains mots reçus ne sont pas des codes

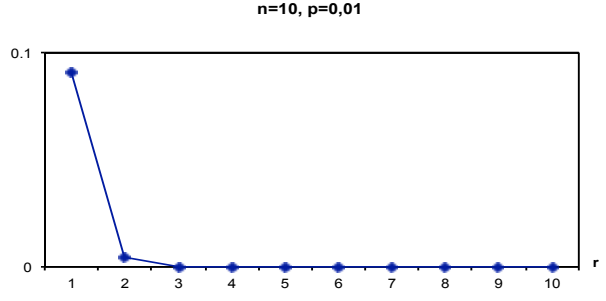
Introduction

- Pour un codage de blocs de k bits vers des blocs de n bits, il y a :
  - 2^k mots d'info à coder
  - 2^n messages reçus possibles de longueur n
    - dont 2^k sont des mots du code
    - et donc (2^n - 2^k) qui ne sont pas corrects

Rappel

p = probabilité qu'un bit soit mal transmis

$$P_{\text{r bits erronés parmi n}} = C_n^r . p^r . (1-p)^{n-r}$$



- Conclusion : si on détecte une erreur, on corrige vers le code demandant le moins de changements, càd le plus "proche" => correction "optimiste"

Introduction

- Codes systématiques
  - Codage : construction du mot de code en ajoutant à la suite des k bits d'information $i_1,i_2,\ldots,i_k$, s=(n-k) bits $l_1,..,l_s$ bits appelés bits de contrôle formant la clé de contrôle on obtient $c_1,\ldots..c_k,\ldots,c_n = i_1,i_2,\ldots,i_k,l_1,..,l_s$
  - Contrôle : les k premiers bits étant les bits d'information, on recalcule à l'arrivée la clé de contrôle $c_{k+1},\ldots,c_n$ à l'aide de $c_1,..,c_k$
    - si idem : ok
    - sinon erreur

Exemple 1 : code de parité

- Transmission d'un texte : 1 caractère = 7 bits (Ascii 0-127)
- Bloc = caractère
- On ajoute un 8^{e} bit / le nombre total de '1' soit pair (ou impair)

  ex (parité paire) : 1 1 0 0 1 0 1 => 1 1 0 0 1 0 1 0

  1 1 0 0 1 1 1 => 1 1 0 0 1 1 1 1

- Les 2^{7}=128 caractères sont codés parmi les 2^{8}=256 messages possibles
- Il s'agit d'un code systématique

Exemple 1 : code de parité (contrôle)

- Réception : la règle de codage est appliquée et la parité du résultat est comparée au 8^{eme} bit reçu

  Ex : 1 1 0 0 1 0 1 1 => incorrect (le 8eme bit devrait être 0)

- Remarques
  - on ne sait pas d'où vient l'anomalie
  - lorsque le nombre de bits erronés est pair, il n'y a pas de détection

  ex : si 1 1 0 0 1 0 1 0 devient 0 0 1 1 0 1 0 1 la communication semble correcte et pourtant 8 erreurs (on a reçu un mot de code)
  - la détection d'anomalies ne permet pas de détecter le nombre d'anomalies

  ex : si 1 1 0 0 1 0 1 0 devient 0 0 1 1 0 1 0 0, on ne pas savoir qu'il y a eu 7 erreurs

- code parité => code détecteur d'un nombre impair d'erreurs mais ne permet pas de corriger des erreurs

Exemple 1 : probabilité de détection

- p : probabilité d'erreur sur chaque bit (hyp constante quel que soit le bit),
- q = 1-p : probabilité de transmission correcte d'un bit

$$P(\text{k erreurs parmi n}) = P(X = k) = C_8^k p^k (1-p)^{8-k}$$

$$P(\text{transmission parfaite}) = P(X = 0) = (1-p)^8 = q^8$$

$$p_{err} = 1 - q^8$$

$$p_{det} = P(\text{une anomalie visible}) = P(X = 1 \bigcup X = 3 \bigcup X = 5 \bigcup X = 7)$$

$$= P(X = 1) + P(X = 3) + P(X = 5) + P(X = 7) = C_8^1 p^1 q^7 + C_8^3 p^3 q^5 + C_8^5 p^5 q^3 + C_8^7 p^7 q^1$$

Pour $p = 0,1$

$p_{det} = 0,42$ => 42/57 càd 74% des messages erronés sont détectés

$p_{err} = 0,57$

Pour $p = 0,01$

$p_{det} = 0,0746$ => 97% des messages erronés sont détectés

$p_{err} = 0,0772$

Exemple 2 : Code de parité croisée

- Bloc : suite de L caractères => suite de 7L bits
- Codage : les L caractères sont rangés dans un tableau de L lignes et 7 colonnes. On ajoute une parité horizontale et une parité verticale

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$	$a_{1,8}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$	$a_{2,8}$
…	…	…	…	…	…	…	…
$a_{L,1}$	$a_{L,2}$	$a_{L,3}$	$a_{L,4}$	$a_{L,5}$	$a_{L,6}$	$a_{L,7}$	$a_{L,8}$
$a_{L+1,1}$	$a_{L+1,2}$	$a_{L+1,3}$	$a_{L+1,4}$	$a_{L+1,5}$	$a_{L+1,6}$	$a_{L+1,7}$	

Code de parité croisée

- Décodage

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$	$a_{1,8}$	0
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$	$a_{2,8}$	0
…	…	…	…	…	…	…	…	0
$a_{L,1}$	$a_{L,2}$	$a_{L,3}$	$a_{L,4}$	$a_{L,5}$	$a_{L,6}$	$a_{L,7}$	$a_{L,8}$	…
$a_{L+1,1}$	$a_{L+1,2}$	$a_{L+1,3}$	$a_{L+1,4}$	$a_{L+1,5}$	$a_{L+1,6}$	$a_{L+1,7}$	$a_{L+1,8}$	0
0	0	0	0	0	0	0	0	

- Un 1 indique une erreur

Code de parité croisée : un bit erroné

- Décodage

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$	$a_{1,8}$	0
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$	$a_{2,8}$	1
…	…	…	…	…	…	…	…	0
$a_{L,1}$	$a_{L,2}$	$a_{L,3}$	$a_{L,4}$	$a_{L,5}$	$a_{L,6}$	$a_{L,7}$	$a_{L,8}$	…
$a_{L+1,1}$	$a_{L+1,2}$	$a_{L+1,3}$	$a_{L+1,4}$	$a_{L+1,5}$	$a_{L+1,6}$	$a_{L+1,7}$	$a_{L+1,8}$	0
0	0	0	1	0	0	0	0	

- Tous les messages ne contenant qu'une erreur sont détectés et peuvent être corrigés (un 1 sur une ligne ou une colonne)

Code de parité croisée : deux bits erronés

- Décodage

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$	$a_{1,8}$	0
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$	$a_{2,8}$	0
…	…	…	…	…	…	…	…	0
$a_{L,1}$	$a_{L,2}$	$a_{L,3}$	$a_{L,4}$	$a_{L,5}$	$a_{L,6}$	$a_{L,7}$	$a_{L,8}$	0
$a_{L+1,1}$	$a_{L+1,2}$	$a_{L+1,3}$	$a_{L+1,4}$	$a_{L+1,5}$	$a_{L+1,6}$	$a_{L+1,7}$	$a_{L+1,8}$	0
0	0	0	1	0	0	1	0	

- Pour 2 erreurs sur une même ligne ou même colonne : détection mais pas de correction possible (quelle ligne ?)

Code de parité croisée : deux bits erronés

- Décodage

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$	$a_{1,8}$	0
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$	$a_{2,8}$	1
…	…	…	…	…	…	…	…	0
$a_{L,1}$	$a_{L,2}$	$a_{L,3}$	$a_{L,4}$	$a_{L,5}$	$a_{L,6}$	$a_{L,7}$	$a_{L,8}$	1
$a_{L+1,1}$	$a_{L+1,2}$	$a_{L+1,3}$	$a_{L+1,4}$	$a_{L+1,5}$	$a_{L+1,6}$	$a_{L+1,7}$	$a_{L+1,8}$	0
0	0	0	1	0	0	1	0	

- Pour 2 erreurs sur des lignes distinctes : détection mais pas de correction possible (2 possibilités)

Rappels mathématiques

- GF(2) = {(0,1),.,⊕} = {(0,1), et, ouex}
  - Addition, multiplication internes
  - Structure de corps (commutatif) : Corps de Gallois (GF : Gallois Field)

⊕	0	1
0	0	1
1	1	0

et

•	0	1
0	0	0
1	0	1

- Remarque : "+" équivaut à "-" => 1 ⊕ 1 = 0
- Notations :
  - ⊕ ⇔ +
  - . ⇔ x

Rappels mathématiques

- V = $[GF(2)]^n$ espace vectoriel de dimension n sur GF(2) :
  - un suite de n bits (011100….110) peut être considéré comme un vecteur de V à n composantes
  - multiplication externe par $\lambda \in \{0,1\}$
  - addition/soustraction de vecteurs (structure de groupe, loi $\oplus$)
    - ex :
      - (1 0 1 0 1 1) + (1 1 1 0 0 1) = (0 1 0 0 1 0)
      - (1 0 1 0 1 1) - (1 1 1 0 0 1) = (0 1 0 0 1 0)
  - produit scalaire
    - <(1 0 1 0 1 1) , (1 1 1 0 0 1)> = 1.1+0.1+1.1+0.0+1.0+1.1 = 1+0+1+0+0+1 = 1
  - Base canonique :
    - e1 = (1 0 0 ……0)
    - e2 = (0 1 0 ……0)
    - …..
    - en = (0 0 0 ……1)

Rappels mathématiques

- Rappels

Soit E un e.v. de dimension n,

Soit F un s.e.v. de E de dimension k

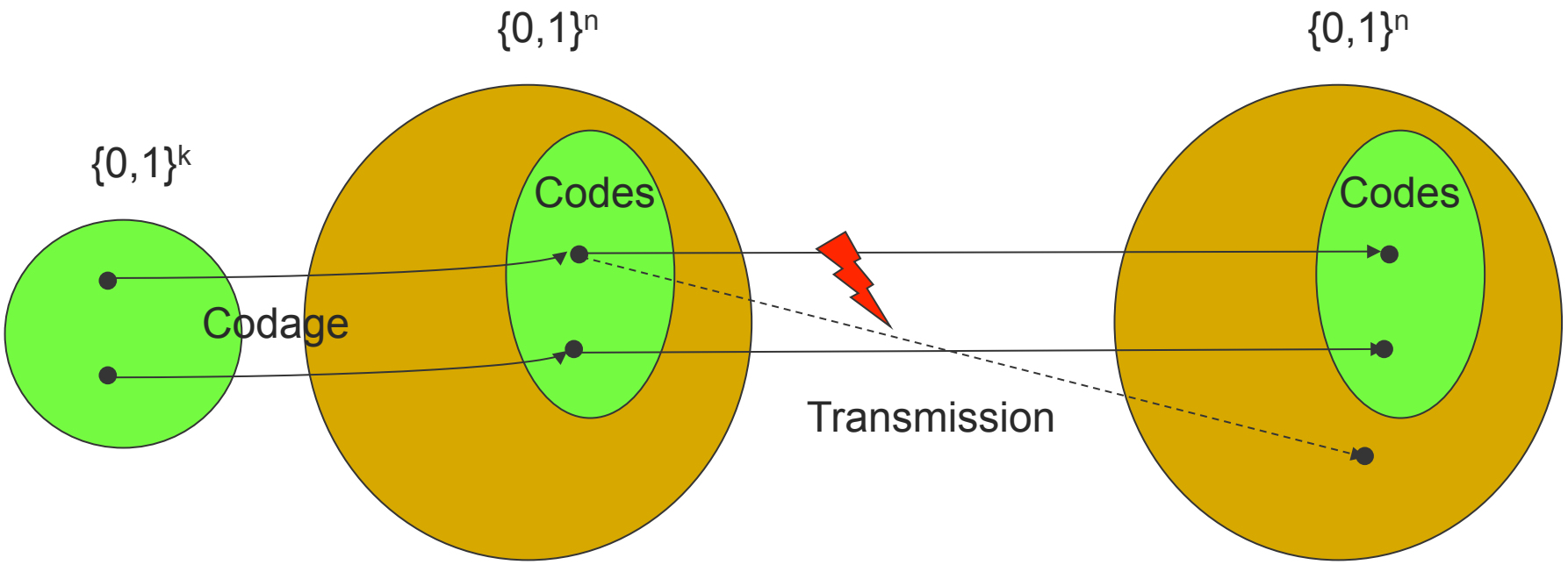
Soit $F^{\perp}$ le s.e.v orthogonal à F càd $F^{\perp} = \{v \in E / \langle v,u \rangle = 0, \forall u \in F\}$

Alors $\boxed{\dim F + \dim F^{\perp} = \dim E}$ càd $\dim F^{\perp} = n-k$

Notations

- soit m un mot codé envoyé. ex : le vecteur (0 0 1 1 0)
- soit m' le mot reçu. ex : le vecteur (0 1 0 1 0)
- m' = m + e où e est le vecteur d'erreur, ici e = (0 1 1 0 0) càd qu'ici deux bits sont faux

Principe d'un code correcteur



Codes en blocs linéaires

- Définition
  - Un code C est dit linéaire si c'est un sous-espace vectoriel de $[GF(2)]^n$
  - Fonction de codage : application linéaire de $[GF(2)]^k$ dans $[GF(2)]^n$
  - C est de dimension k, C contient 2^k vecteurs (codes) de n bits.
  - n est la longueur, k la dimension (k est la taille des blocs à coder)
  - C est noté C(n,k).
- Conséquence : le code de (00…00) est (000….00) puisque C linéaire

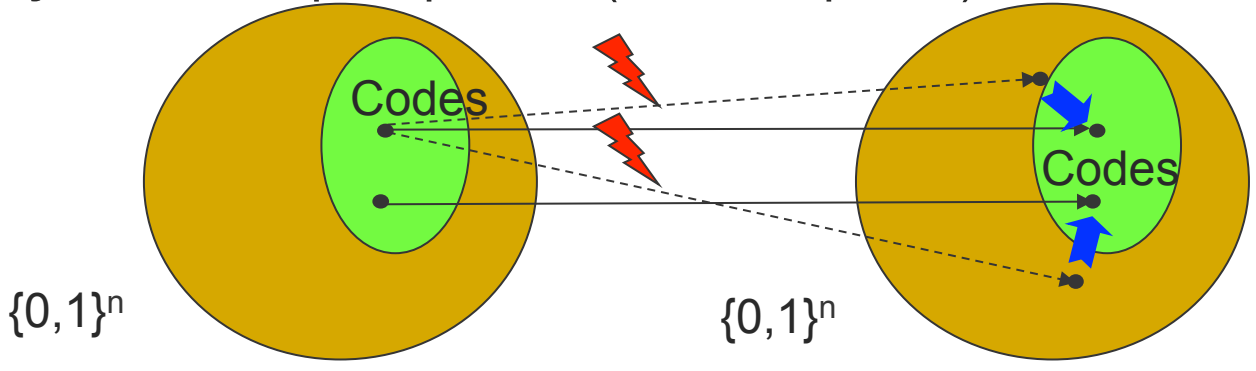
- Définition : Soit $v \in V$, on appelle poids de v noté w(v) le nombre de composantes non nulles.

- Définition : d = distance de Hamming = Nbre de bits distincts
  - d(u,v) = w(u-v) = w(u+v)

Codes en blocs linéaires

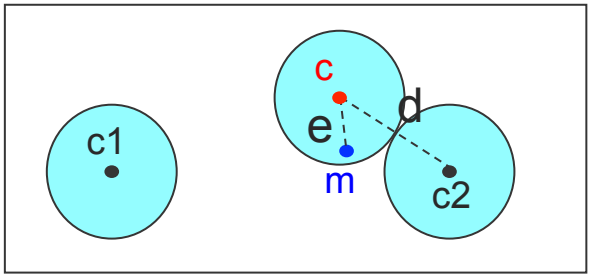
- Ex : m_1=110101, m_2=111001 => $d(m_1,m_2)=2$
- Définition : distance d'un code linéaire :
  - d = Min d(u,v) = Min w(u-v) => d(u,v) ≥ d, ∀u, ∀v
- Propriété : Min w(u-v) = Min w(u') (puisque u-v=u' ∈ C car C est un espace vectoriel)
- La distance d'un code linéaire est égale au minimum du poids des codes non nuls

  d =Min w(u)
- Définition : La correction à distance minimale fait correspondre à chaque mot reçu le mot le plus proche (cf. intro. proba)

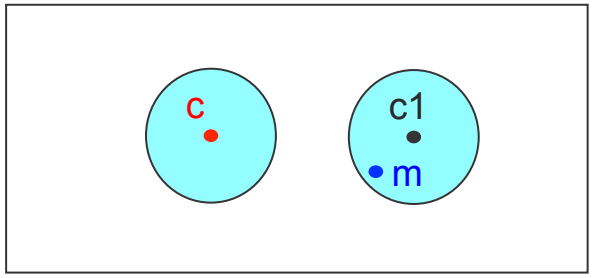


Capacité de correction

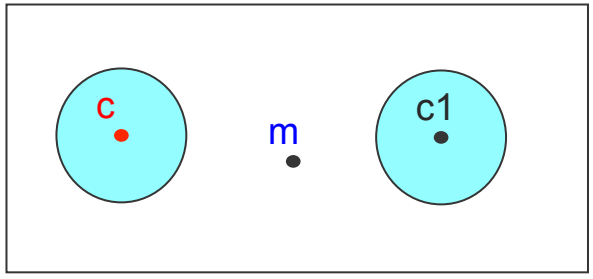
Soit c1,c2, ….les mots du code. Soit c le code émis, m le mot reçu.
Soit une "boule" de centre un code, de rayon d/2.

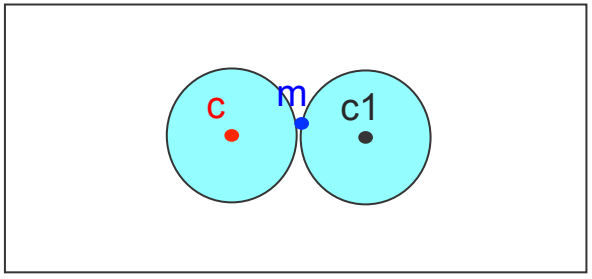


Si e=d(c,m) < d/2, correction exacte



Si d(c,m) > d/2, correction erronée





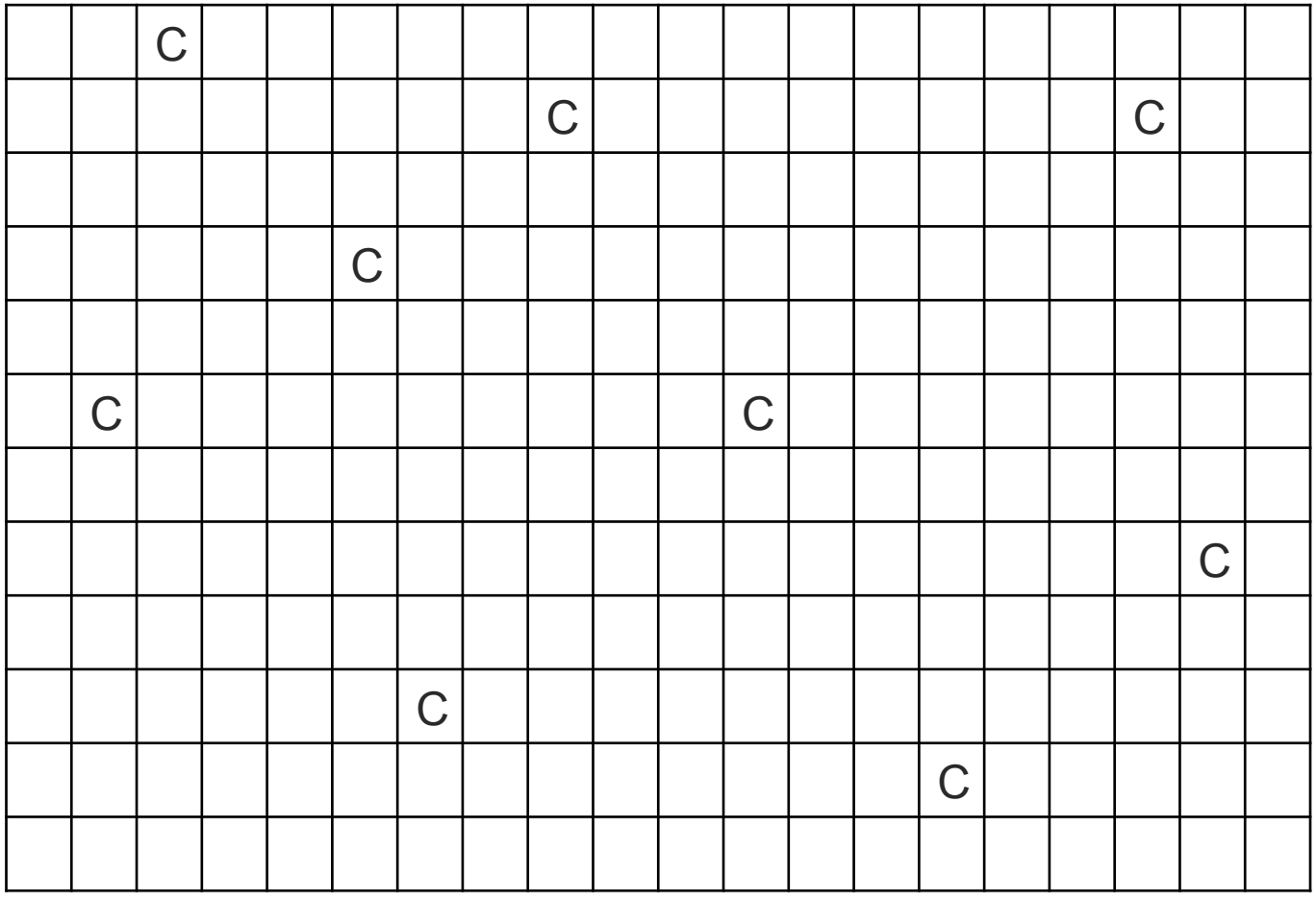
Si d(c,m) = d/2, correction peut être erronée

Capacité de détection et correction

il est possible de détecter d-1 erreurs par mot

il est possible de corriger t=⌊(d-1)/2⌋ erreurs par mot

Capacité de détection/correction



d = 5

Codes en blocs linéaires

- Lemme : Un code linéaire de distance minimum d peut détecter jusqu'à d-1 erreurs et en corriger jusqu'à t avec d=2t+1 ou d=2t+2, si on utilise la règle de décodage à distance minimum

- $t=\lfloor (d-1)/2 \rfloor$

- Notations
  - Un code de dimension k de longueur n de distance minimale d est noté C(n,k,d)
  - Un code qui corrige jusqu'à t erreurs est appelé t-correcteur

Codes en blocs linéaires

- Exemple : n=6,k=3 (mots de 3 bits codés en mots de 6 bits)

a	v	w(v)
000	000000	0
001	110110	4
010	011101	4
011	101011	4
100	100101	3
101	010011	3
110	111000	3
111	001110	3

- Poids minimum W=3 => d=3, on peut corriger jusqu'à $\lfloor (d-1)/2 \rfloor = 1$ erreur

Codes linéaires : matrice génératrice G

- C(n,k) sous-espace vectoriel de $[GF(2)]^n$
- Soit {g1,g2,…..,gk} un base de C
- On peut écrire
  - $C = \{u \in V / u = Ga, a=(a1,a2,….,ak)^t \in [GF(2)]^k \}$ où G est la nxk matrice dont les colonnes sont les gi de la base de C
  - G est appelée la **matrice génératrice** de C (rang de G = k). C'est la matrice de l'application de codage.

