TS219 CODAGE CANAL

Benoît ESCRIG
ENSEIRB-MATMECA/IRIT

Bibliographie

- J. PROAKIS : Digital Communications,
  - Ed. McGraw Hill, 2001.
- S. LIN, D.J. COSTELLO : Error Control Coding : Fundamentals and Applications,
  - Ed. Prentice-Hall, 1983.
- A. GLAVIEUX, M. JOINDOT : Communications numériques. Introduction,
  - Ed. Masson, 1996.

Plan du cours

1. Introduction
2. Codes en blocs linéaires
3. Codes convolutifs
4. Combinaisons de codes

Plan du cours

1. **Introduction**
   1. **Contexte**
   2. Principe
2. Codes en blocs linéaires
3. Codes convolutifs
4. Combinaisons de codes

Contextes d'application

DONNÉES ORIGINALES
(données émises/stockées) **0 1 0 1 0 0 1 1 0 0 0 1 1 1**

DONNÉES À TRAITER
(données reçues/lues) **0 1 0 0 0 0 1 1 1 0 0 1 1 1**

- Systèmes de télécommunications
  - Exemples : WiFi, TNT, Bluetooth, 3G, …
  - Causes de l'altération : mauvaises conditions de réception.
  - Résultat : les données reçues sont différentes des données émises.
- Stockage de données sur support physique
  - Stockage : écriture sur support physique (CD, disque dur).
  - Causes de l'altération : CD rayé, …
  - Résultat : les données lues sont différentes des données écrites.

Enjeux soulevés par l'altération des données

- L'équipement qui va utiliser les données (téléphone mobile, lecteur DVD) ne sait pas que le flux de données comporte des erreurs.
- L'impact de l'utilisation de données altérées dépend de l'application et du type d'altération.
  - Fort impact : un bit faux dans le codage d'un numéro de carte bleue .
  - Faible impact : plusieurs paquets manquants dans le codage d'une conversation téléphonique.

DONNÉES → BLOC UTILISATEUR DES DONNÉES

Enjeu

- Comment savoir si un flux de données comporte des erreurs ?
  - 1ère étape : détecter les erreurs.
- Que faire lorsque le flux de données à traiter est erroné ?
  - Corriger les erreurs (protocoles de la couche 1 et lecture de données sur support CD).
  - Solliciter l'émission de nouvelles données (protocoles à partir de la couche 2).
  - Interpoler entre les données précédentes et les données suivantes (applications).

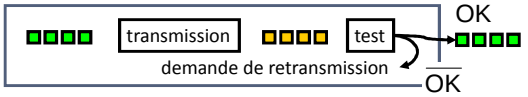
Solutions au dessus de la couche physique

- Couche application : interpoler entre les données précédentes et les données suivantes
  - Technique possible lorsque les données traitées sont très redondantes et que la perte d'un bloc de données n'altère pas la qualité du service.
  - Exemple : film, conversation téléphonique.
- Couches réseau (2,3,4) : solliciter l'émission de nouvelles données
  - Mécanisme ARQ (Automatic Repeat reQuest).

RÉCEPTION

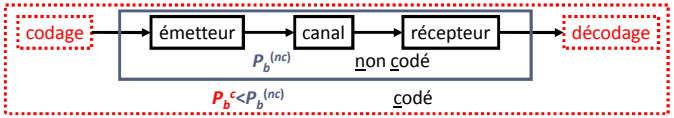
TRAITEMENT

vers la couche supérieure

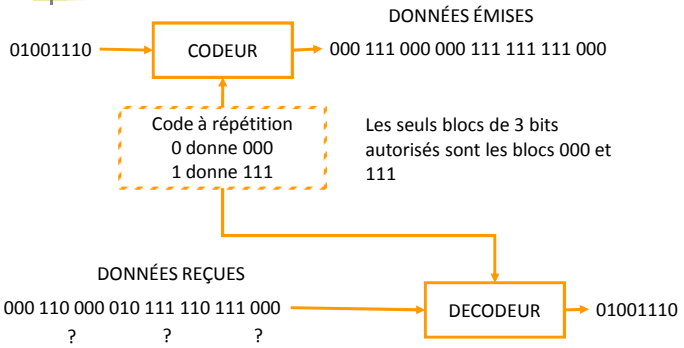


Détecter et corriger les données altérées

- **Codes correcteurs d'erreurs** : rajouter des bits de redondance aux bits d'information (opération inverse à l'opération de compression).
- Codage :
  - Blocs de *k* bits en entrée.
  - Blocs de *n* bits en sortie (*n*>*k*).
- Décodage : retrouver les blocs de *k* bits à partir des blocs de *n* bits.
- Application aux télécommunications : **codage canal**
  - Adapter le codage au canal de transmission pour améliorer les performances en termes de probabilité d'erreur binaire P_b.



Exemple : code à répétition C(3,1)



Limitation des codes correcteurs

- Si les erreurs sont rares, elles sont effectivement corrigées.

  000 110 000 010 111 110 111 000

- Sinon, le décodeur va générer des erreurs là où il n'y en avait pas.

  000 110 000 010 111 110 111 000

- Conséquence : un décodeur ne peut traiter correctement des données qu'à partir d'une certaine qualité à l'entrée du décodeur (aux environs de 1 bit faux sur 10 ou 1 bit faux sur 100).

Inconvénient des codes correcteurs d'erreurs

- En théorie, l'ajout de redondance contribue à augmenter les besoins en bande passante (T_{bloc}, temps d'émission d'un bloc de données).
- En pratique, le débit utile est diminué car le débit des données codées est fixé par la norme.
- Plus il y a de protection (et donc de redondance – *k*/*n* petit), plus le débit utile diminue.

	Cas non codé	Cas codé
Nombre de bits émis / bloc	*k*	*n*
Débit	k/T_{bloc}	n/T_{bloc}

	Données non codées	Données codées
Débit	*k*/*n* . *D*	*D* (fixe)

Enjeu des codes correcteurs d'erreurs

- Plus il y a de redondance,
  - Plus la capacité de correction est grande.
  - Plus la consommation en bande passante est grande.
- Enjeu : fournir la meilleure capacité de correction en utilisant le moins de bits de redondance.
- Intérêt pour les systèmes de télécommunications
  - Les techniques de transmission augmentent le rapport signal à bruit à la réception mais n'empêchent pas les erreurs.
  - Les codes correcteurs permettent alors d'améliorer un peu plus les performances en termes de BER (Bit Error Rate).