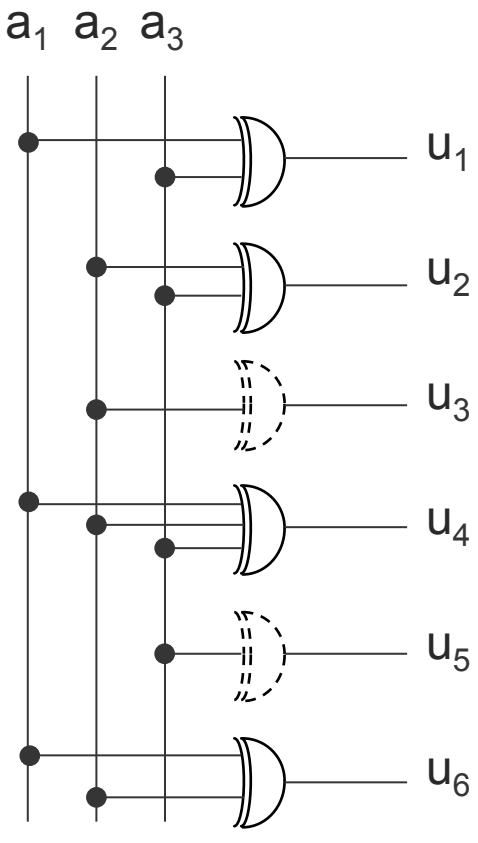
- On veut construire un code linéaire C(6,3)
  - choisir 3 vecteurs indépendants de $[GF(2)]^6$ (base de C)
  - par exemple

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \qquad G \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

a	u = Ga
000	000000
001	110110
010	011101
011	101011
100	100101
101	010011
110	111000
111	001110

Structure de codeur

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$



Codes linéaires : matrice génératrice G

- remarque : Pour un code C fixé, toute application linéaire faisant correspondre à la base canonique de $[GF(2)]^k$ une base quelconque du code définit le même code.
  - l'ensemble des vecteurs de code est le même
  - seule la correspondance entre mots d'information et codes est différente

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

e1 e2 e3

a	u = Ga	
000	000000	
001	110110	*
010	011101	*
011	101011	
100	100101	
101	010011	
110	111000	
111	001110	*

$$G' = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

e2 | e2 + e3 | e1 + e2 + e3

a	u = G'a
000	000000
001	001110
010	101011
011	100101
100	011101
101	010011
110	110110
111	111000

Codes linéaires : matrice de contrôle H

- Principe de la détection d'erreurs : détecter que le mot reçu appartient bien au code càd qu'il appartient bien au sous espace vectoriel ⇔ pas de composantes dans le s.e.v orthogonal.

- Soit $C^{\perp}$ le sous-espace vectoriel orthogonal à C
  - $C^{\perp} = \{v \in V / \langle u,v \rangle = 0$, pour tout $u \in C\}$
  - $C^{\perp}$ est de dimension n-k, il peut être considéré comme un code linéaire $C^{\perp}$ (n,n-k) qui a une nx(n-k) matrice génératrice notée H.

- On a $(C^{\perp})^{\perp} = C$ donc si H est la matrice génératrice de $C^{\perp}$, on a :
  $v \in C \Leftrightarrow H^t.v = 0$ => Détection d'erreurs

- Rem : rôles duaux de C et $C^{\perp}$, en particulier pour $v \in C^{\perp}$, on a $G^t.v = 0$

- H^t est appelée matrice de contrôle (souvent notée H, avec H^t la matrice génératrice de $C^{\perp}$)

- Un code linéaire admet comme matrice de contrôle la transposée de la matrice génératrice de son code orthogonal

- De même C peut être défini comme le noyau d'un application linéaire de $[GF(2)]^n$ dans $[GF(2)]^{n-k}$ de matrice H

Construction de H

Sur l'exemple précédent G = $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

- Construction de H : on cherche une base de $C^{\perp}$, ici dim $C^{\perp}$ = 3. On cherche donc 3 vecteurs linéairement indépendants de $C^{\perp}$:
  - Soit V = $(v1\ v2\ v3\ v4\ v5\ v6)^t \in C^{\perp}$, on a : $G^t\ (v1\ v2\ v3\ v4\ v5\ v6)^t = (0\ 0\ 0)^t$ c-à-d une base du noyau de H
- On doit avoir (en utilisant G) :

  v1 + v4 + v6 = 0
  v2 + v3 + v4 + v6 = 0
  v1 + v2 + v4 + v5 = 0

  - En choisissant par exemple v3, v4 et v6, on obtient

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{matrix} \\ \\ v3 \\ v4 \\ \\ v6 \end{matrix}$$

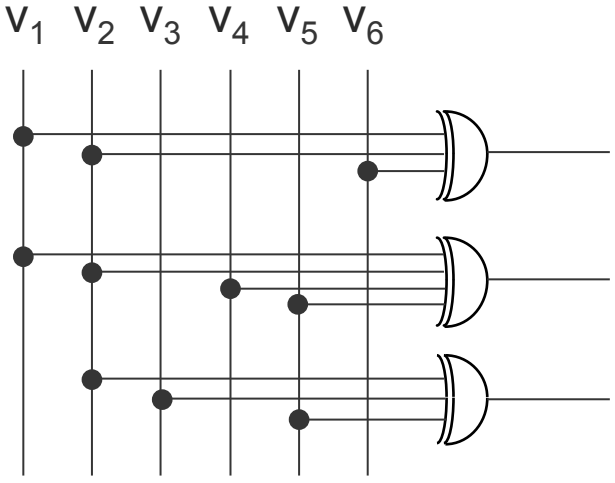
- remarques :
  - Un autre choix de variables donnerait une autre base de solutions et donc une autre matrice de contrôle
  - On peut construire H en remarquant que H.G = matrice nulle

Structure de contrôleur

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$H^t = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Matrice de contrôle



Exercice

- Verifier que Hv = 0 pout tout v appartenant au code
- Pour v n'appartenant pas au code, verifier que Hv est différent de 0 (vecteur nul).

Code $C^{\perp}$

- Le code orthogonal $C^{\perp}$ est donc un code de longueur 6 et de dimension 6-3=3

- H est utilisée pour la détection et la correction

Dans la suite H désigne la matrice de contrôle, càd la transposée de la matrice génératrice du code orthogonal.

a	u = Ha	w(v)
000	000000	0
001	011010	4
010	110110	4
011	101100	4
100	110001	3
101	101011	3
110	000111	3
111	011101	3

- Proposition 1 : Soit C un code linéaire de matrice de contrôle H. Il existe un mot de code de poids w si et seulement si il existe w colonnes de H linéairement dépendantes
  - Dem : un mot de code v vérifie l'équation H.v = 0. Si on note h_i la $i^{ème}$ colonne de H, on peut écrire l'équation précédente $\sum_{i=1}^{n} v_i.h_i = 0$. Donc si v de poids w, on a w coefs ≠0.

- Corollaire 1: Un code linéaire C de matrice de contrôle H a un poids minimum w si et seulement si tout ensemble de w-1 colonnes de H est une famille libre.

- Remarque la distance minimum d d'un code linéaire C(n,k) doit satisfaire :
  $d \leq n-k+1$

Code étendu

- Code étendu : Soit C(n,k,d). On considère le code C'(n+1,k) de distance minimum d ou d+1 où chaque mot du code $v'=(v_1,v_2,\ldots,v_{n+1})$ est tel que $v=(v_1,v_2,\ldots.v_n)$ et $v_{n+1}= f(v)$ pour une certaine fonction f.

- Le cas le plus classique est $v_{n+1}= f(v) = \sum v_i$: parité, tous les codes de C' ont un poids pair. La matrice de contrôle H' est obtenue en ajoutant à H une ligne de 1 et une colonne de 0

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$H' = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Code tronqué

- Code tronqué : Soit C(n,k,d). On considère le code C'(n,k) obtenu en supprimant une ou plusieurs coordonnées de chaque mot de C(n,k,d). Chaque fois qu'on supprime une coordonnée, la longueur diminue d'une unité mais la dimension reste constante. La distance minimum peut diminuer mais aussi peut ne pas diminuer (dans ce cas même qualité de détection/correction mais + court).

$$G=\begin{pmatrix}1&0&1\\0&1&1\\0&1&0\\1&1&1\\0&0&1\\1&1&0\end{pmatrix}$$

en supprimant la deuxième coordonnée

$$G'=\begin{pmatrix}1&0&1\\0&1&0\\1&1&1\\0&0&1\\1&1&0\end{pmatrix}$$

a	v = G'a	w(v)
000	00000	0
001	10110	3
010	01101	3
011	11011	4
100	10101	3
101	00011	2
110	11000	2
111	01110	3

Forme systématique

- Idée : Passage G à H : résolution de n-k équations à n inconnues => passage facile forme systématique

- Définition : C(n,k) est dit sous forme systématique si le mot d'information se trouve dans les k premières coordonnées. A une permutation près cela signifie que le mot de code est composé du mot d'information et d'un suffixe. On parle aussi de code séparable.

$$i_1 i_2 \ldots . i_k \longrightarrow i_1 i_2 \ldots . i_k c_1 c_2 \ldots . c_l$$

$\mathbf{G} = \begin{pmatrix} \mathbf{I_k} \\ \hline \mathbf{P} \end{pmatrix}$ où I_k est la matrice identité de dimension k, et P une matrice (n-k)xk

- Exemple : code de parité $G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$

- Définition : Deux codes C(n,k) et C'(n,k) sont dits équivalents ssi il existe une permutation S qui transforme tout mot de code de C en un mot de code de C', c-à-d :
  G'= G.S et H'= H.S

Forme systématique

- Proposition : Tout code linéaire C(n,k) est équivalent à un code linéaire sous forme systématique C'(n,k) avec

$$G' = \begin{pmatrix} I_k \\ = \\ P \end{pmatrix} \text{ et } H' = \begin{pmatrix} -P^t \\ = \\ I_{n-k} \end{pmatrix} = \begin{pmatrix} P^t \\ = \\ I_{n-k} \end{pmatrix} \text{ càd matrice de contrôle} = \left(P \middle| I_{n-k}\right)$$

Dem : par combinaisons linéaires des colonnes de G on peut trouver une matrice identité d'ordre k (puisque le rang est k), ce qui ne change pas le sous-espace vectoriel. Par une permutation des lignes on peut placer la matrice identité dans les k premières colonnes (ce qui change le sev) => G'. Les lignes de H' sont orthogonales aux lignes de G'. Comme elles sont linéairement indépendantes alors H' peut être considéré comme la matrice de contrôle de C'.

Exemple 1

$$G=\begin{pmatrix}1&0&1\\0&1&1\\0&1&0\\1&1&1\\0&0&1\\1&1&0\end{pmatrix}$$

1/ On remplace la troisième colonne par la somme de la 1ere et de la 3eme

$$G'=\begin{pmatrix}1&0&0\\0&1&1\\0&1&0\\1&1&0\\0&0&1\\1&1&1\end{pmatrix}$$

2/ On fait la permutation de lignes (1 3 5 4 2 6)

$$G'=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\\0&1&1\\1&1&0\\1&1&1\end{pmatrix}\begin{matrix}I_k\\\\\\P\\\\\end{matrix}\quad\Leftrightarrow\quad H'=\begin{pmatrix}0&1&1\\1&1&1\\1&0&1\\1&0&0\\0&1&0\\0&0&1\end{pmatrix}\begin{matrix}-P^t\\\\\\\\\\I_{n-k}\end{matrix}$$

Exemple 2 : code par parité pour des mots de 4 bits

$i_1, i_2, i_3, i_4 \longrightarrow i_1, i_2, i_3, i_4, i_1+i_2+i_3+i_4$

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Codes en blocs linéaires : en résumé

- Codage de l'information c = Ga
- Décodage à la réception : Hv

  $c \rightarrow v = c+e \rightarrow s = Hv = H(c+e) = H(c) + H(e) = 0 + H(e) = H(e)$

- s = H(e) est le syndrome du mot reçu dim (n-k)
- Erreurs détectées si s≠0 (il faut que e ne soit pas un mot du code)
- w(e)= nombre d'erreurs.

Correction Décodage des codes linéaires

- But : donner un règle de décodage à distance minimum

- Principe
  - soit m le mot reçu ($H(m) \neq 0$)
  - on ajoute à m le vecteur ec (vecteur de correction) tel que
    - 1/ m+ec soit un mot du code
    - 2/ ec soit de poids minimum (décodage à distance minimum)
  - remarque ec = e (vecteur d'erreur)

Correction Décodage des codes linéaires

- **Définitions**
  - Deux vecteurs sont dits équivalents si H(u) = H(v) (même syndrome)
  - Relation d'équivalence => classes d'équivalence
  - Nombre d'éléments par classe : 2^k = Cte
    - les mots du code, de syndrome nul, forment une classe qui possède 2^k éléments.
    - toutes les classes ont même cardinal :
      - H(u) = H(v) ⇔ H(u)+H(v) =0 ⇔ H(u+v) =0 ce qui implique que (u+v) appartient au code.
      - Les vecteurs de même syndrome que u s'obtiennent en ajoutant un vecteur de code à u. Leur ensemble est la classe de u notée : u+C= {v∈[GF(2)]^n /v=u +v', v' ∈C} appelé translaté de u. Donc le cardinal de la classe est celui du code càd 2^k

- En résumé :
  - Il y a donc 2^{n-k} classes de 2^k éléments
  - Une classe est obtenue en ajoutant tous les vecteurs de code à un représentant de la classe

Correction Décodage des codes linéaires

- Correction :

  1/ on calcule le syndrome du mot reçu m,

  2/ puisque le syndrome de la somme de 2 vecteurs de même syndrome est nulle (la somme est donc un mot du code), on ajoute un mot d'erreur ec qui a ce syndrome pour obtenir un "bon" code

  3/ Le mot d'erreur à choisir ec est un mot :

  - de même syndrome que m
  - qui doit être de poids le + faible (pour obtenir le mot de code le + proche)

Tableau standard

- On écrit tous les vecteurs de $[GF(2)]^n$ comme éléments d'un tableau à 2^k colonnes et 2^{n-k} lignes(càd les syndromes).
- Chaque ligne est un translaté de C.

Code		0	v_1	v_2	…	v_j	…	v_N
u_1+C	:	u_1	u_1+v_1	u_1+v_2	…	u_1+v_j	…	u_1+v_N
…	:	…	…	…	…	…	…	…
u_i+C	:	u_i	u_i+v_1	u_i+v_2	…	u_i+v_j	…	u_i+v_N
…		…	…	…	…	…	…	…
u_M+C	:	u_M	u_M+v_1	u_M+v_2	…	u_M+v_j	…	u_M+v_N

avec $N = 2^k-1$ et $M = 2^{n-k}-1$

Le tableau standard de décodage est un tableau dont chaque ligne est un translaté de C (càd tous les éléments de la ligne ont même syndrome)

Algorithme de correction

- A l'arrivée du message m, son syndrome est calculé par H(m) => classe de m
- Dans cette classe, un vecteur de poids minimum est pris comme vecteur de correction ec (décodage à distance minimum)
- Le message est corrigé en : (m+ec)

- Remarque :
  - si H(m)=0, m est un mot du code et on accepte le message reçu comme exact

Décodage par syndrome : tableau standard

- soit v' le vecteur reçu, calculer son syndrome H(v').
- déterminer le représentant ui du syndrome
- décoder v' par v'-u_i
- Exemple précédent

$$G=\begin{pmatrix}1&0&1\\0&1&1\\0&1&0\\1&1&1\\0&0&1\\1&1&0\end{pmatrix} \qquad H=\begin{pmatrix}1&1&0&0&0&1\\1&1&0&1&1&0\\0&1&1&0&1&0\end{pmatrix}$$

	Vecteurs								Syndrome
C	000000	110110	011101	101011	100101	010011	111000	001110	000
u_1+C	100000	010110	111101	001011	000101	110011	011000	101110	110
u_2+C	010000	100110	001101	111011	110101	000011	101000	011110	111
u_3+C	001000	111110	010101	100011	101101	011011	110000	000110	001
u_4+C	000100	110010	011001	101111	100001	010111	111100	001010	010
u_5+C	000010	110100	011111	101001	100111	010001	111010	001100	011
u_6+C	000001	110111	011100	101010	100100	010010	111001	001111	100
u_7+C	001001	111111	010100	100010	101000	011010	110001	000111	101

Décodage par syndrome

- mot de code envoyé 100101, et soit une erreur sur le cinquième bit -> mot reçu 100111
- 1/ Calcul du syndrome

$$H(100111)^t = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

- Puisque le syndrome n'est pas le vecteur $(000)^t$, il y a détection d'erreur
- 2/ déterminer le représentant de poids minimum qui a le même syndrome $(011)^t$: $(000010)^t$ (pb de place et de temps)
- 3/ décoder $(100111)^t$ par :

  le mot de code $(100111)^t – (000010)^t = (100101)^t$

Tableau standard

- Lemme : Si le code linéaire est t-correcteur alors tous les vecteurs de poids inférieur ou égal à t sont dans des translatés différents. Dans ce cas, on a l'inégalité :

$$C_n^0 + C_n^1 + \ldots\ldots + C_n^t \leq 2^{n-k}$$

C_n^i = nombre de choix de i éléments parmi n

  - Dem: Supposons que v_1 et v_2 soient de poids inférieurs à t et qu'ils soient dans le même translaté.

    On a donc v_1-$v_2 \in C$ et donc $w(v_1-v_2) \geq d \geq 2t+1$

    (puisque $t \leq (d-1)/2$).

    Mais on a $w(v_1-v_2)=w(v_1+v_2) \leq w(v_1)+w(v_2) \leq 2t$. Contradiction.

- Ce lemme permet de choisir comme représentants tous les vecteurs de poids inférieurs ou égal à t

Tableau standard réduit

- Tableau de décodage mal commode pour n grand (pour n=32 et k=6 il faudrait 16 Gbytes de mémoire => utilisation de la matrice de contrôle.
- Principe : on ne prend pour chaque syndrome que le représentant de poids minimal (qui est le vecteur de correction)
- Algo de remplissage :
  - pour chaque vecteur de poids 1, on calcule le syndrome
    - si le syndrome n'a pas déjà été obtenu, on prend le vecteur comme représentant de la classe
  - on réitère avec les vecteurs de poids 2, puis de poids 3 etc…

Exemple

- Soit le code C(4,2) de matrice génératrice $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$
- La matrice de contrôle est $\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$
- La fonction de codage fait correspondre à chaque vecteur de 2 bits un vecteur de 4 bits.
- Il y a 2^4 messages (reçus) possibles qui se répartissent en 2^{n-k}= 4 classes suivent les syndromes :
  - S0= (0,0), S1=(0,1), S2=(1,0) , S3=(1,1)

Tableau standard réduit : exemple

- Classe 0 : (0 0 0 0) (1 0 1 0) (0 1 1 1) (1 1 0 1)

  Base

- Classe 1 :
  - soit m=(0 0 0 1), m n'appartient pas au code
    - $H(m) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- Classe 2
  - soit m= (0 0 1 0), $H(m) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- Classe 3
  - soit m = (0 1 0 0), $H(m) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Tableau standard réduit : exemple

Syndromes	S0 = (0 0)	S1 = (0 1)	S2 = (1 0)	S3 = (1 1)
ec	(0 0 0 0)	(0 0 0 1)	(0 0 1 0)	(0 1 0 0)
classes	0	1	2	3

Soit m = (1 0 1 0) le mot envoyé
Soit m' = (1 0 1 1) le mot reçu

$$Hm' = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

=> ec = vecteur d'erreur = (0 0 0 1) => m corrigé = m' + ec = (1 0 1 0)

2ème Partie : Quelques codes

- Capacité de correction t croissante avec d

  puisque $t=\lfloor (d-1)/2 \rfloor$

- Pour augmenter d, il faut augmenter n
- mais si redondance n-k augmente, le rendement k/n diminue

Efficacité de la correction

- Capacité de correction : $t=\lfloor (d-1)/2 \rfloor$
- Evaluation de la distance minimale
  - pas facile si code grand
- Majoration de d :
  - Dans un code C(n,k), la distance minimale est inférieure ou égale à n-k+1.

$$d \leq n-k+1$$

  - Par exemple : si on code des mots de 5 bits en mots de 8 bits. On a $d \leq n-k+1=8-5+1= 4$. Donc $t \leq 1$. On ne pourra corriger au mieux qu'1 bit, quelque soit le code choisi.

Majoration de d : démonstration

Soit k'=n-k+1 et soit $[GF(2)]^{k'}$ un s.e.v. de $[GF(2)]^{n}$.

Chaque vecteur v de ce s.e.v. peut être considéré comme un vecteur dont les n-k'=k-1 premières composantes sont nulles : v=(0…0v1v2….vk')

C et $[GF(2)]^{k'}$ sont 2 s.e.v., ils ont donc le vecteur nul en commun

S'ils n'avaient que le vecteur nul en commun, leurs dimensions vérifieraient $k'+k \leq n$.

En effet, soit (ej) une base de C et (e'k) une base de $[GF(2)]^{k'}$ et soit une combinaison linéaire nulle des ej et e'k :

$$\sum \lambda_j e_j + \sum \gamma_k e'_k = 0 \ càd \ \sum \lambda_j e_j = \sum \gamma_k e'_k$$

Si $C \cap [GF(2)]^{k'} = 0$, alors $\sum \lambda_j e_j = \sum \gamma_k e'_k = 0$

D'où, $\forall j : \lambda_j = 0$ et $\forall k, \gamma_k = 0$, donc (ej),(e'k) est une famille libre, d'où $k'+k \leq n$

Or k+k' = k + (n-k+1) = n+1 > n.

Donc il existe au moins un vecteur non nul de C qui appartient également à $[GF(2)]^{k'}$.

Son poids est au plus égal à k' (puisque ses k-1 premières composantes sont nulles).

Or d est inférieure ou égale au poids d'un vecteur.

Donc la distance minimale de C est inférieure ou égale à k'=n-k+1

Minoration de d

- Théorème : d ≥ (w+1) si et seulement si tout sous-ensemble de w vecteurs colonnes de la matrice de contrôle H est libre
- Remarques
  - si c est un mot de code non nul de poids w dont les composantes non nulles occupent les positions $j_1, j_2, \ldots j_w$:

    $S(c) = H(c) = 0 = H_{j1} + \ldots + H_{jw.}$ Donc il existe w colonnes non nulles de H linéairement liées.
  - si il existe w colonnes $H_{j1}, \ldots, H_{jw}$ non nulles de H linéairement liées par la relation $0 = H_{j1} + \ldots + H_{jw,}$ il existe un message de poids w dont le syndrome est nul, il s'agit donc d'un mot de code de poids w

Minoration de d

- Dem :
  - a/ supposons que toutes les ensembles de w colonnes soient libres

    Il n'existe donc aucune relation de la forme $H_{j1} + \ldots + H_{jw}=0$, donc pas de mot émis de poids w. Or si un ensemble est libre, tous les sous-ensembles le sont également, il n'existe donc pas de mot de code w-1, w-2, etc.., càd pas de mot de code de poids inférieur à w. Donc $d \geq w+1$
  - b/ réciproque , soit $d \geq w+1$

    aucun mot non nul n'est de poids w, il n'existe donc pas de relation du type $H_{j1} + \ldots + H_{jw}=0$, tous les sous-ensembles de w colonnes de H sont donc libres.

Les relations en résumé

- d =Min w(u) (w(u) = poids du vecteur u où u $\in$ C)

- Détection : jusqu'à d-1 erreurs
- Correction : jusqu'à t=$\lfloor (d-1)/2 \rfloor$ erreurs

- $d \leq n-k+1$ = redondance +1
- $d \geq (w+1) \Leftrightarrow$ tout sous-ensemble de w vecteurs colonnes de H est libre

Codes parfaits

- p = proba. qu'un bit soit mal transmis
- t = capacité de correction =>
  - Messages avec t erreurs ou moins sont bien corrigés
  - Certains messages avec plus de t erreurs sont corrigés
- Proba d'exactitude après décodage

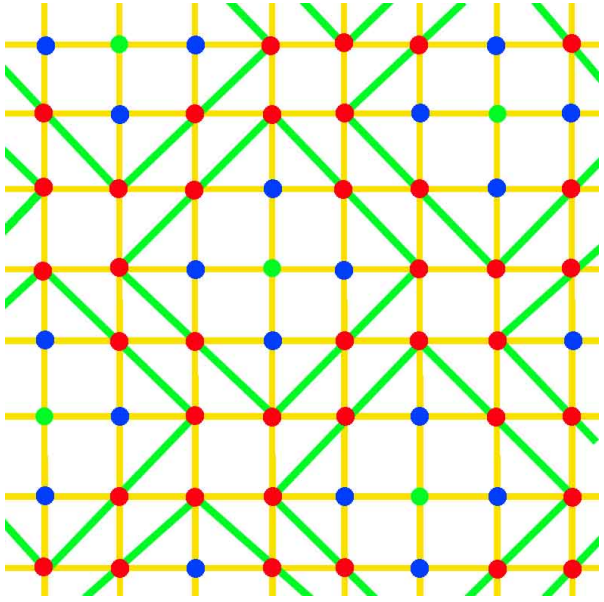
$$p_{exa} \geq \Pr(X=0) + \Pr(X=1) + \ldots\ldots + \Pr(X=t)$$

$$\geq \sum_{k=0}^{t} C_n^k p^k (1-p)^{n-k}$$

- Code parfait : Un code de capacité de correction t est dit parfait, si tout message ayant plus de t erreurs est à coup sur incorrectement corrigé
- Pour un code parfait : $p_{exa} = \sum_{k=0}^{t} C_n^k p^k (1-p)^{n-k}$
- Exemple : le code C5,1 est parfait.
  - En effet, 2 mots (0,0,0,0,0) et (1,1,1,1,1) => t=2.
  - Si l'erreur est de poids inférieur ou égale à 2, message bien corrigé à coup sur
  - Si l'erreur est de poids supérieure ou égale à 2 , incorrectement corrigé à coup sur

Exemple de code parfait de distance 5

Tous les mots de code sont à distance 5 l'un de l'autre => t =2



Mot de code

Mot à distance 1

Mot à distance 2

Code parfait : les boules de rayon t couvrent l'espace

CODES de HAMMING (canaux peu bruités)

- Code de Hamming (1948):
  - code linéaire
    - corrigeant toutes les erreurs de poids 1 (de façon fiable)
    - de rendement $\rho = k/n$ maximum pour une redondance fixée m = n-k
- Théorème
  - un code c(n,k) est au moins 1-correcteur si et seulement si toutes les colonnes de sa matrice de contrôle sont distinctes et non nulles
  - dem : colonnes distinctes ⇔ tout sous-ensemble de 2 colonnes est libre ⇔

    1/ $d \geq 3$ d'après le théorème sur la minoration ($d \geq w+1$)

    2/ $t \geq 1$ puisque $t=\lfloor (d-1)/2 \rfloor$
- Rendement

  1/ Les n colonnes de la matrice de contrôle H sont des vecteurs de $[GF(2)]^m$ avec m= n-k, il y a 2^m vecteurs dans $[GF(2)]^m$

  2/ Si le code est de capacité de correction $t \geq 1$, les n colonnes de H sont non nulles et toutes distinctes. On a donc : $n \leq 2^m - 1$

  3/ rendement $\rho = k/n = (n-m)/n = 1-m/n$

  ρ est maximum si m/n minimum, c-à-d si n est maximum ce qui implique $n = 2^m - 1$ (et donc $k = n-m = 2^m - m - 1$)

- H a donc pour colonnes tous les vecteurs de $[GF(2)]^m$ sauf le vecteur nul

Codes de Hamming

- Ce sont des codes $(n,k,d)=(2^m-1, 2^m-m-1, 3)$ avec m = la redondance
  - En pratique, on fixe la redondance m, ce qui donne k et n
- On code donc chaque bloc de $k=2^m-m-1$bits par un bloc de $n=2^m-1$ bits
- rem : la valeur de m est au moins 2 (si m=1, alors k=0, non sens)

m $n = 2^m-1$ k = n-m	2 3 1	3 7 4	4 15 11	5 31 26	6 63 57	7 127 120
codes	$\mathcal{H}(3,1)$	$\mathcal{H}(7,4)$	$\mathcal{H}(15,11)$	$\mathcal{H}(31,26)$	$\mathcal{H}(63,57)$	$\mathcal{H}(127,120)$
rendement	0,33	0,57	0,73	0,83	0,90	0,94

- $\mathcal{H}(7,4)$: 7 bits de code, pour 4 bits à coder.
  il peut y avoir 7 erreurs de 1 bit => 3 bits pour les identifier => 4+3 = 7

Codes de Hamming

- Propriétés

  1/ distance minimale 3

  2/ capacité de correction 1

  3/ parfait càd toute erreur à distance 1 (càd une erreur sur un seul bit) d'un code est corrigée

- dem
  - 1/ $d \geq 3$ puisque $t \geq 1$. Tous les vecteurs de $[GF(2)]^m$ formant les colonnes de H, la somme de deux vecteurs colonnes est donc une colonne de H. Il existe donc 3 colonnes linéairement liés càd telles que $Hi+Hj+Hk=0$. Il existe donc des mots de code de poids 3, donc $d=3$.
  - 2/ évident

Codes de Hamming

- dem 3/
  - soit B(c,d/2) les boules ayant pour centre un mot c du code et un rayon égal à d/2.
  - Chaque boule contient tous les messages à distance 1 de son centre. Il y en a n.
  - Les 2^k boules en contient donc $n*2^k = (2^m-1)*2^k$.
  - Le nombre de messages erronés est égal à

    $(2^n-2^k) = 2^k (2^{n-k}-1) = (2^m-1)* 2^k$
  - si m a une erreur de poids 1, il se trouve dans la boule b(c,3/2), à la distance 1; il est à coup sur corrigé exactement
  - si m a une erreur de poids supérieur à 1, il est à coup sur inexactement corrigé.

Codes de Hamming : exemple

- Pour m=3, soit $\mathcal{H}(7,4)$ de matrice de contrôle : $\begin{pmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{pmatrix}$

- Correction : les colonnes de H représentent tous les syndromes des messages erronés et donc les représentants des classes sont les vecteurs $(1000000)^t$, $(0100000)^t$, …, $(0000001)^t$
- Soit m = $(1\ 0\ 1\ 0\ 1\ 0\ 1)^t$. Son syndrome est $(1\ 0\ 0)^t$. Il est donc erroné. Or $(1\ 0\ 0)^t$ est à la colonne 4 qui est le syndrome de $(0\ 0\ 0\ 1\ 0\ 0\ 0)^t$.
- m est donc remplacé par :

  $(1\ 0\ 1\ 0\ 1\ 0\ 1)^t + (0\ 0\ 0\ 1\ 0\ 0\ 0)^t = (1\ 0\ 1\ 1\ 1\ 0\ 1)^t$
- Donc
  - si m a effectivement un erreur de poids 1, celle-ci est située sur la 4ème colonne et le message est corrigé
  - si m a une erreur de poids supérieur à 1, le vecteur $(1\ 0\ 1\ 1\ 1\ 0\ 1)^t$ n'est pas le mot émis

Codes de Hamming : en pratique

- Obtenus par la matrice de contrôle
- H contient tous les vecteurs sauf le vecteurs nul
- Par exemple la matrice de contrôle H de $\mathcal{H}(15,11)$ (m=4) est:

$$\begin{pmatrix} 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{pmatrix}$$

- Par permutation des colonnes, on obtient toute une famille de codes de Hamming

Codes de Hamming : en pratique (1)

Rappel : les codes de Hamming permettent de coder des mots de longueur 1= (2^2-1-2) ou 4 (2^3-1-3) ou11 = (2^4-1-4) ou 26 = (2^5-1-5) ou 57 = (2^6-1-6) ou120 = (2^7-1-7),…

Comment obtenir un code de Hamming pour les autres longueurs ?

Soit par exemple des mots de longueur 3 : $V := \begin{bmatrix} v1 \\ v2 \\ v3 \end{bmatrix}$

On agrandit le mot jusqu'à la dimension k' (4,11,26..) immédiatement supérieure.

Par exemple, pour k = 3, on considère que c'est un mot de longueur 4 ,dont le premier bit v_0 est fixé à une valeur quelconque, par exemple 0 (en d'autres termes $\{0,1\}^3$ est considéré comme un s.e.v de $\{0,1\}^4$).

$V' := \begin{bmatrix} v0 \\ v1 \\ v2 \\ v3 \end{bmatrix}$

On effectue le codage sur la nouvelle forme, puis on fixe dans l'expression des bits de code la valeur de v_0 choisie.

Codes de Hamming : en pratique (2)

Par exemple, on fixe la valeur de v_0 à 0 et on utilise un code de Hamming séparable

La valeur de ce bit v_0 restera inchangée.

Soit alors les matrices de H(7,4) séparable, par exemple :

$$H := \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad G := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

$$G.V' = C' = \begin{bmatrix} v0 \\ v1 \\ v2 \\ v3 \\ v1 + v2 + v3 \\ v0 + v2 + v3 \\ v0 + v1 + v3 \end{bmatrix} \Rightarrow_{v0 = 0} C = \begin{bmatrix} v1 \\ v2 \\ v3 \\ v1 + v2 + v3 \\ v2 + v3 \\ v1 + v3 \end{bmatrix}$$

Codes de Hamming : en pratique (3)

En codage : la première colonne de G ajoute les termes en v_0.

$$V := \begin{bmatrix} 0 \\ v1 \\ v2 \\ v3 \end{bmatrix}$$

$$G := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 \\ v1 \\ v2 \\ v3 \\ v1 + v2 + v3 \\ v2 + v3 \\ v1 + v3 \end{bmatrix}$$

=> En prenant v_0 =0, on peut supprimer la première colonne et la première ligne de G

$$V := \begin{bmatrix} v1 \\ v2 \\ v3 \end{bmatrix}$$

$$G := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} v1 \\ v2 \\ v3 \\ v1 + v2 + v3 \\ v2 + v3 \\ v1 + v3 \end{bmatrix}$$

Codes de Hamming : en pratique (4)

En verif : la première colonne de H ajoute les termes en c_0 qui vaut 0.

$$\begin{bmatrix} 0 \\ c1 \\ c2 \\ c3 \\ c4 \\ c5 \\ c6 \end{bmatrix}$$

$$H := \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c1 + c2 + c3 + c4 \\ c2 + c3 + c5 \\ c1 + c3 + c6 \end{bmatrix}$$

Comme c_0 =0, on peut supprimer la première colonne de H

$$\begin{bmatrix} c1 \\ c2 \\ c3 \\ c4 \\ c5 \\ c6 \end{bmatrix}$$

$$H := \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c1 + c2 + c3 + c4 \\ c2 + c3 + c5 \\ c1 + c3 + c6 \end{bmatrix}$$

Codes de Hamming : en pratique résumé (5)

Pour des mots d'information de longueur k autre que 4,11,26, etc.., on considère un codage de Hamming séparable pour des mots de longueur k' = 4,11, 26 immédiatement supérieure (on "ajoute" k'-k bits).

On supprime dans la matrice génératrice les (k'-k) premières lignes et (k'-k) premières colonnes.

On supprime dans la matrice de contrôle les (k'-k) premières colonnes

Le nombre de bits de redondance est le même que pour des mots d'information de longueur k' :

- pour des mots de longueur 2 ou 3, on ajoute 3 bits comme pour des mots de 4 bits
- pour des mots de longueur 5,6,7,8,9,10, on ajoute 4 bits comme pour des mots de 11 bits
- etc…

Codes de Hamming étendus

On ajoute un bit de parité qui est la parité de tous les bits de données.
Ainsi, la distance minimale d passe de 3 à 4.
Toutes les erreurs sur 1 bit, 2 bits et 3 bits peuvent être détectées.
Attention, toujours seules les erreurs sur 1 bit peuvent être corrigées.
Par exemple le code $\mathcal{H}(7,4)$

$$\mathbf{G} := \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

1011 est codé en 0110011

$\mathcal{H}(7,4)$ peut être étendu à $\mathcal{H}(8,4)$ en rajoutant un bit de parité

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

1011 est codé en 01100110

remarque : changement de parité

CODES de REED-MULLER (canaux très bruités)

- Def : le code RM(r,M) est un code linéaire de paramètres $n = 2^M$ et de dimension $k = C_M^0 + C_M^1 + \ldots.. + C_M^r$.

- Construction : La base du code est composée des k vecteurs à 2^M composantes $V_0, \ldots., V_M$ (càd les colonnes de G) tels que :
  - V_0 = 11……11 (il y a 1 vecteur : $1 = C_M^0$)
  - V_i est composé de 2^{i-1} "zéros" suivis de 2^{i-1} "uns" suivis de 2^{i-1} "zéros" etc …, pour i=1,…,M (il y a M vecteurs : $M = C_M^1$)
  - Les autres vecteurs sont obtenus par :
    - $\{V_{i1}.V_{i2} / i1 \neq i2\} \cup \ldots. \cup \{V_{i1}\ldots V_{ir} / ij \neq ik\}$ (il y en a $C_M^2 + \ldots.. + C_M^r$)

- Propriété : La distance minimum d est 2^{M-r} (intéressant pour les canaux très bruités). dém par récurrence.
  - pour RM(1,M) : d = n/2
  - RM(1,5) utilisé pour satellite Mariner
    mots de code de longueur $n=2^5=32$,
    mots d'information de longueur $k = C_5^0+C_5^1=6$,
    (32/2-1)/2=7 erreurs par mot corrigibles

Codes RM(r,4) M=4 => n = 2^4 = 16

V_0 = 1111111111111111
V_4 = 0000000011111111
V_3 = 0000111100001111
V_2 = 0011001100110011
V_1 = 0101010101010101

r=1 (k=5)

$V_{43}=V_4V_3$ = 0000000000001111
$V_{42}=V_4V_2$ = 0000000000110011
$V_{32}=V_3V_2$ = 0000001100000011
$V_{41}=V_4V_1$ = 0000000001010101
$V_{31}=V_3V_1$ = 0000010100000101
$V_{21}=V_2V_1$ = 0001000100010001

r=2 (k=11)

$V_{432}=V_4V_3V_2$ = 0000000000000011
$V_{431}=V_4V_3V_1$ = 0000000000000101
$V_{421}=V_4V_2V_1$ = 0000000000010001
$V_{321}=V_3V_2V_1$ = 0000000100000001

r=3 (k=15)

$V_{4321}=V_4V_3V_2V_1$ = 0000000000000001

r=4 (k=16)

Vecteurs de la base des sev => matrices génératrices : G(RM(1,4)) =

$$\begin{pmatrix}
1&0&0&0&0\\
1&0&0&0&1\\
1&0&0&1&0\\
1&0&0&1&1\\
1&0&1&0&0\\
1&0&1&0&1\\
1&0&1&1&0\\
1&0&1&1&1\\
1&1&0&0&0\\
1&1&0&0&1\\
1&1&0&1&0\\
1&1&0&1&1\\
1&1&1&0&0\\
1&1&1&0&1\\
1&1&1&1&0\\
1&1&1&1&1
\end{pmatrix}$$

Codes RM(2,4) M=4, r=2 => n = 2^4 = 16, k = 11, d = 2^{4-2}=4

v_0 = 1111111111111111
v_4 = 0000000011111111
v_3 = 0000111100001111
v_2 = 0011001100110011
v_1 = 0101010101010101

r=1 (k=5)

$v_{43}=v_4v_3$ = 0000000000001111
$v_{42}=v_4v_2$ = 0000000000110011
$v_{32}=v_3v_2$ = 0000001100000011
$v_{41}=v_4v_1$ = 0000000001010101
$v_{31}=v_3v_1$ = 0000010100000101
$v_{21}=v_2v_1$ = 0001000100010001

r=2 (k=11)

matrice génératrice : G(RM(2,4)) =

1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	1	0
1	0	1	1	0	0	0	1	0	0	0
1	0	1	1	1	0	0	1	0	1	1
1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	1	0	0
1	1	0	1	0	0	1	0	0	0	0
1	1	0	1	1	0	1	0	1	0	1
1	1	1	0	0	0	0	0	0	0	0
1	1	1	0	1	1	0	0	1	1	0
1	1	1	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1

Codes RM propriétés

- Le code orthogonal de RM(r,m) est le code RM(m-r-1,m)
  - le tableau précédent donne la matrice génératrice et la matrice de contrôle H
- Décodage par logique majoritaire :
  - exemple : RM(2,4), n=16, k=11, d=4 (code 1-correcteur)

Soit le mot d'info : $a = a_0a_4a_3a_2a_1a_{43}a_{42}a_{41}a_{32}a_{31}a_{21}$

Soit le mot codé : $x = Ga^t = a_0V_0+a_4V_4+a_3V_3+a_2V_2+a_1V_1+a_{43}V_{43}+a_{42}V_{42}+a_{41}V_{41}+a_{32}V_{32}+a_{31}V_{31}+a_{21}V_{21}$

$= x_0\, x_1\, x_2\, x_3\, x_4x_5x_6\, x_7\, x_8\, x_9\, x_{10}\, x_{11}\, x_{12}\, x_{13}\, x_{14}\, x_{15}$

Soit le mot reçu : $y = y_0\, y_1\, y_2\, y_3\, y_4y_5y_6\, y_7\, y_8\, y_9\, y_{10}\, y_{11}\, y_{12}\, y_{13}\, y_{14}\, y_{15}$

Objectif : reconstruire x à partir de y avec y=x+e

Codes RM : décodage RM(2,4)

$$
\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0\\
1&0&0&1&1&0&0&0&0&0&1\\
1&0&1&0&0&0&0&0&0&0&0\\
1&0&1&0&1&0&0&0&0&1&0\\
1&0&1&1&0&0&0&1&0&0&0\\
1&0&1&1&1&0&0&1&0&1&1\\
1&1&0&0&0&0&0&0&0&0&0\\
1&1&0&0&1&0&0&0&1&0&0\\
1&1&0&1&0&0&1&0&0&0&0\\
1&1&0&1&1&0&1&0&1&0&1\\
1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&0&1&1&0&0&1&1&0\\
1&1&1&1&0&1&1&1&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}
\begin{pmatrix} a0\\ a4\\ a3\\ a2\\ a1\\ a43\\ a42\\ a41\\ a32\\ a31\\ a21 \end{pmatrix}
=
\begin{pmatrix} y0\\ y1\\ y2\\ y3\\ y4\\ y5\\ y6\\ y7\\ y8\\ y9\\ y10\\ y11\\ y12\\ y13\\ y14\\ y15 \end{pmatrix}
$$

On a donc :

$y_0 = a_0$
$y_1 = a_0+a_1$
$y_2 = a_0+a_2$
$y_3 = a_0+a_2+a_1+a_{21}$
$y_4 = a_0+a_3$
$y_5 = a_0+a_3+a_1+a_{31}$
$y_6 = a_0+a_3+a_2+a_{32}$
$y_7 = a_0+a_3+a_2+a_1+a_{32}+a_{31}+a_{21}$
$y_8 = a_0+a_4$
$y_9 = a_0+a_4+a_1+a_{41}$
$y_{10} = a_0+a_4+a_2+a_{42}$
$y_{11} = a_0+a_4+a_2+a_1+a_{42}+a_{41}+a_{21}$
$y_{12} = a_0+a_4+a_3+a_{43}$
$y_{13} = a_0+a_4+a_3+a_1+a_{43}+a_{41}+a_{31}$
$y_{14} = a_0+a_4+a_3+a_2+a_{43}+a_{42}+a_{32}$
$y_{15} = a_0+a_4+a_3+a_2+a_1+a_{43}+a_{42}+a_{41}+a_{32}+a_{31}+a_{21}$

Décodage des codes RM par logique majoritaire

1/ On cherche d'abord les coef. a_{ij} correspondants à V_iV_j. On a :

$a_{21} = y_0+y_1+y_2+y_3$
$= y_4+y_5+y_6+y_7$
$= y_8+y_9+y_{10}+y_{11}$
$= y_{12}+y_{13}+y_{14}+y_{15}$

$a_{31} = y_0+y_1+y_4+y_5$
$= y_2+y_3+y_6+y_7$
$= y_8+y_9+y_{12}+y_{13}$
$= y_{10}+y_{11}+y_{14}+y_{15}$

idem pour a_{32} ,a_{43}, a_{42},a_{41}

On donne à a_{ij} la valeur majoritaire des seconds termes des équations

2/ chercher $y'=y-(a_{43}V_{43}+a_{42}V_{42}+a_{41}V_{41}+a_{32}V_{32}+a_{31}V_{31}+a_{21}V_{21})$, y' ne dépend que de $a_0a_4a_3a_2a_1$

$y'_0 = a_0$
$y'_1 = a_0+a_1$
$y'_2 = a_0+a_2$
$y'_3 = a_0+a_2+a_1$
$y'_4 = a_0+a_3$
$y'_5 = a_0+a_3+a_1$
$y'_6 = a_0+a_3+a_2$
$y'_7 = a_0+a_3+a_2+a_1$

$y'_8 = a_0+a_4$
$y'_9 = a_0+a_4+a_1$
$y'_{10} = a_0+a_4+a_2$
$y'_{11} = a_0+a_4+a_2+a_1$
$y'_{12} = a_0+a_4+a_3$
$y'_{13} = a_0+a_4+a_3+a_1$
$y'_{14} = a_0+a_4+a_3+a_2$
$y'_{15} = a_0+a_4+a_3+a_2+a_1$

Décodage des codes RM par logique majoritaire

3/ chercher les coef. a_i correspondants à V_i. On a

$a_1 = y'_0+y'_1$
$= y'_2+y'_3$
$= y'_4+y'_5$
$= y'_6+y'_7$
$= y'_8+y'_9$
$= y'_{10}+y'_{11}$
$= y'_{12}+y'_{13}$
$= y'_{14}+'y_{15}$

idem pour a_2, a_3, a_4

On donne à a_i la valeur majoritaire des seconds termes des équations

4/ chercher $y'' = y'-(a_4V_4+a_3V_3+a_2V_2+a_1V_1)$
$= a_0V_0$ + erreur

5/ chercher le coef a_0 qui correspond à V_0 : si il y a plus de 0 que de 1, on donne la valeur 0, sinon 1

CODES de BERGER

- Définition : Code de Berger est un code séparable obtenu en rajoutant le nombre de “0” contenus dans I ; le nombre r = n-k de bits de contrôle est donné par $\lceil \log_2 (k+1) \rceil$
  - Ex : 0100101 est codé en 0100101 100
- Erreur unidirectionnelle : uniquement changement de 0 en 1 (ou de 1 en 0) pas de changement simultané de 0 en 1 et de 1 en 0.
- Propriété : Ce code est le code séparable optimal pour la détection des erreurs unidirectionnelles, en ce sens que c’est celui qui nécessite le moins de bits de contrôle pour un nombre donné de bits d’information.
- Un code de Berger est à longueur maximale si le nombre de bits d’information est $k = 2^r-1$ un tel code est complet puisque les r bits de contrôle prennent les 2^r configurations possibles. Les longueurs courantes d’information étant des multiples ou des puissances de 2, les codes de Berger correspondants ne sont pas à longueur maximale et sont donc incomplets.
- Codeurs, détecteurs très simples (compteurs)
- Pas de correction

CODES POLYNOMIAUX

- Suite de bits de p bits : $a_1,a_2,\ldots,a_p$ ⇔ élément de $[GF(2)]^p$ ⇔ coefficients du polynôme $a_1x^{(p-1)}+a_2x^{(p-2)}+\ldots+a_p$
  - ⚠ Numérotation des coef.
  - Ex : 1011 ⇔ x^3+x+1
- P_{p-1} (ensemble des polys de degré < p à coef. dans {0,1}) est isomorphe à $[GF(2)]^p$
  - ⚠ degré du poly = p-1 pour une suite de p bits
- Base canonique : { x^{p-1}, x^{p-2},…..,x^2,x,1}
- Division polynomiale
  - P(x) = D(x).Q(x)+R(x) avec R(x)=0 où deg(R)<deg(D)
  - Ex : $x^3+x^2+x=(x+1)(x^2+1)+1$

    (1 1 1 0) = (0 0 1 1) (0 1 0 1) + (0 0 0 1)

Codes polynomiaux

- Codage : polynôme de P_{k-1} → polynôme de P_{n-1}

- Définition :
  - code linéaire /
  - tous les mots de code sont des multiples de l'un d'eux noté g(x) (au sens produit de polynômes)

- Le polynôme g(x) servant à construire le codage est appelé polynôme générateur

- Un code servant à coder des mots de longueur k par des mots de longueur n est noté Cn,k(g)

Codes polynomiaux

- Fonction de codage
  - soit g(x) de degré s, P_{k-1} l'ensemble des mots d'information i(x), et l'espace vectoriel P_{s+k-1}
  - soit f: $i(x) \xrightarrow{f} i(x)g(x)$
  - f est linéaire et injective
  - f est donc une application de codage de P_{k-1} dans P_{s+k-1}; le code est :
    - linéaire de longueur n = (s+k)
    - engendré par g(x) de degré s= n-k

- Polynôme générateur
  - l'unique polynôme de degré minimal, (n-k) qui est
  - le polynôme codant i(x) = 1 (càd $0.x^{(k-1)}+0x^{(k-2)}+\ldots 0x+1$)
  - => définir le polynôme générateur ⇔ définir le codage de i(x)=1

Codage

- Soit à coder les mots de P_{k-1}
  - Si le degré s de g(x) est fixé, il détermine la longueur du code : n = s + k
  - Si la longueur n du code est fixée, tout polynôme de degré s=n-k engendre un code Cn,k

- Codage par multiplication de polynômes
  - la fonction de codage f(i(x)) = i(x).g(x) permet de construire un code polynomial tel que

$$C_{n,k} = \{c(x) = i(x) .g(x) \text{ avec } i(x) \in P_{k-1} \text{ et } deg(g)=n-k\}$$

Exemple : C(5,3)

- Pour un code polynomial de longueur 5 et de dimension 3 (càd de 3 bits vers 5 bits), le polynôme générateur doit être de degré 5-3=2, ce peut être x^2, x^2+1, x^2+x+1 ou x^2+x
- Soit par exemple $g(x) = x^2+x$, en appliquant $c(x)=i(x).g(x)$ on obtient

i	i(x)	c(x) = i(x).g(x)	c
0 0 0	0	0	0 0 0 0 0
0 0 1	1	x^2+x	0 0 1 1 0
0 1 0	x	x^3+x^2	0 1 1 0 0
0 1 1	x+1	x^3+x	0 1 0 1 0
1 0 0	x^2	x^4+x^3	1 1 0 0 0
1 0 1	x^2+1	$x^4+x^3+x^2+x$	1 1 1 1 0
1 1 0	x^2+x	x^4+x^2	1 0 1 0 0
1 1 1	x^2+x+1	x^4+x	1 0 0 1 0

Matrice génératrice caractéristique

- Code linéaire, matrice génératrice. En codant la base canonique de P_{k-1} par la fonction de codage, on obtient une base de P_{n-1} => matrice génératrice caractéristique (très facile à obtenir)

- Soit {e1,e2,…..,ek} la base canonique de l'espace des mots d'info. i(x)

  $e_1(x) = x^{k-1}$ $\quad e_1 = (1\ 0\ 0\ \ldots\ 0)$

  ….

  $e_{k-1}(x) = x$ $\quad e_{k-1} = (0\ 0\ 0 \ldots 1\ 0)$

  $e_k(x) = 1$ $\quad e_k = (0\ 0\ 0\ \ldots..1)$

  Ce qui donne :

  x^{k-1} codé en $x^{k-1}.g(x)$,

  .....

  x codé en x.g(x)

  1 codé en g(x)

Matrice génératrice caractéristique

- Les polynômes de code obtenu forment un base du code, sous-espace vectoriel de dimension k de P_{n-1}, chacun représente une des k colonnes d'une matrice génératrice G(g) que l'on peut écrire

$$G(g) = \left(\left(x^{k-1}.g(x)\right)...\left(x.g(x)\right)\left(g(x)\right)\right)$$

- Les colonnes de G sont les vecteurs formés par les coef. des polys

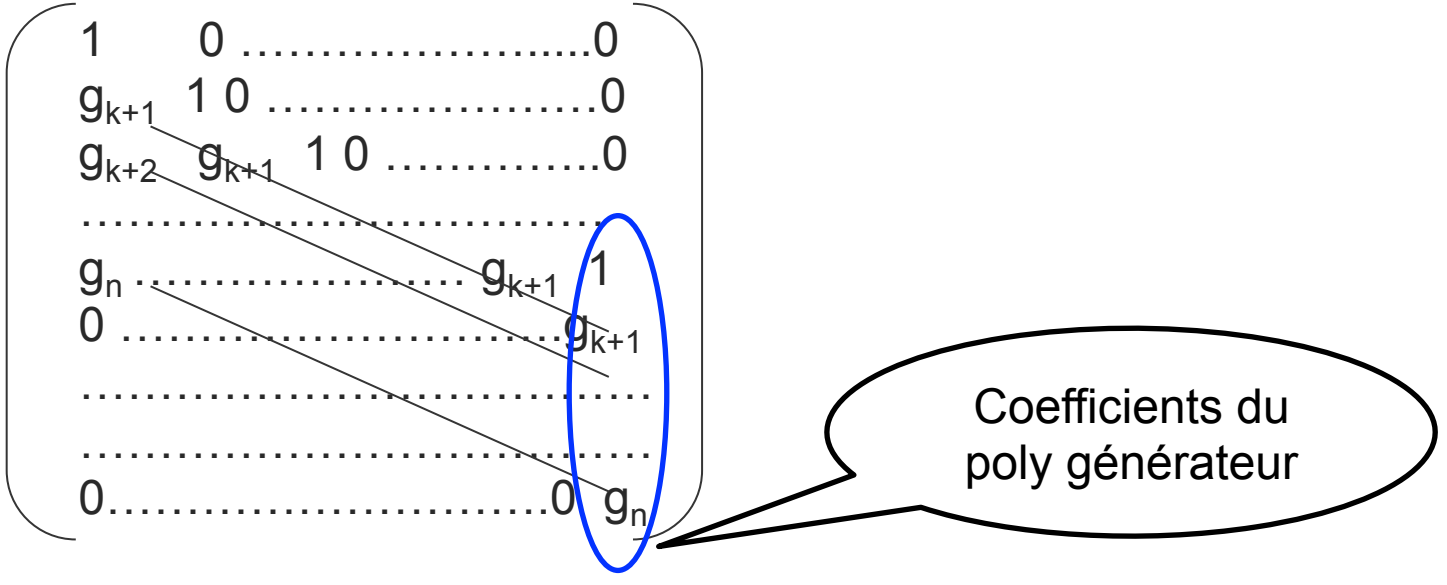
- Si $g(x)= x^{n-k}+g_{k+1}.x^{n-(k+1)}+\ldots+g_{n-1}.x+g_n$,  numérotation coefs

il vient :

  - (0 …. 0 1 g_{k+1} ……..…g_n) qui correspond à g(x)
  - (0 …. 1 g_{k+1} ……g_n 0) qui correspond à x.g(x)
  - ……
  - (1 g_{k+1} ……g_n 0 …….0) qui correspond à x^{k-1}.g(x)

Matrice génératrice caractéristique

- La matrice G(g) est donc de la forme



- Exemple : sur l'exemple précédent $g(x) = x^2+x$ qui correspond au vecteur (0 0 1 1 0) de $[GF(2)]^5$, càd
  - $g_n = 0$
  - $g_{n-1} = g_{k+1} = 1$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Codage systématique (par décalage et division)

- Soit g(x) de degré s avec s=n-k, le codage se fait en 3 étapes (multipli.(décalage) puis division de poly)

- Soit i(x) = $i_1.x^{k-1}+\ldots.+i_k$. En codage systématique, tout poly de code est de la forme :
  - $c(x) = (i_1.x^{n-1}+\ldots.+i_k x^{n-k}) + a_1.x^{n-k-1}+\ldots..+a_{n-k+1}.x+a_{n-k}$

    $= x^{n-k}\,(i_1.x^{k-1}+\ldots.+i_k) + \underbrace{a_1.x^{n-k-1}+\ldots..+a_{n-k+1}.x+a_{n-k}}_{\text{clef de contrôle}}$

soit $c(x) = x^{n-k}\, i(x) + a(x)$ avec a(x) =0 ou deg(a(x)) < (n-k) avec a(x) (poly associé à la clef de contrôle).

Codage systématique (par multiplication et division)

- Codage

  1/ calculer le produit $x^{n-k}.i(x)$ (cela revient à décaler les coef. de i(x) vers la gauche de n-k positions)

  2/ calcul de la clef de contrôle a(x)

  - il faut que $x^{n-k}\ i(x) + a(x)$ soit un mot du code i.e. de la forme $i^*(x).g(x)$ avec $i^*(x) \in P_{k-1}$
  - ce qui implique $x^{n-k}\ i(x) = i^*(x).g(x) + a(x)$
    - on a deg(a(x)) = 0 ou deg(a(x)) < (n-k)
    - donc a(x) est le reste de la division de $x^{n-k}.i(x)$ par g(x)

  3/ concaténer $x^{n-k}\ i(x)$ et a(x)

  - En résumé : $c(x) = x^{n-k}.i(x) + (x^{n-k}.i(x) \bmod g(x))$

Règle du code systématique

- Pour construire Cn,k de générateur g(x) par un codage systématique, il faut pour chaque i(x) :
  - 1/ calculer le produit x^{n-k} i(x) => coefs de plus haut degré
  - 2/ diviser x^{n-k} i(x) par g(x), le reste de la division est la clef a(x) associée à i(x) => coefs de plus petit degré
  - 3/ ajouter (ce qui revient à concaténer) x^{n-k} i(x) et a(x)
- Remarque : tout code polynomial peut être construit par un codage systématique
  - En effet, le codage systématique repose sur le poly. gén. g(x). Or il n'existe qu'un seul code, pour n et k fixé défini par un poly. g(x) donné.

Codage systématique : exemple

Soit le code précédent C5,3 de poly $g(x) = x^2+x$

- 1/ Calcul de $x^{n-k}\, i(x) = x^{5-3}.i(x) = x^2.i(x)$
- 2/Tableau des clefs a(x) par : $x^2.i(x) = (x^2+x).Q(x) + a(x)$

i	i(x)	$x^2.i(x)$	$=(x^2+x)\ Q(x) \quad +a(x)$	a(x)
0 0 0	0	0	$=(x^2+x)\ 0$	0
0 0 1	1	x^2	$=(x^2+x)\ 1 \quad +x$	x
0 1 0	x	x^3	$=(x^2+x)\ (x+1) \quad +x$	x
0 1 1	x+1	x^3+x^2	$=(x^2+x)\ x$	0
1 0 0	x^2	x^4	$=(x^2+x)\ (x^2+x+1) \quad +x$	x
1 0 1	x^2+1	x^4+x^2	$=(x^2+x)\ (x^2+x)$	0
1 1 0	x^2+x	x^4+x^3	$=(x^2+x)\ x^2$	0
1 1 1	x^2+x+1	$x^4+x^3+x^2$	$=(x^2+x)\ (x^2+1) \quad +x$	x

Codage systématique : exemple

- 3/ Codage par : $c(x)=x^2.i(x)+a(x)$

i	$x^2.i(x)$	a(x)	$c(x)=x^2.i(x)+a(x)$	c
0 0 0	0	0	0+0	0 0 0 0 0
0 0 1	x^2	x	x^2+x	0 0 1 1 0
0 1 0	x^3	x	x^3+x	0 1 0 1 0
0 1 1	x^3+x^2	0	x^3+x^2	0 1 1 0 0
1 0 0	x^4	x	x^4+x	1 0 0 1 0
1 0 1	x^4+x^2	0	x^4+x^2	1 0 1 0 0
1 1 0	x^4+x^3	0	x^4+x^3	1 1 0 0 0
1 1 1	$x^4+x^3+x^2$	x	$x^4+x^3+x^2+x$	1 1 1 1 0

Comparer avec le code obtenu par multiplication par g(x) :

i	c
0 0 0	0 0 0 0 0
0 0 1	0 0 1 1 0
0 1 0	0 1 1 0 0
0 1 1	0 1 0 1 0
1 0 0	1 1 0 0 0
1 0 1	1 1 1 1 0
1 1 0	1 0 1 0 0
1 1 1	1 0 0 1 0

Détection d'erreurs

- Solution 1 : Par matrice de contrôle (cf. codes linéaires)
- Solution 2 : Par division de polynômes :
  - Soit m(x) le message reçu
    - $m(x) = g(x)\,Q(x) + R(x)$ avec $R(x) = 0$ ou $\deg(R) < \deg(g)$
    - si $R(x) = 0$, m(x) est un polynôme du code, m(x) accepté
    - si $R(x) \neq 0$, m(x) est un message erroné
- Fonction syndrome
  - soit $s = \deg(g)$, $R(x) \in P_{s-1}$, $R(x) = m(x) \bmod g(x)$
  - l'application qui à chaque m(x) fait correspondre le reste de la division de m(x) par g(x) est une application linéaire de P_{n-1} dans P_{s-1} dont le code est le noyau, propre au code polynomial et toute matrice associée à cette application est une matrice de contrôle du code

Capacité de détection

- Possibilités de détection liées au poly. générateur
- Soit m(x) un message erroné provenant du code c(x), m(x) = c(x) + e(x) où e(x) = $e_1x^{n-1}+e_2x^{n-2}+..+e_n$ poly d'erreur de transmission
- Un message erroné est reconnu comme tel ssi e(x) n'est pas un polynôme du code càd si g(x) ne divise pas e(x)

- Erreurs de poids 1
  - La détection de tous les messages ayant une erreur de poids 1 est assurée si le poly générateur a au moins 2 termes non nuls
    - une erreur de poids 1 correspond à un poly e(x) = x^l, càd un monome
    - un poly avec deux termes non nuls ne pouvant diviser un monôme, g(x) ne peut diviser e(x) de la forme x^l

Capacité de détection

- Erreurs de poids 2
  - La détection de tous les messages ayant une erreur de poids 2 est assurée si le poly générateur ne divise aucun poly de la forme x^j+x^k pour tout j et k tels que $1 \leq j-k \leq n-1$
    - une erreur de poids 2 correspond à un poly $e(x) = x^j+x^k = x^k (x^{j-k}+1)$. Il faut donc que g(x) ne divise aucun poly de cette forme pour toute valeur de j et k, càd $0 \leq k < j \leq n-1$ càd $j-k \leq n-1$

- Erreurs de poids impair
  - La détection de tous les messages ayant une erreur de poids impair est assurée si le poly générateur est un multiple de (x+1)
    - dem : si $g(x)=(x+1)\,Q(x)$ alors x=1 est racine de g(x)
    - un poly d'erreurs de poids impair contient donc un nombre impair de termes non nuls, il vaut donc 1 pour x=1. Il ne peut pas être divisible par x+1

Capacité de détection

- Erreurs en salves (càd plusieurs bits successifs erronés)
- Salves courtes (de longueur t ≤ s, degré de g(x), s=n-k)
  $e(x) = x^j(x^{t-1}+\ldots+1)$ (Longueur t, commençant à la position j)
  - Si g(x) a le terme en x^0 , il ne divise pas x^j
  - Comme t ≤ s, g(x) ne divise pas $(x^{t-1}+\ldots+1)$

  => Si g(x) a le terme en x^0, il ne divise pas e(x)
  Toutes les erreurs jusqu'à la longueur s sont détectées
- Salves longues (Longueur t = s+1)
  Indétectable si seulement si la salve est identique à g(x)
  $g(x) = x^s+ \ldots + 1$: s-1 bits entre x^s and x^0
  $e(x) = 1 + \ldots + 1$ doivent coïncider
  Probabilité de ne pas détecter l'erreur : $2^{-(s-1)}$
- Salves plus longues (longueur t > s+1)
  Probabilité de ne pas détecter l'erreur : 2^{-s}

Capacité de détection : exemple

- Soit le code précédent engendré par $g(x) = x^2+x$ (n=5)
  - Ayant 2 termes il détecte toutes les erreurs sur 1 bit
  - Puisque toutes les erreurs x^k sont détectées, pour que les erreurs de poids 2 soient détectées, il faut que g(x) ne divise aucun poly de la forme $x^{j-k}-1$, avec $1 \leq j-k \leq 4$
    - x^2+x ne divise pas $x+1$
    - x^2+x ne divise pas x^2+1 puisque $x^2+1 = (x^2+x)\ 1+ (x+1)$
    - x^2+x ne divise pas x^3+1 puisque $x^3+1 = (x^2+x)\ x + (x^2+1)$
    - x^2+x ne divise pas x^4+1 puisque $x^4+1 = (x^2+x)\ (x^2+x+1)+ (x+1)$

    Donc toutes les erreurs de poids 2 sont détectées
  - x^2+x est multiple de $x+1$, donc toutes les erreurs de poids impairs sont détectées
  - Toutes les salves erronées de 2 bits sont détectées

Codes polynomiaux : en résumé (1)

- Codage :
  - par multiplication :
    - $c(x) = i(x).g(x)$
  - par division (si code systématique)
    - $c(x) = x^{n-k}.i(x) + R(x)$ avec $R(x)$= Reste $(x^{n-k}.i(x) / g(x))$
- Contrôle :
  - par division par g(x)
  - par matrice de contrôle
  - algèbre des poly modulo un poly p(x) de degré n g(x)
    - si $p(x) = g(x)*\gamma(x)$
    - $g(x)*\gamma(x) = 0 \bmod p(x) \Leftrightarrow$ c(x) appartient au code

Codes polynomiaux : en résumé (2)

- Détection d'erreur
  - dépend de la nature du poly. générateur
- Correction
  - capacité de correction dépend de la distance, difficile à déterminer
  - particulariser p(x) pour évaluer rapidement la capacité de correction => codes cycliques
  - choisir g(x) pour obtenir une capacité de correction minimum garantie => codes BCH

CRC exemples

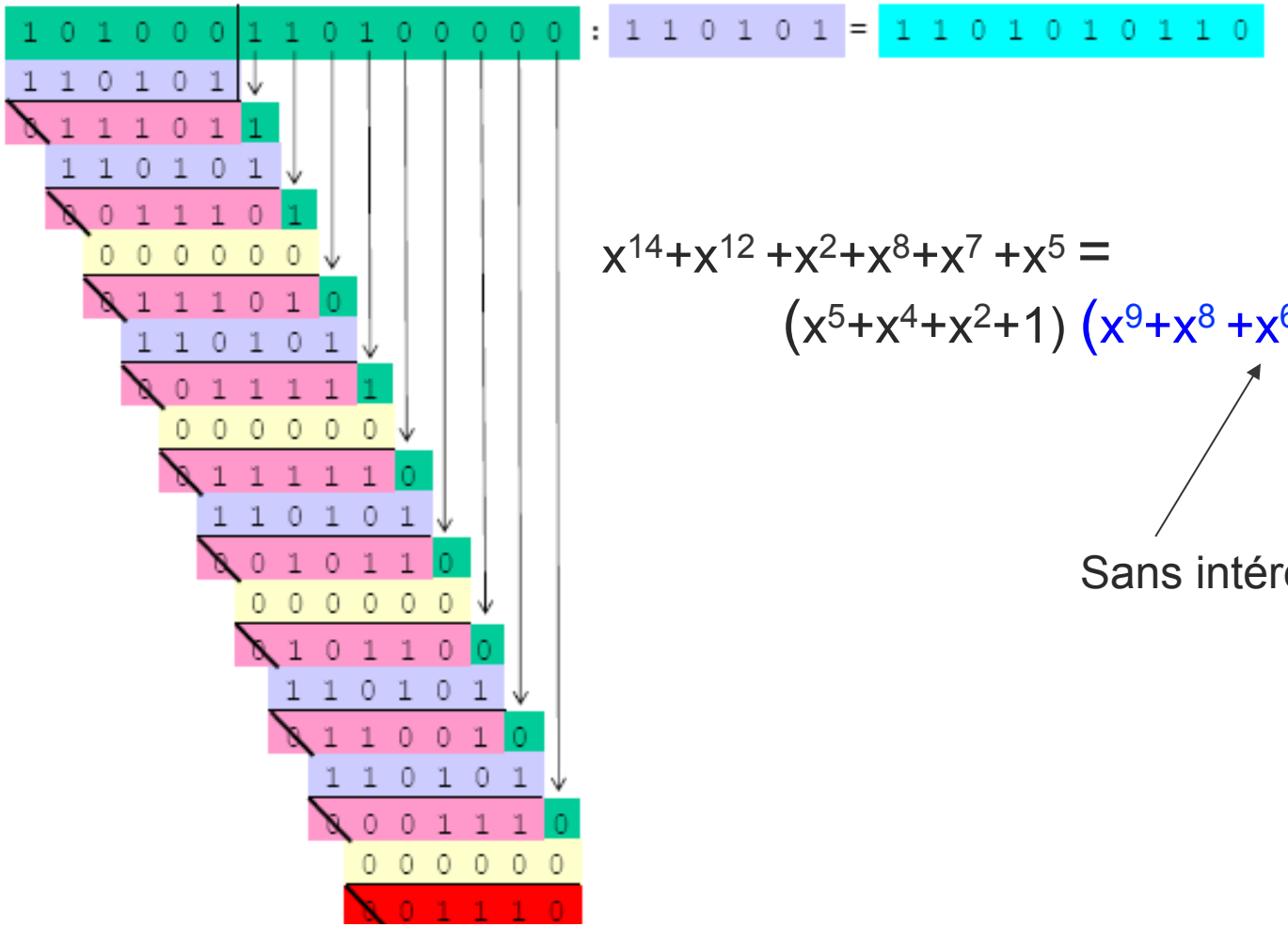
- Exemples:
  - CRC-12 $= x^{12}+x^{11}+x^3+x^2+x+1$
    CRC-16 $= x^{16}+x^{15}+x^2+1$
    CRC-CCITT $= x^{16}+x^{12}+x^5+1$
  - CRC-16 et CRC-CCITT (téléphonie sans fil) détectent
    - Erreurs simples et doubles
    - Erreurs impaires
    - Salves de 16 bits ou moins
    - 99.997% des salves de 17 bits
    - 99.998% des salves de 18 bits ou plus

CRC usuels

Nom	Polynôme générateur
CRC-1	$x + 1$ (Bit de parité)
CRC-4-ITU	$x^4 + x + 1$ (ITU-T G.704)
CRC-5-EPC	$x^5 + x^3 + 1$ (Gen 2 RFID)
CRC-5-ITU	$x^5 + x^4 + x^2 + 1$ (ITU-T G.704)
CRC-5-USB	$x^5 + x^2 + 1$ (USB token packets)
CRC-6-ITU	$x^6 + x + 1$ (ITU-T G.704)
CRC-7	$x^7 + x^3 + 1$ (telecom systems, ITU-T G.707, ITU-T G.832, MMC,SD)
CRC-8-CCITT	$x^8 + x^2 + x + 1$ (ATM HEC), ISDN Header Error Control and Cell Delineation ITU-T I.432.1 (02/99)
CRC-8-Dallas/Maxim	$x^8 + x^5 + x^4 + 1$ (1-Wire bus)
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$
CRC-8-SAE J1850	$x^8 + x^4 + x^3 + x^2 + 1$
CRC-8-WCDMA	$x^8 + x^7 + x^4 + x^3 + x + 1$[
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$ (ATM; ITU-T I.610)
CRC-11	$x^{11} + x^9 + x^8 + x^7 + x^2 + 1$ (FlexRay)
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$ (telecom systems)
CRC-15-CAN	$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$ (Bisync, Modbus, USB, ANSI X3.28, many others; also known as *CRC-16* and *CRC-16-ANSI*)
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$ (X.25, HDLC, XMODEM, Bluetooth, SD, many others; known as *CRC-CCITT*)
CRC-16-T10-DIF	$x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (SCSI DIF)
CRC-16-DNP	$x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^2 + 1$ (DNP, IEC 870, M-Bus)
CRC-16-DECT	$x^{16} + x^{10} + x^8 + x^7 + x^3 + 1$ (telephones sans fil)
CRC-24	$x^{24} + x^{22} + x^{20} + x^{19} + x^{18} + x^{16} + x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^3 + x + 1$ (FlexRay[)
CRC-24-Radix-64	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$ (OpenPGP)
CRC-30	$x^{30} + x^{29} + x^{21} + x^{20} + x^{15} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^2 + x + 1$ (CDMA)
CRC-32-IEEE 802.3	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (V.42, MPEG-2, PNG[19], POSIX cksum)
CRC-32C (Castagnoli)	$x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1$ (iSCSI & SCTP, G.hn payload)
CRC-32K (Koopman)	$x^{32} + x^{30} + x^{29} + x^{28} + x^{26} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{11} + x^{10} + x^7 + x^6 + x^4 + x^2 + x + 1$
CRC-32Q	$x^{32} + x^{31} + x^{24} + x^{22} + x^{16} + x^{14} + x^8 + x^7 + x^5 + x^3 + x + 1$ (aviation; AIXM[)
CRC-64-ISO	$x^{64} + x^4 + x^3 + x + 1$ (HDLC — ISO 3309, Swiss-Prot/TrEMBL; considered weak for hashing
CRC-64-ECMA-182	$x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$ (as described in ECMA-182)

Implantation du reste de la division de polynômes dans $(GF[2])^n$

reste de la division de $x^{14}+x^{12}+x^2+x^8+x^7+x^5$ par $x^5+x^4+x^2+1$



$$x^{14}+x^{12}+x^2+x^8+x^7+x^5 =$$
$$(x^5+x^4+x^2+1)\,(x^9+x^8+x^6+x^4+x^2+x) + (x^3+x^2+x)$$

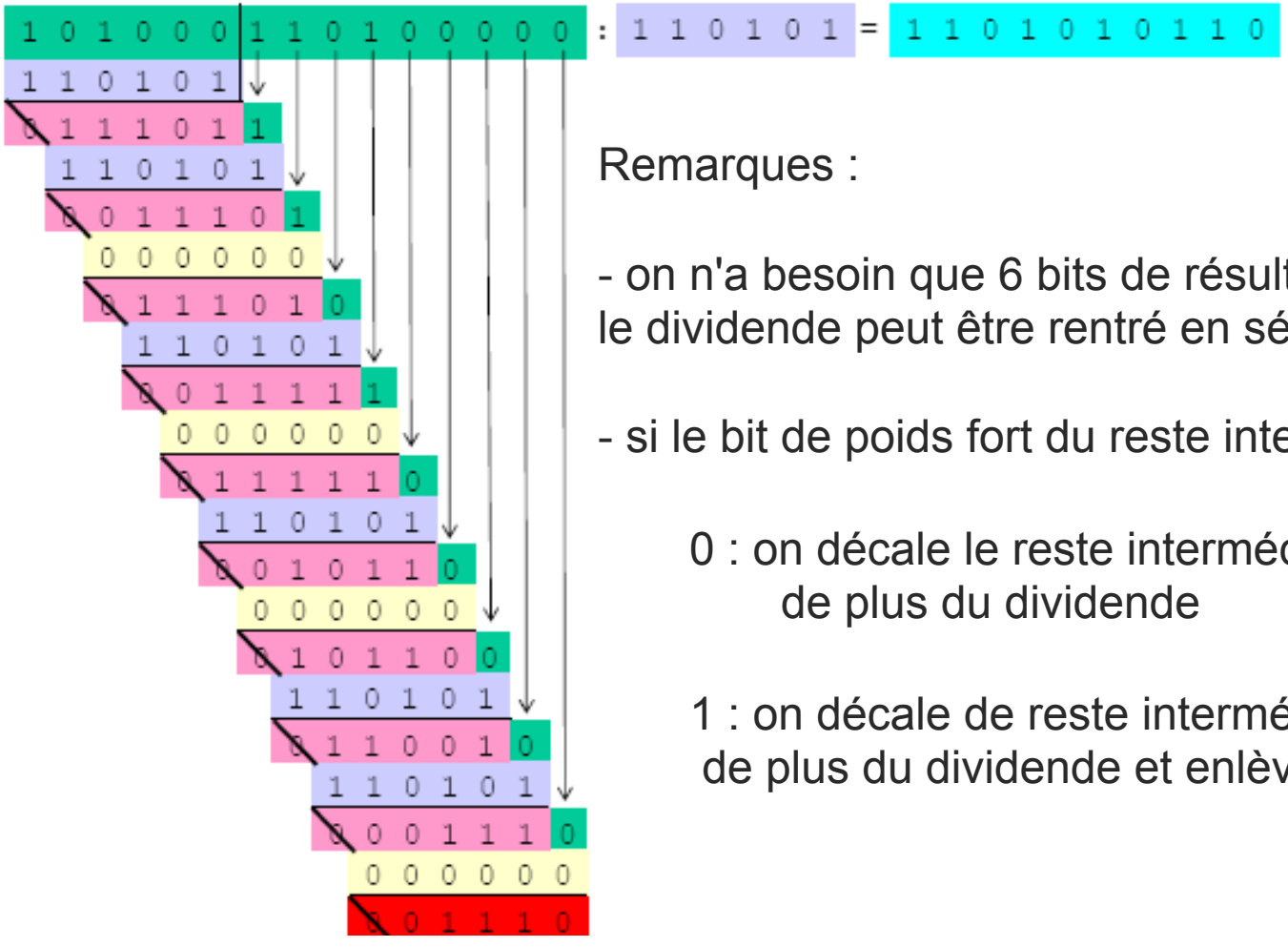
Sans intérêt

Syndrome

= Reste

Implantation du reste de la division de polys

division de $x^{14}+x^{12}+x^{8}+x^{7}+x^{5}$ par $x^{5}+x^{4}+x^{2}+1$



Remarques :

- on n'a besoin que 6 bits de résultats intermédiaire, le dividende peut être rentré en série

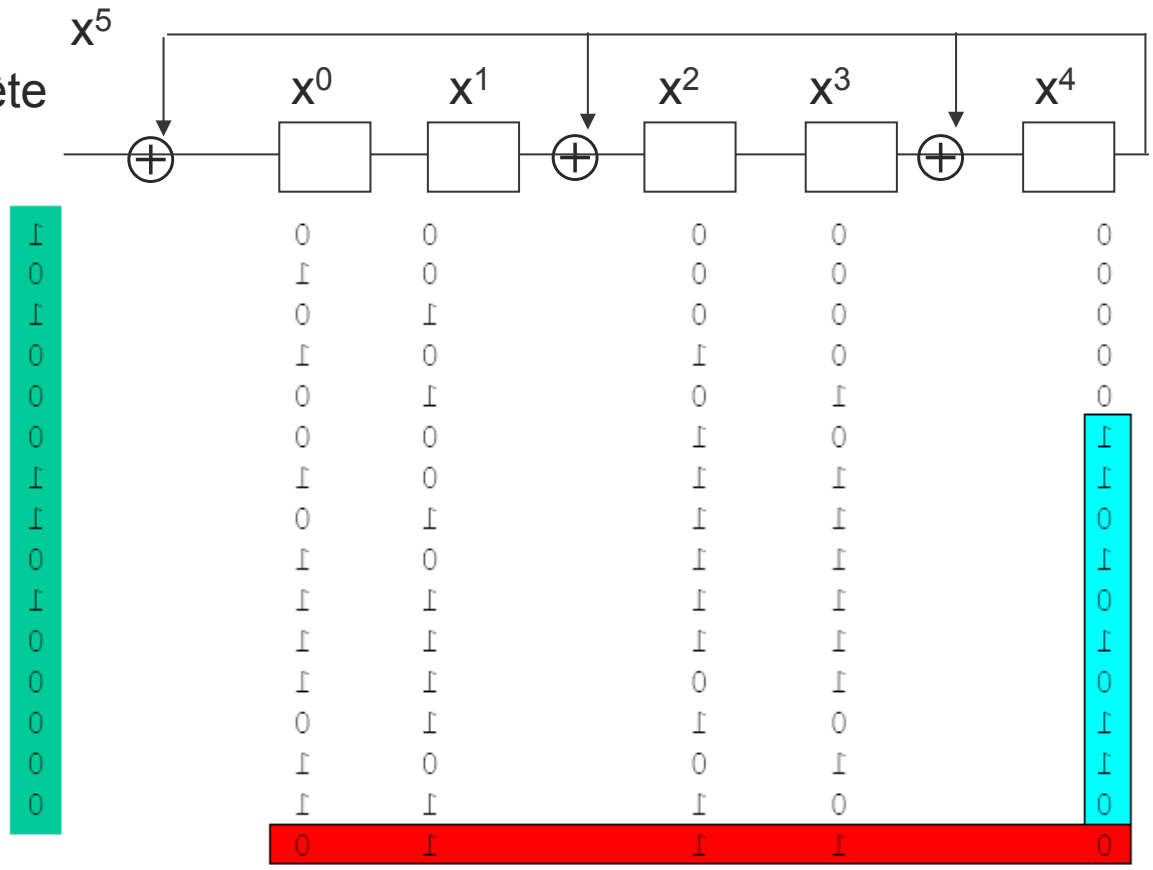
- si le bit de poids fort du reste intermédiaire est

  0 : on décale le reste intermédiaire en incluant un bit de plus du dividende

  1 : on décale de reste intermédiaire en incluant un bit de plus du dividende et enlève (i.e. ajoute) le diviseur

Implantation du reste de la division de polys

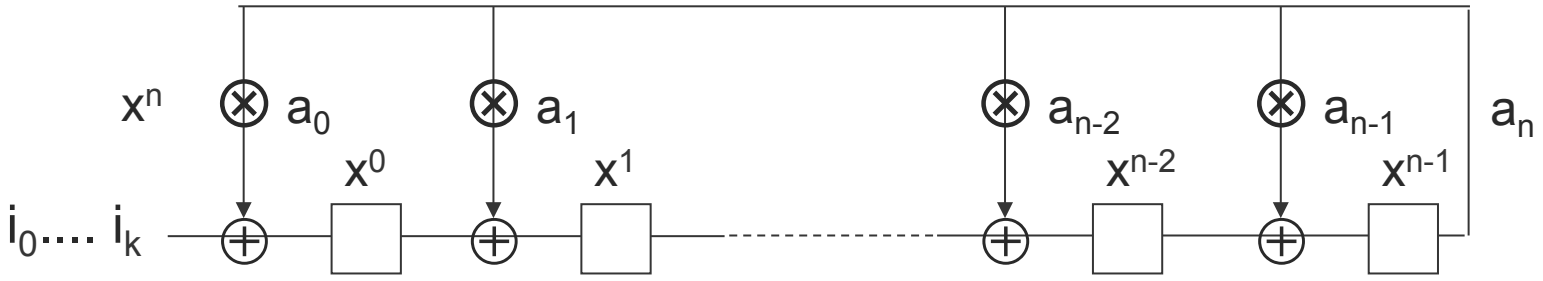
Entrée du dividende, puissance forte en tête
$i_0 \ldots i_k$



Remarques :
- si le bit de poids fort = 0, décalage
- si le bit de poids fort =1, décalage et addition du diviseur (càd soustraction)

Forme générale d'un diviseur

Division par $a_0 + a_1x + \ldots\ldots + a_nx^n$



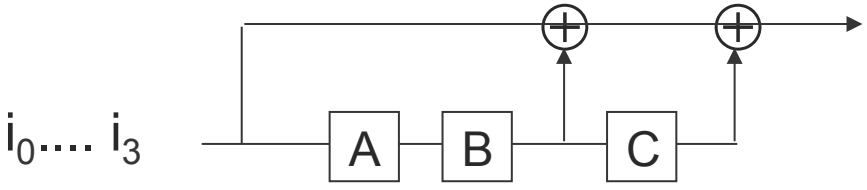
la connexion existe si $a_i = 1$
la connexion n'existe pas si $a_i = 0$

Codage : Implantation multiplication de polynômes

- $c(x) = i(x).g(x)$

Exemple : $g(x) = x^3+x+1$, $i(x) = x^3+x^2+1$

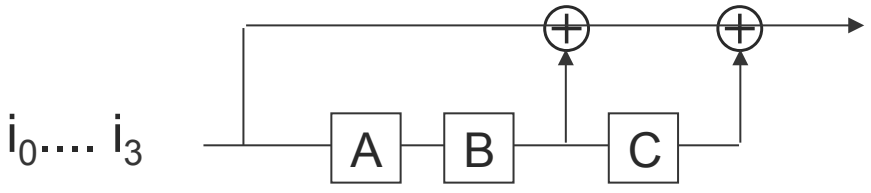
Initialement les bascules sont à 0



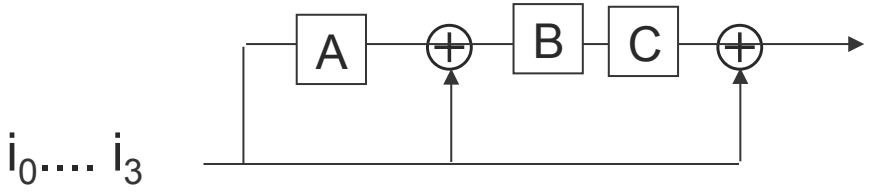
t	E	ABC	Sortie	
0		000		
1	i3 =1	100	1	x^6
2	i2=1	110	1	x^5
3	i1=0	011	1	x^4
4	i0=1	101	1	x^3
5	0	010	1	x^2
6	0	001	1	x^1
7	0	000	1	x^0

Codage : Implantation multiplication

Exemple : $g(x) = x^3+x+1$



Autre implantation



CODES CYCLIQUES ou CRC (Cyclic Redundancy Check)

- Codes polynomiaux + propriété de stabilité par permutation circulaire des mots. Leur polynôme générateur divise x^n+1 (n longueur du code). => construction immédiate de la matrice de contrôle
- Définition : un code est dit cyclique si
  - il est linéaire
  - il est stable par permutation circulaire des mots.
- Notation
  - $\forall$ k , c $\in$ C $\Leftrightarrow$ $\sigma^k(c) \in C$, où σ^k est une rotation de k positions

Exemple de code C(4,3) cyclique

Mots d'info	Mots de code	Permutés circulaires			
0 0 0	0 0 0 0	0 0 0 0			
0 0 1	0 0 1 1	0 0 1 1	1 0 0 1	1 1 0 0	0 1 1 0
0 1 0	0 1 0 1	0 1 0 1	1 0 1 0		
0 1 1	0 1 1 0	0 1 1 0	0 0 1 1	1 0 0 1	1 1 0 0
1 0 0	1 0 0 1	1 0 0 1	0 1 1 0	0 0 1 1	1 1 0 0
1 0 1	1 0 1 0	1 0 1 0	0 1 0 1		
1 1 0	1 1 0 0	1 1 0 0	0 1 1 0	0 0 1 1	1 0 0 1
1 1 1	1 1 1 1	1 1 1 1			

Forme polynomiale d'un code cyclique

- Soit c(x) un mot du code c=(c1,c2,….,cn) $c(x)= c_1x^{n-1}+c_2x^{n-2}+..+c_n$

- Une permutation vers la gauche transforme c(x) en c*(x)
  avec $c^*(x) = c_2x^{n-1}+c_3x^{n-2}+..+c_nx+c_1$
  $c^*(x) = x\,(c_2x^{n-2}+c_3x^{n-3}+..+c_n)+c_1 = x\,(c(x)-c_1x^{n-1})+c_1 = x\,(c(x)+c_1x^{n-1})+c_1 =$
  $(x^n+1)c_1+ xc(x)$
  - donc $c^*(x) = (x^n+1)c_1+ xc(x)$ et $xc(x) = (x^n+1)c_1+ c^*(x)$

- c*(x) est soit nul si c(x) est nul soit de degré inférieur ou égal à n => c*(x) est le reste de la division euclidienne de x.c(x) par (x^n+1) càd
  $x.c(x) \mod (x^n+1) = c^*(x) \in C$

- La relation de cyclicité devient (pour un nombre quelconque k de rotations)
  $\forall k , c(x) \in C \Leftrightarrow x^k.c(x) \bmod (x^n+1) \in C$

Forme polynomiale d'un code cyclique

- Un code cyclique étant linéaire, la somme de polynômes $x^k.c(x) \bmod(x^n+1)$ est un mot du code càd
  - $\forall\ a(x) = \sum a_{n-k}\ x^k$ avec $a_{n-k} \in \{0,1\}$
  - $c(x) \in C \Leftrightarrow a(x).c(x) \bmod(x^n+1) \in C$
- Cette relation est vraie en particulier pour tout $a(x) \in P_{n-1}$
- => Son poly générateur divise x^n+1 (dem. à faire)

- Définition 2 : un code est cyclique ssi
  - c'est un code polynomial
  - son poly générateur g(x) divise x^n+1

Exemple

- Soit le code C(4,3) précédent, par définition son générateur est le polynôme codant i(x)=1 associé à (0 0 1)

(0 0 1) est codé par (0 0 1 1) qui correspond à x+1 ➔ g(x) = x+1

(x+1) divise bien (x^4+1) :

$$(x^4+1) = (x+1)(x^3+x^2+x+1) + 0$$

Mots d'info	Mots de code
0 0 0	0 0 0 0
0 0 1	0 0 1 1
0 1 0	0 1 0 1
0 1 1	0 1 1 0
1 0 0	1 0 0 1
1 0 1	1 0 1 0
1 1 0	1 1 0 0
1 1 1	1 1 1 1

Matrice génératrice

- Comme tout code polynomial, le code cyclique C(n,r) de générateur g(x) admet pour base les r polynômes $x^{r-1}g(x),\ldots, xg(x),g(x)$
- Tous les polynômes de cette base, multiples de g(x), sont de degré inférieur à n, il sont également multiples de g(x) modulo x^n+1. Ils sont associés aux vecteurs : $\sigma^{k-1}(g)$, …., $\sigma^2(g)$, $\sigma(g)$, g. Donc la matrice peut s'écrire :

$$G(g) = \left[\left(\sigma^{k-1}.g(x)\right)..\left(\sigma.g(x)\right)\left(g(x)\right)\right]$$

- Remarque : $g(x) = g_k x^{n-k}+..+g_n$ est tel que :
  - $g_k = 1$ comme tout code polynomial
  - $g_n = 1$, sinon g(x) serait divisible par x et alors x^n+1 serait divisible par x, ce qui faux
  - Le premier et le dernier coef. de g(x) valent donc 1

Exemple

- Pour le code précédent C(4,3), les 3 vecteurs exprimés dans la base canonique de $[GF(2)]^4$:
- $\sigma^2(g)=(1\ 1\ 0\ 0)$, $\sigma(g)=(0\ 1\ 1\ 0)$, et $g = (0\ 0\ 1\ 1)$ constituent une base , le code admet comme matrice génératrice :

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & \mathbf{1} \\ 0 & 0 & \mathbf{1} \end{pmatrix}$$

Contrôle des codes cycliques

- Théorème : soit p(x) tel que $x^n+1 = p(x).g(x)$.
  $c(x) \in C \Leftrightarrow p(x).c(x) \bmod(x^n+1) = 0$
  p(x) est appelé le polynôme de contrôle

- *Facilité des calculs mod(x^n+1)*
- Polynôme de contrôle :

$p(x) = p_{n-r}\, x^k+\ldots.+p_n$

$c(x) = c_1x^{n-1}+\ldots. +c_{n-k}\, x^k+\ldots.+c_n$

$p(x).c(x) = A_1x^{N-1}+\ldots. +A_{N-k}\, x^k+\ldots.+A_N$ avec N=n+k (equ.1)

on remarque que pour $0 \le j \le k-1$

$x^{n+j} = (x^n+1).x^j+x^j$

$x^{n+j} \bmod (x^n+1) = x^j$ d'où les différents $x^i \bmod (x^n+1)$ dans equ. 1

x^j	x^{n+k-1}	…	x^{n+j}	…	x^n	x^{n-1}	…	x^k	x^{k-1}	…	x^j	…	1
$x^j \bmod (x^n+1)$	x^{k-1}	…	x^j	…	1	x^{n-1}	…	x^k	x^{k-1}	…	x^j	…	1

Contrôle des codes cycliques

- Donc dans $p(x).c(x) \bmod(x^n+1)$ le terme en x^j est égal :
  - si $0 \le j \le k-1$, à la somme des termes en x^j et x^{n+j} figurant dans $p(x).c(x)$
  - si $k \le j \le n-1$, au terme en x^j figurant dans $p(x).c(x)$

- Autrement dit si $p(x).c(x) \bmod (x^n+1) = M_1 x^{n-1} + \ldots + M_n$

  $M_{n-j}x^j = A_{N-j}x^j + A_{N-(n+j)}\,x^{n+j}$ pour $0 \le j \le k-1$

  $= A_{N-j}x^j$ pour $k \le j \le n-1$

Exemple :

Le code C4,3 de générateur $g(x)=(x+1)$. On a $x^4+1= (x^3+ x^2+x+1)\,(x+1)$.

Le poly de contrôle est $p(x)= x^3+ x^2+x+1$.

On raisonne donc modulo (x^4+1)

1/ Soit le poly de code $c(x)= x^3+1$, on a :

$p(x).c(x)= (x^6+ x^5+x^4)+ 0x^3+(x^2+x+1)$

Or $x^6= x^{4+2} \Rightarrow x^6 \bmod (x^4+1)=x^2$,

$x^5= x^{4+1} \Rightarrow x^5 \bmod (x^4+1)= x$,

$x^4 \Rightarrow x^4 \bmod (x^4+1)= 1$.

Donc $p(x)c(x) \bmod (x^4+1)= (x^2+x+1)+(x^2+x+1)=0$

2/ Soit le message $m(x)= x^3+ x+1$, $p(x).m(x) = (x^6+ x^5)+x^3+(1)$.

Donc $p(x).m(x) \bmod (x^4+1)= (x^2+ x)+x^3+1 \neq 0 \Rightarrow m(x) \notin C$

Codes cycliques : Piégeage des erreurs corrigibles

- Une erreur est corrigible si son poids est inférieur à t (capacité de correction). Principe "piéger" les erreurs lorsqu'elles sont corrigibles.
- Remarques
  - un message m présente une erreur sur la première composante ssi son vecteur d'erreur e (inconnu mais de même syndrome) possède le chiffre 1 en 1^{ere} position.
  - si un message m provient du mot de code c, avec une erreur de poids k, tout permuté $\sigma^j(m)$ provient de $\sigma^j(c)$ avec une erreur de même poids que égale à $\sigma^j(e)$ (évident puisque $\sigma^j()$ est linéaire)
  - Il est donc évident que
    - $\sigma^j(c)$ est un mot du code cyclique
    - $w(\sigma^j(e))= w(e) = k$
  - Donc si m a une erreur e, un de ses permutés a une erreur commençant par le chiffre 1
  - rappel : chaque erreur e corrigible est unique et de poids minimum

Codes cycliques : Piégeage des erreurs corrigibles

- Algorithme 1
  - On ne retient du tableau standard que :
    - les erreurs corrigibles et commençant par 1
    - leurs syndromes => Table de correction
  - Le message m étant corrigible, lui ou l'un de ses permutés $\sigma^j(m)$ a une erreur commençant par 1 figurant dans la table (seule erreur de poids minimum dans sa classe de syndrome).
  - L'erreur e* étant repérée, l'erreur se déduit par
    - $e(m)= \sigma^{n-j}(e^*)$

Codes cycliques : Piégeage des erreurs corrigibles

Exemple : soit le code cyclique C(5,1) de générateur $x^4+x^3+x^2+x+1$, il correspond au codage : 0 -> (0 0 0 0 0) ; 1 -> (1 1 1 1 1). La distance minimale étant 5 sa capa de correction est 2 (5-1/2).

Table de correction : on rassemble toutes les erreurs de poids inférieur à 2, commençant par 1 et leur syndrome, calculé par exemple comme le reste de la division de e(x) par g(x) càd S(e)= e(x) mod $(x^4+x^3+x^2+x+1)$

e		S(e)	
1 0 0 0 0	x^4	x^3+x^2+x+1	1 1 1 1
1 1 0 0 0	x^4+x^3	x^2+x+1	0 1 1 1
1 0 1 0 0	x^4+x^2	x^3+x+1	1 0 1 1
1 0 0 1 0	x^4+x	x^3+x^2+1	1 1 0 1
1 0 0 0 1	x^4+1	x^3+x^2+x	1 1 1 0

Soit c=(1 1 1 1 1) le mot émis et m le message reçu comportant 2 erreurs, m=(1 0 1 0 1), càd m(x)= x^4+x^2+1.

En appliquant l'algo, on a :

1/ $\sigma^0(m)$=m et S(m)= x^3+x qui $\notin$ T.

2/ $\sigma^1(m)$=(0 1 0 1 1) = x^3+x+1 et

S($\sigma^1(m)$)= x^3+x+1 qui $\in$**T.**

$\sigma^1(m)$ a pour erreur 1 0 1 0 0

Donc m a pour erreur σ^{5-1}(1 0 1 0 0) =(0 1 0 1 0)

Le message est corrigé en :

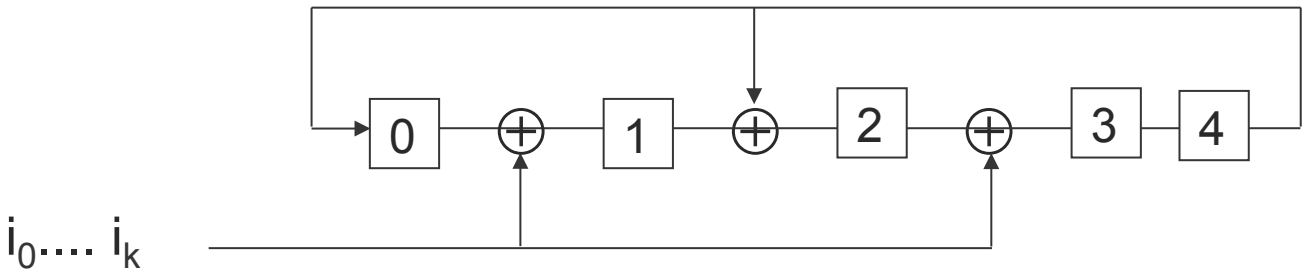
(1 0 1 0 1) + (0 1 0 1 0) = (1 1 1 1 1)

Codes cycliques : en résumé

- Codage :
  - par multiplication :
    - $c(x) = i(x).g(x)$
  - par division (si code systématique)
    - $c(x) = x^{n-k}.i(x) + R(x)$ avec $R(x)$= Reste $(x^{n-k}.i(x) / g(x))$
- Décodage :
  - par division
    - $g(x)$
    - x^n+1

Implantation : multiplication / division

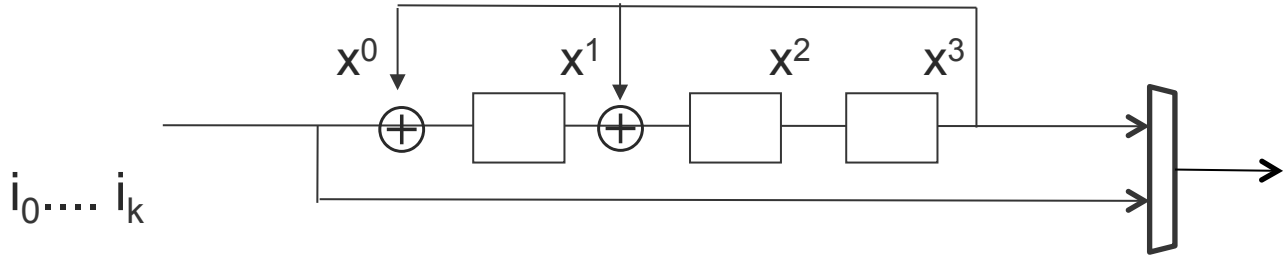
Multiplication par $x+x^3$, le résultat est divisé par $1+x^2+x^5$



Exemple de codeur pour codage systématique

Rappel : $c(x) = x^{n-k}.i(x) + R(x)$ avec $R(x)$= Reste $(x^{n-k}.i(x) / g(x))$

Exemple : C(7,4) avec $g(x) = x^3+x+1$



Pendant 4 cycles, voie du bas du mux :

Sur la sortie, on a les bits d'information.

En même temps on calcule le reste de la division par g(x).

Pendant les 3 cycles suivants, voie du haut du mux :

On ressort le reste

Exemples de codes cycliques

- **Exemples de polynômes générateurs**

  - Code **BCH** (Bose-Chaudhuri - Hocquenghem)

    $g(x) = (1 + x + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2)$ → l=15, m=10, e=3
    R = 33%

  - Code **Golay**

    $g(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$ → l=23, m=11, e=3
    R = 52 %

  -X25 (HDLC) (ex: RNIS)

  $x^{16}+x^{12}+x^5+1$

Erreurs unidirectionnelles

- Pour l'instant on a supposé que les "1" pouvaient devenir des "0" et les "0" des "1".
- Définition : une erreur est dite *unidirectionnelle* si les "1" sont changés en "0" (ou les "0" en "1") mais pas les deux à la fois
  - 0 1 0 1 1 -> 0 0 0 1 0 est unidirectionnelle
  - 0 1 0 1 1 -> 1 0 0 0 1 n'est pas unidirectionnelle
- Soient X et Y deux k-uplets. On note N(X,Y) les nombre de changement 1->0 de X à Y.
  - X = 1 0 1 0 1 0 et Y = 1 1 0 1 0 1, N(X,Y) =2 et N(Y,X)=3
  - N(X,Y)+N(Y,X) = distance de Hamming entre X et Y
- Définition : X *couvre* Y (noté X≥Y) si yi=1 implique xi=1, pour tout i, càd les positions des 1 dans Y est un sous-ensemble des 1 dans X. Si X couvre Y alors N(X,Y)=0.
  - si X = 1 0 1 0 1 0 et Y = 1 0 1 0 0 1 alors X≥Y

Erreurs unidirectionnelles et codes non ordonnés

- X≥Y : relation d'ordre partiel
- Définition : un code est dit non ordonné si chaque mot de code ne couvre aucun autre mode du code
- Propriété : un code non ordonné détecte toutes les erreurs unidirectionnelles.
  - En effet une erreur unidirectionnelle ne peut transformer un mot du code en un autre mot du code.
- Codes non ordonnés : non séparables et séparables
- Par exemple les codes m-parmi-n sont non ordonnés et non séparables alors que les codes de Berger sont non ordonnés et séparables. Ces deux codes détectent les erreurs de multiplicité 1 et les erreurs unidirectionnelles multiples.

Codes m-parmi-n (non ordonnés non séparables)

- Codes m-parmi-n : tous les mots du code ont exactement m "1" et n-m "0".
- Le nombre total de mots de code valides est C^m_n

  ex : 2-parmi-4 a 6 mots de codes valides :

  0011, 0101, 1001, 0110, 1010, 1100
- Pour m=k et n=2k, un cas particulier est le code "k-paire double-rail" : chaque mot de code contient k bits d'information et k bits de contrôle qui sont les compléments bit à bit des bits d'information.
  - code 2-paire double-rail : 0011,1001,0110,1100

Codes de Berger (séparable, non ordonné)

- Un codes de Berger de longueur n a k bits d'information et c=⌈log2(k+1)⌉ bits de contrôle et n=k+c
- C'est le code le moins redondant pour détecter les erreurs simples et les erreurs unidirectionnelles multiples.
- Un mot de code est construit en comptant le nombre de bits à 1 dans la partie information et en ajoutant les complément bit à bit de ce nombre comme bits de contrôle.
  - ex : k = 0101000, c= ⌈log2(7+1)⌉=3 bits de contrôle
  - nombre de "1"= 2, codé sur 3 bits : 010 , son complément 101, donc le mot de code est : 0101000 101
  - 101 est aussi le nombre de 0 dans 0101000

Codes détecteurs d'erreurs t-directionnelles

- Définition : Un erreur directionnelle est dite t-directionnelle si le nombre de bits erroné est inférieur ou égal à t.
- Codes de Borden : les mots de code de longueur n ont un poids w (le nombre de bits à 1) qui est congruent à $\lfloor n/2 \rfloor \bmod(t+1)$
- Le nombre de mots de codes est égal à $\sum_{w \equiv \lfloor n/2 \rfloor \bmod(t+1)} C_n^w$
- Par exemple pour n=12 et t=3. Le nombre de mots de code est

$$\sum_{w \equiv \lfloor 12/2 \rfloor \bmod(3+1)} C_{12}^w = \sum_{w \equiv 6 \bmod(4)} C_{12}^w = \sum_{w \equiv 2 \bmod(4)} C_{12}^w = C_{12}^2 + C_{12}^6 + C_{12}^{10} = 1056$$

- Lorsque t = n/2, un code de Borden a un poids constant, il détecte toutes les erreurs simples et les erreurs unidirectionnelles. C'est le code optimal au sens de la redondance pour ce type de détection.

Erreurs t-directionnelles et Codes de Bose-Lin

- Codes de Bose-Lin pour détecter des erreurs unidirectionnelles de multiplicité 2,3 et 6 en n'utilisant que resp. 2,3 et 4 bits de contrôle.
- Les bits de contrôle pour les erreurs 2 et 3 directionnelles sont la valeur de k_0 mod 4 et k_0 mod 8 resp. où k_0 est le nombre de 0 dans les bits d'information.
- Pour les erreurs 6-unidirectionnelles, les bits de contrôle sont la valeur de ($k_0$mod8) +4
- Par exemple les bits de contrôle du mot 1 0 1 1 0 1 0 1 1 1 0 0 0 1 1 1 sont
  - 1 0 (=2) pour les erreurs 2-directionnelles
  - 1 1 0 (=6) pour les erreurs 3-directionnelles
  - 1 0 1 0 (=10) pour les erreurs 3-directionnelles

Codes convolutifs

- **Généralités**
  - Les symboles sont traités de manière continue (flux)

- k symboles d'informations en entrée et n symboles de sorties et n=x.k

- Les symboles de sorties dépendent des informations d'entrées précédentes

$$\rho = \frac{k}{n}$$

Codes convolutifs

- Radio numérique
- Téléphones mobiles
- Bluetooth
- Comm. Satellites

Codes convolutifs

- **Généralités**
  - Les symboles sont traités de manière continue (flux)

- k symboles d'informations en entrée et n symboles de sorties et n=x.k

- Les symboles de sorties dépendent des informations d'entrées précédentes

$$\rho = \frac{k}{n}$$

Codes convolutifs

- **Codes convolutifs systématiques**
  - Mot de code : $C = [I_1 Y_1 I_2 Y_2 \ldots\ldots I_j Y_j \ldots\ldots]$

    avec $I_j = [I_j^1 \ldots\ldots I_j^{k_0}]$ Information

    $Y_j = [Y_j^1 \ldots\ldots Y_j^{m_0}]$ Contrôle

- **Codes convolutifs non systématiques**
  - Mot de code : $C = [U_1 U_2 \ldots\ldots U_j \ldots\ldots]$

Codes convolutifs

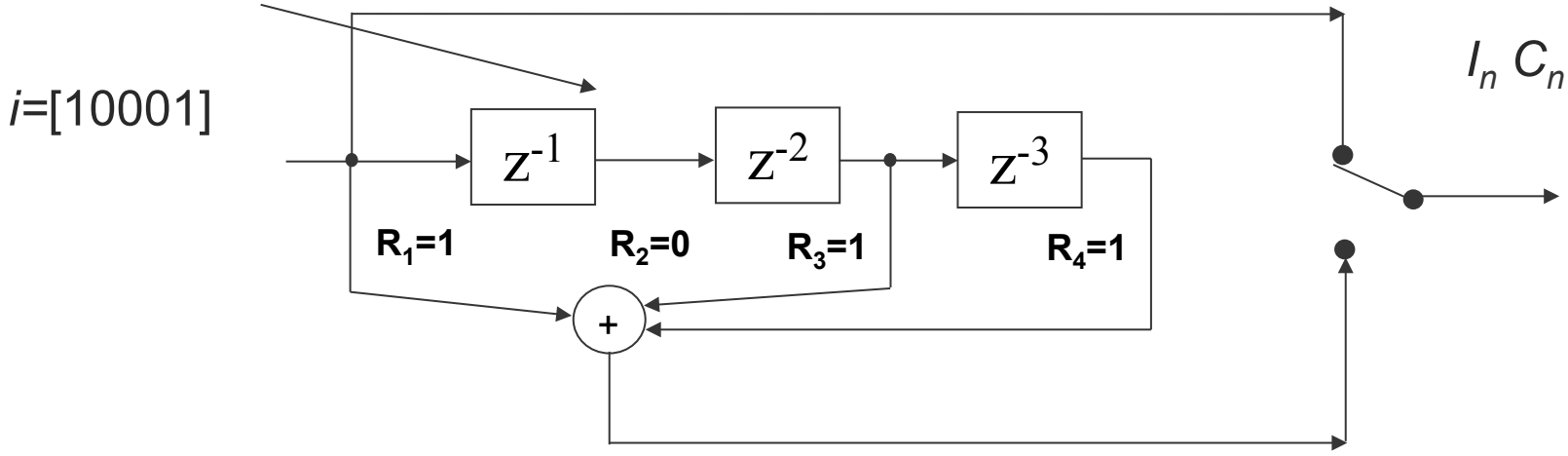
- **Codes convolutifs systématiques :**

- Exemple : contrainte l=4, m=1, n=2

$$c_n = R4.i_{n-3} + R3.i_{n-2} + R2.i_{n-1} + R1.i_n$$

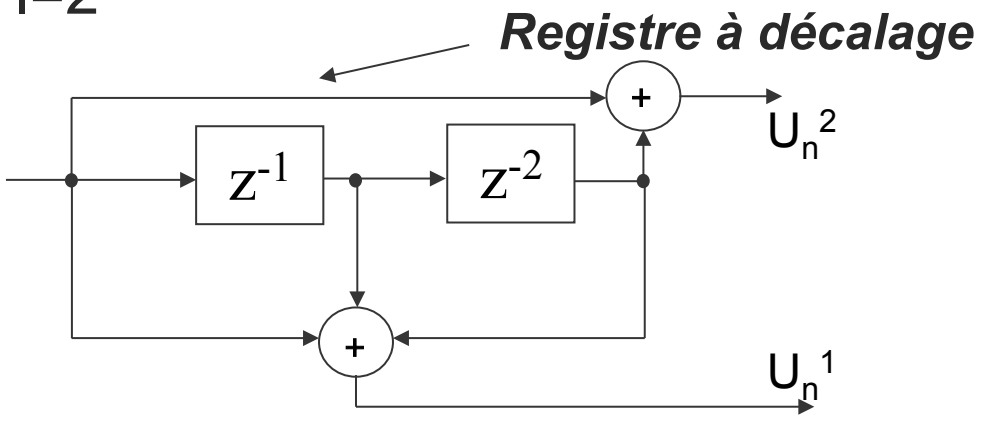
Avec ***R***=[1011]

Registre à décalage



Codes convolutifs

- **Codes convolutif non systématiques**
  - Exemple : m=3, k=1, l=2



$$U_n^2 = i_n \oplus i_{n-2}$$

$$U_n^1 = i_n \oplus i_{n-1} \oplus i_{n-2}$$

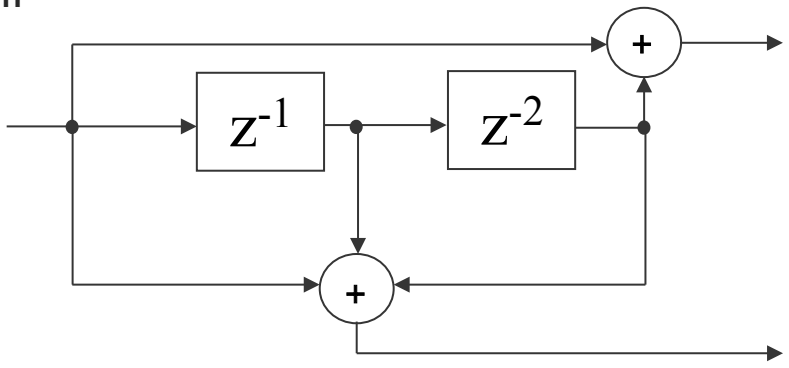
$$R = \frac{1}{2}$$

$$U_n^i = \sum_{j=0}^{2} g_{ij} . i_{n-j}$$

Avec $g_2=[1\ 0\ 1]=5$ et $g_1=[1\ 1\ 1]=7$ → Code 7,5

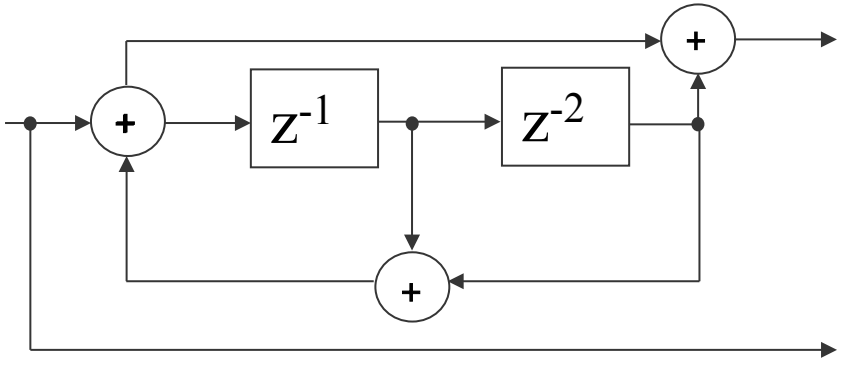
Codes convolutifs : non-récursifs, récursifs

Non-récursif



En général, non systématique

Récursif



En général, systématique

Codes convolutifs (5)

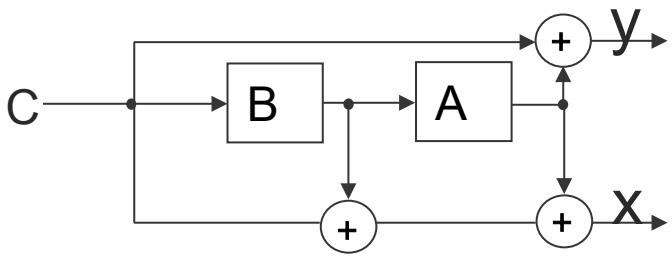
- **Codes convolutif non systématiques : Machine d'états**

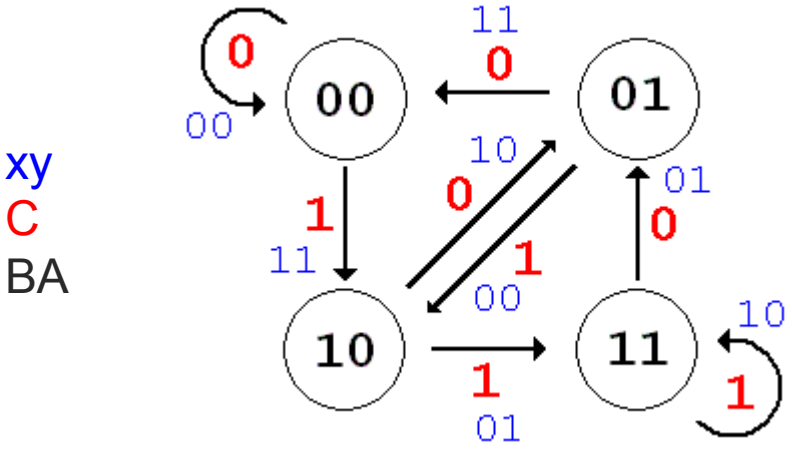
$$U_n^2 = i_n \oplus i_{n-2}$$

$$U_n^1 = i_n \oplus i_{n-1} \oplus i_{n-2}$$

i_n	i_{n-1} i_{n-2}	n	n+1	$U^1{}_n$ $U^2{}_n$
0 1	0 0	**A**	**A** **C**	0 0 1 1
0 1	0 1	**B**	**A** **C**	1 1 0 0
0 1	1 0	**C**	**B** **D**	1 0 0 1
0 1	1 1	**D**	**B** **D**	0 1 1 0

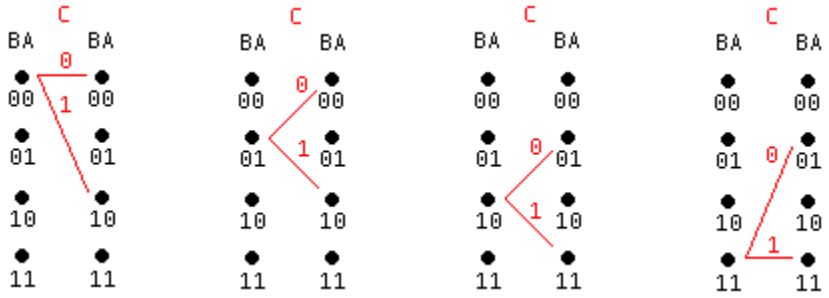
Analyse séquentielle





Remarque :
pas de changement de 00 à 11
changements sur un seul bit

Analyse séquentielle : treillis



Présent ----------> Futur

BA -C--> CBA=(XY) next BA

---

00 -0--> 000=(00) ===> 00

00 -1--> 100=(11) ===> 10

---

01 -0--> 001=(11) ===> 00

01 -1--> 101=(00) ===> 10

---

10 -0--> 010=(10) ===> 01

10 -1--> 110=(01) ===> 11

---

11 -0--> 011=(01) ===> 01

11 -1--> 111=(10) ===> 11

Exemple de codage

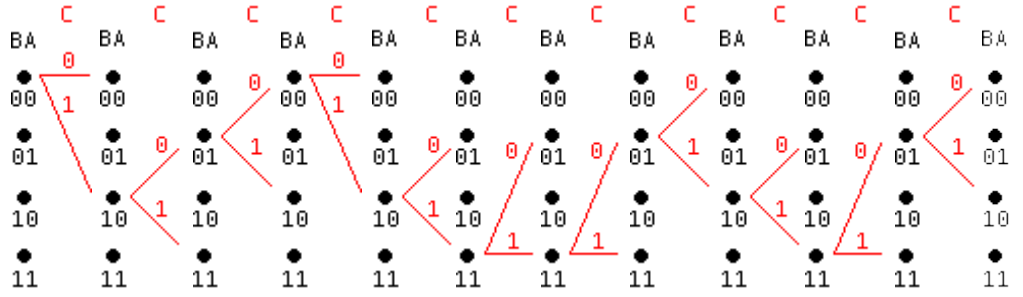
à transmettre : 100111011

BA -C--> CBA=(XY) next BA

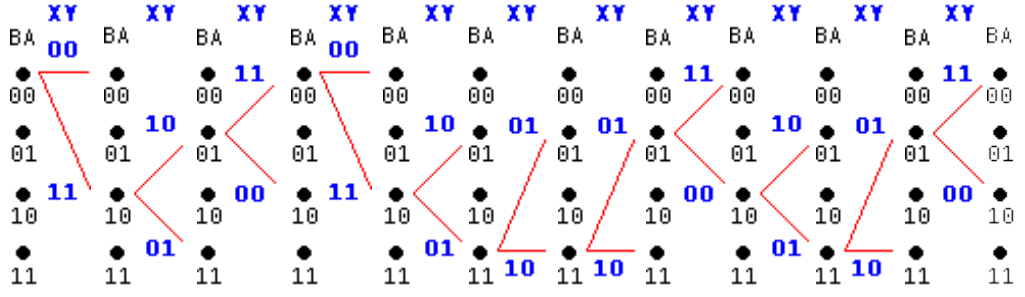
00 -1--> 100=(11) ===> 10
10 -0--> 010=(10) ===> 01
01 -0--> 001=(11) ===> 00
00 -1--> 100=(11) ===> 10
10 -1--> 110=(01) ===> 11
11 -1--> 111=(10) ===> 11
11 -0--> 011=(01) ===> 01
01 -1--> 101=(00) ===> 10
10 -1--> 110=(01) ===> 11
--- vidange avec zeros à la fin
11 -0--> 011=(01) ===> 01
01 -0--> 001=(11) ===> 00

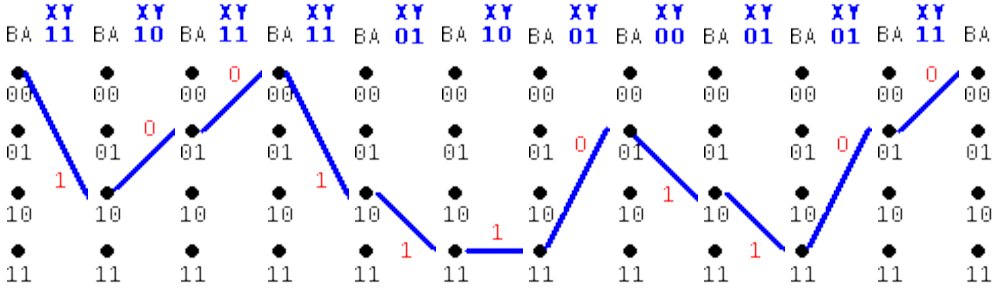
transmis : 11 10 11 11 01 10 01 00 01 01 11

Exemple de codage 2



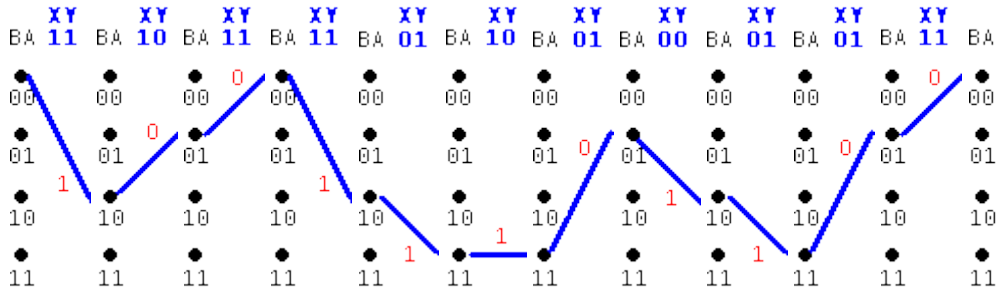
Bits d'entrée 1 0 1 1 1 0 1 1 0 0 0





Bits de sortie 11 10 11 11 01 10 01 00 01 01 11

Codage



Bits de sortie 11 10 11 11 01 10 01 00 01 01 11

A la réception, on retrace le chemin.
La correction est possible puisqu'une erreur introduit une déviation dans ce chemin

Décodage

Si pas d'erreur 11 10 11 11 01 10 01 00 01 01 11

BA -C--> CBA=(XY) next BA

---

00 -0--> 000=(00) ===> 00

00 -1--> 100=(11) ===> 10

---

01 -0--> 001=(11) ===> 00

01 -1--> 101=(00) ===> 10

---

10 -0--> 010=(10) ===> 01

10 -1--> 110=(01) ===> 11

---

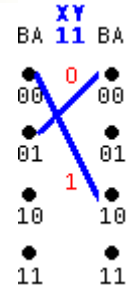
11 -0--> 011=(01) ===> 01

11 -1--> 111=(10) ===> 11

L'état initial étant 00, les seuls états suivants possibles sont 00 et 01.
Comme on a reçu 11=> l'état suivant est 10, et l'entrée est 1
Les états suivants de 10 sont 01 et 11
Comme on a reçu 10 => l'état suivant est 01, et l'entrée est 0
etc…

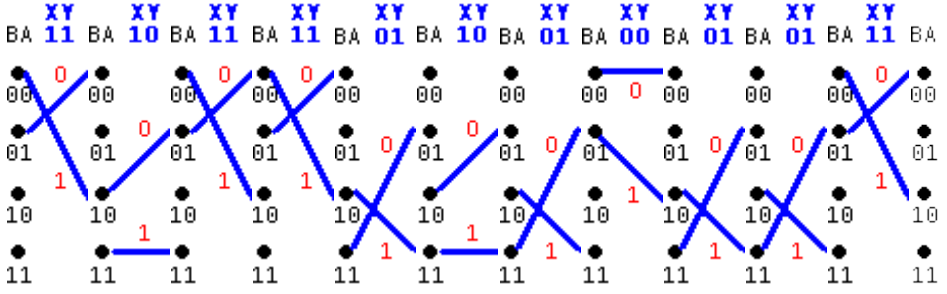
=> Le message envoyé est 1 0 1 1 1 0 1 1 0 0 0

Décodage : par treillis

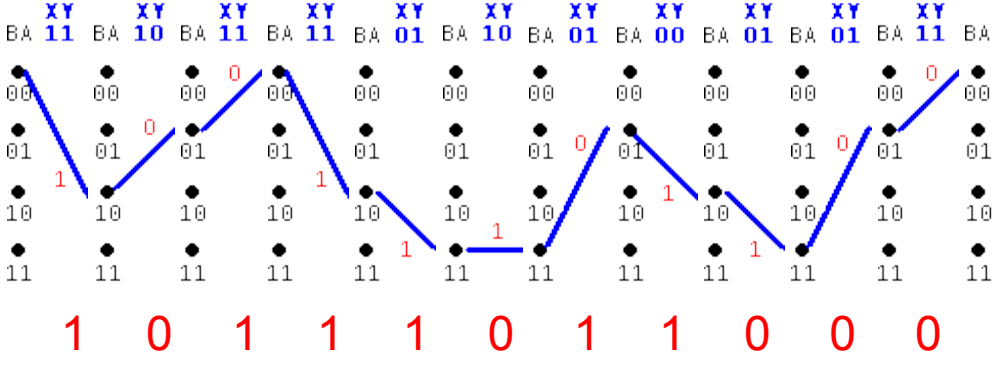


Le premier 11 peut représenter soit un 1 en allant de l'état 00 à 10,
soit un 0 en allant de 01 à 00

Pour un symbole reçu donné, la valeur du bit d'origine dépend de l'état courant

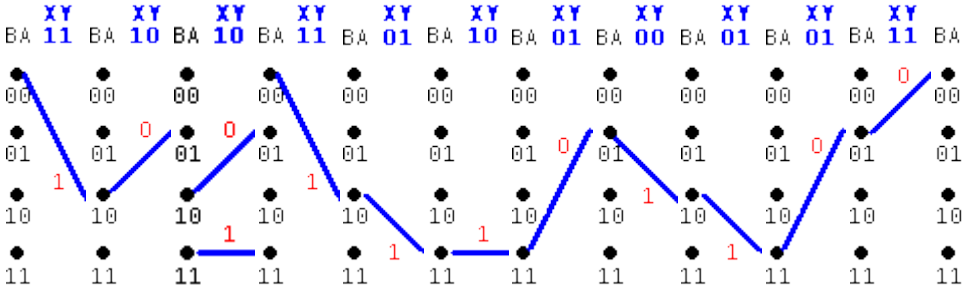


Le chemin est continu si il n'y a pas eu d'erreurs



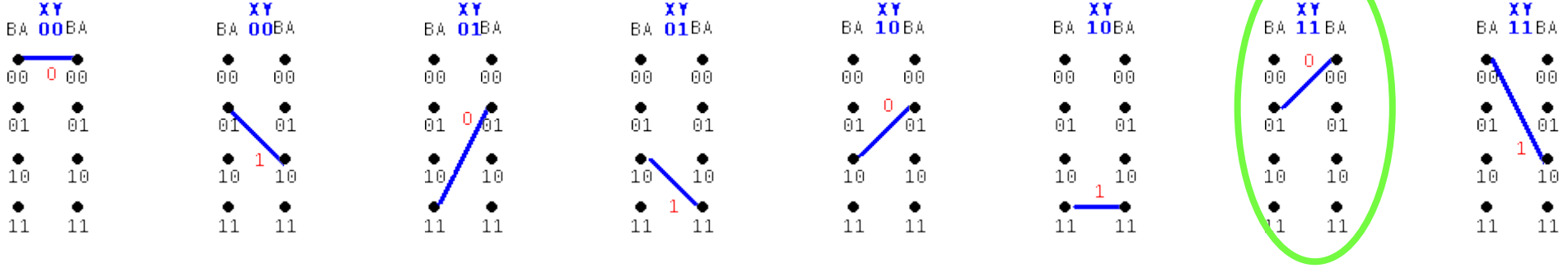
Exemple : une seule erreur

11 10 10 11 01 10 01 00 01 01 11



1 0 ? 1 1 0 1 1 0 0 0

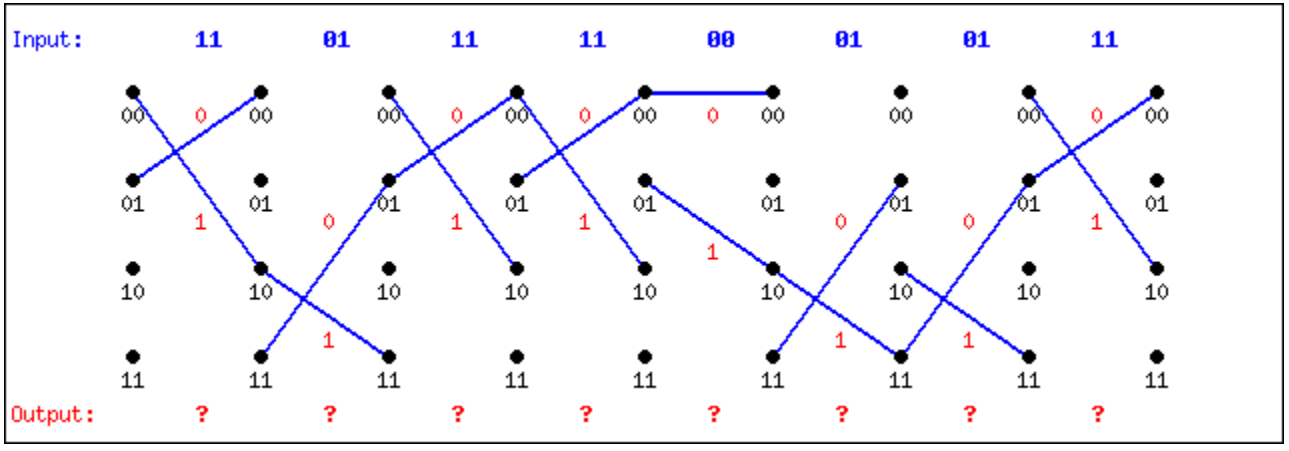
Il faut reconstruire un chemin continu. On cherche celui qui "colle" le mieux.
4 symboles possibles et 8 transitions.



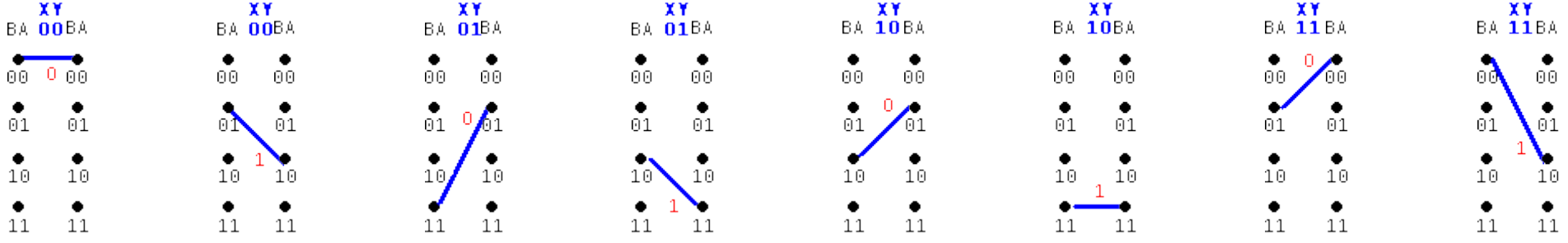
Données reçues	11	10	10	11	01	10	01	00	01	01	11
Données corrigées	11	10	11	11	01	10	01	00	01	01	11
Différence	00	00	01	00	00	00	00	00	00	00	00

Algorithme de Viterbi

11 01 11 11 00 01 01 11



Pas de chemin évident. On cherche le chemin continu le plus probable



Viterbi

Le chemin le plus probable est celui qui est à distance minimale des symboles reçus.
On suppose que chaque bit différent peut être corrigé

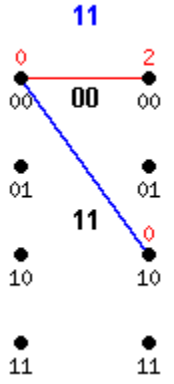
A chaque instant, le circuit de codage ne peut avoir que 4 états.
Caque état ne peut conduire qu'à 2 autres états à l'instant suivant
Inversement, on ne peut arriver dans un état qu'à partir de 2 états.

Autrement dit, 2 chemins exactement conduisent à chacun des 4 états.

En général, un de ces 2 chemins aura la plus petite erreur jusqu'à ce point dans le treillis.
En choisissant le 'meilleur chemin jusque là' et en passant à l'état suivant ,
le meilleur chemin complet est celui avec la plus petite erreur totale.

Exemple 11 01 11 11 00 01 01 11

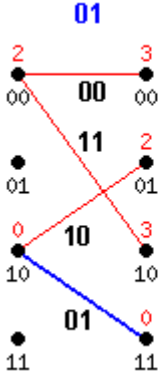
Etape 1



Si ce chemin est suivi , erreur =2

Si ce chemin est suivi , erreur =0

Etape 2



Si ce chemin est suivi , erreur = 2 + 1 =3

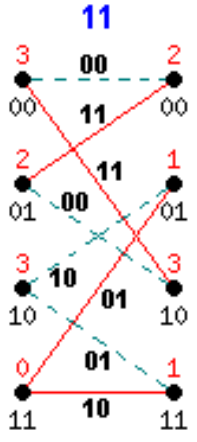
Si ce chemin est suivi , erreur = 0 + 2 =2

Si ce chemin est suivi , erreur = 2 + 1 =3

Si ce chemin est suivi , erreur = 0 + 0 =0

Exemple 11 01 11 11 00 01 01 11

Etape 3



Si ce chemin est suivi , erreur = 2 + 0 =2
Si ce chemin est suivi , erreur = 3 + 2 =5 X

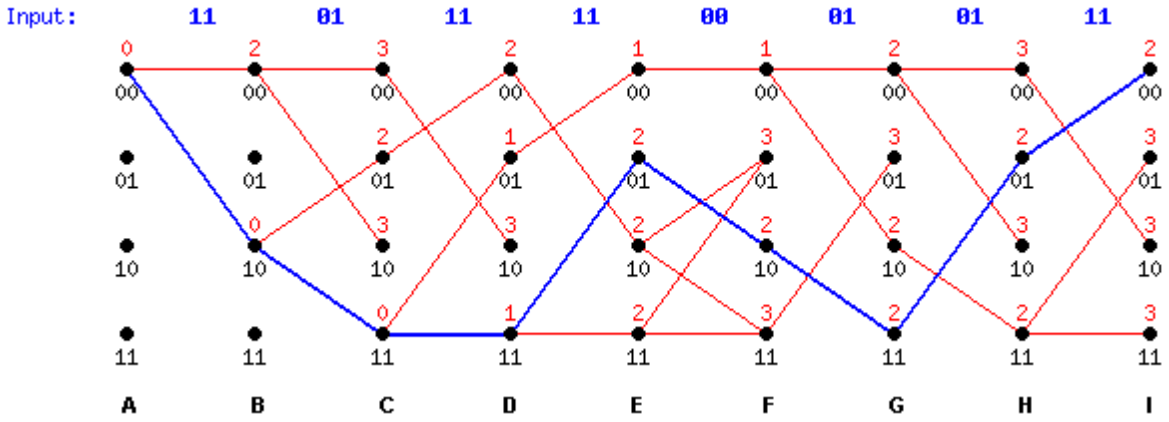
Si ce chemin est suivi , erreur = 0 + 1 =1
Si ce chemin est suivi , erreur = 3 + 1 =4 X

Si ce chemin est suivi , erreur = 3 + 0 =3
Si ce chemin est suivi , erreur = 2 + 2 =4 X

Si ce chemin est suivi , erreur = 0 + 1 =1
Si ce chemin est suivi , erreur = 3 + 1 =4 X

Exemple

Etc…



Turbo-codes

- **Origine**

  Les turbo codes sont nés au sein de l'École nationale supérieure des télécommunications de Bretagne (ENST Bretagne), suite aux travaux de Claude Berrou, Alain Glavieux et *Punya Thitimajshima*, publiés en 1993 et présentés lors de l'*International Conference on Communications* de juin 1993 à Genève en Suisse.

- **Principe des turbo codes**

  Le principe des turbo codes, comme tout code correcteur d'erreur, est d'introduire une redondance dans le message afin de le rendre moins sensible aux bruits et perturbations subies lors de la transmission. Le décodage, lui, est une collaboration entre deux décodeurs, d'où l'appellation *turbo*.

- **Codage**

  Un turbo codeur classique résulte de l'association de deux codeurs. Il s'agit souvent de codeurs convolutifs récursifs systématiques (*RSC : Recursive Systematic Coder*) car leur récursivité apportent des propriétés pseudo-aléatoires intéressantes.

  Concrètement, le turbo codeur produira typiquement trois sorties à envoyer sur le canal de transmission (après modulation éventuelle) :

  la sortie *ys* dite systématique, c'est-à-dire l'entrée même du codeur (la séquence *u*)

  la sortie de parité 1 : la sortie du premier codeur

  la sortie de parité 2 : la sortie du deuxième codeur. La différence entre ces deux sorties vient du fait que la trame *u* est entrelacée avant d'entrer dans le deuxième codeur. Elle est *mélangée*.

  Ce codage permet donc de répartir l'information apporté par un bit de la trame *u* sur ses voisins (avec le codage de parité 1) et même sur toute la longueur de la trame transmise (avec le codage de parité 2). Ainsi, si une partie du message est fortement abîmée pendant la transmission, l'information peut encore se retrouver ailleurs.

- **Décodage**

  Le décodage, nous l'avons dit, est le fruit de la collaboration de deux décodeurs. Ceux-ci vont s'échanger de l'information de manière itérative afin d'améliorer la fiabilité de la décision qui sera prise pour chaque symbole.

- **Utilisation de turbo code**

  En raison de sa performance, turbo code a été adopté par plusieurs organismes pour être intégré dans leurs standards. C'est ainsi que la NASA a décidé d'utiliser les turbo codes pour toutes ses sondes spatiales, construites à partir de 2003.

  De son côté, l'Agence spatiale européenne (ESA) a été la première agence spatiale à utiliser turbo code, avec sa sonde lunaire Smart 1.

  Plus près de nous, turbo code est utilisé par l'UMTS et l'ADSL 2.

Codes non ordonnés et erreur unidirectionnelles

- Une erreur est unidirectionnelle ssi elle ne fait que des changements de 0 vers 1 (ou 1 vers 0) mais pas les deux à la fois
- Un code non ordonné est capable de détecter toutes les erreurs unidirectionnelles (qque soit leur multiplicité).
- Cest parce que dans un tel code un erreur unidirectionnelle ne peut pas transformer un mot de code en un autre mot de code
- Les codes non ordonnés peuvent être séparables ou non.
- Par exemple : les codes m parmi n et les codes de Berger sont non ordonnnés

Code m parmi n

- Code m parmi n: tous les codes ont exactement m bits à 1 et (n-m) à 0. Le nombre total de codes valides est C_n^m
- Rem : ce n'est pas un code linéaire puisque le code de 0..0 n'est pas 0…0
- Ex : (0011), (0101),(1001),(0110),(1010),(1100) six codes valides et une erreur unidirectionnelle ne peut pas trnsformer un mot de code en un autre mot de code
- Si m=k et n=2k est connu sous le terme code k parmi 2k. Le cas particulier d'un codage utilisant seulement 2^k codes possibles parmi les C_{2k}^k valeurs possibles est connu sous le terme de code k-paires double rail. Chaque mot de code a ainsi k bits d'info et k bits de vérif.

Code m parmi n

- Le code 2-paires double rail est (0011) (1001)(0110)(1100)
- Un cas particulier est le code 1 parmi n (appelé aussi one-hot) : un seul bit à 1 parmi les n

Codes de Berger

- Rappel : code séparable dont les bits de verif. sont la valeur binaire du nombre de bits à 1 du mot d'info (en fait son complément à 1)
- Les codes de Berger sont les codes les moins redondants pour détecter les erreurs simples et les erreurs unidirectionnelles
- Nombre de bits de verif. r : r = n-k= $\lceil Log2(k+1) \rceil$
- Ex : info sur 5 bits => 2^5 valeurs possibles,
- nombre de bits à 1= {0,1,2,3,4,5} => 6 valeurs possibles
- => il faut $\lceil Log2(6) \rceil$ = 3 bits pour coder les valeurs de 0 à 6
- Ex :

(0 1 0 0 1) : 2 bits à 1 -> (0 1 0 0 1 $\overline{0\ 1\ 0}$) = (0 1 0 0 1 1 0 1)

Codes de Berger

- Variantes

  On peut coder le nombre de 1 (code B1) ou le nombre de 0 (code B0)
- Montrer que les erreurs simples et les erreurs unidirectionnelles sont détectées.
- Si le nombre de bits d'info k = 2^r-1 le code est dit de longueur maximale
- Le code (0 1 0 1 0 0 0 1 0 1) est de longueur maximale puisque avec 3 bits on peut coder jusqu'à la valeur 7 (7 bits à 1) et le mot de code est de 7 bits
- Le code précédent (0 1 0 0 1 1 0 1) n'est pas de longueur maximale puisque on utilise 3 bits pour coder des valeurs qui vont de 0 jusqu'à 5 uniquement

Erreurs t-unidirectionelles

- En général les codes sont faits pour détecter des erreurs unidirectionnelles sur t parmi les n bits (càd de poids t et dites erreurs t-unidirectionnelles) plutôt que sur tous les n bits.
- Remarque :
  - Les erreurs unidirectionnelles ont un poids au maximum égal à n/2.

Erreurs t-unidirectionelles : code de Borden

- Codes de Borden :
- Les mots de code de longueur n ont un poids w qui est congru à $\lfloor n/2 \rfloor$ modulo(t+1)
- Le nombre de mots de code est $C(n,t) = \sum_{w \equiv \lfloor n/2 \rfloor \bmod(t+1)} C_n^w$

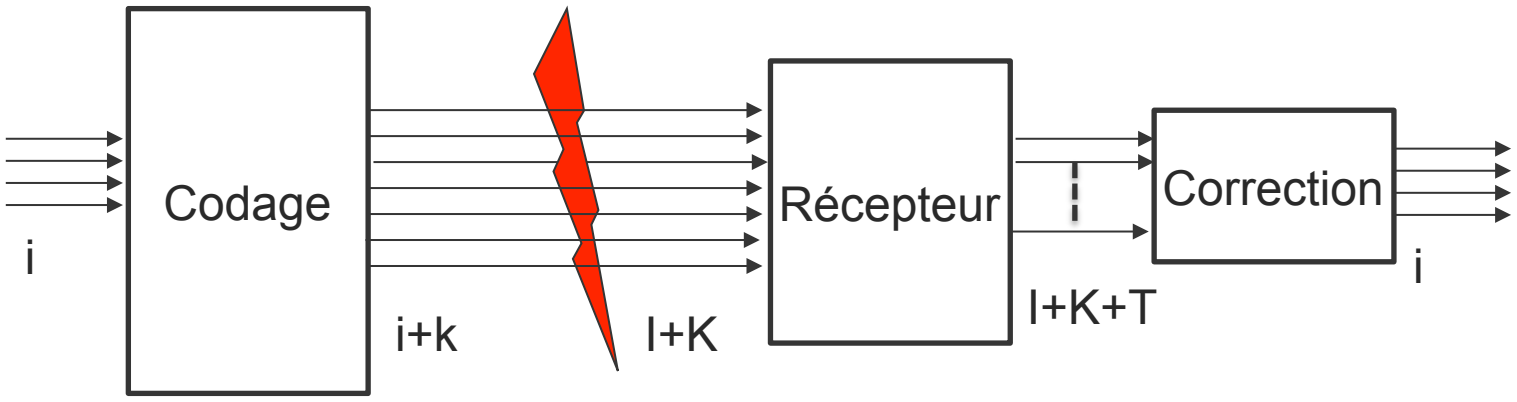
- Ex: Si n=12 et t=3 le nombre de mots de code est

$$C(12,3) = \sum_{w \equiv 6 \bmod(4)} C_{12}^w = \sum_{w \equiv 2 \bmod(4)} C_{12}^w = C_{12}^2 + C_{12}^6 + C_{12}^{10} = 1056$$

- Si t=n/2 , un code de Borden est un code m parmi 2m est détecte toutes les erreurs unidirectionnelles

Erreurs t-unidirectionelles : code de Borden

- Codes de Bose-Lin
  - Codes séparables capables de détecter des erreurs unidirectionnelles de poids 2, 3 et 6 en utilisant resp. 2, 3, et 4 bits de verif.
  - Ces codes sont optimaux
  - Pour les erreurs de poids 2 (resp.3) les bits de verif codent k0 mod 4 (resp. k0 mod 8) où k0 est le nombre de 0 dans le mot d'info.
  - Pour les erreurs 6-unidirectionnelles, les bits de verif. codent (k0 mod 8)+4.
  - Ex : pour le mot 1011010111000111 les bits de verif. sont resp. (10) = 2, (110) = 6 et (1010)=10.


Codage
Récepteur
Correction
i
i+k
I+K
I+K+T
i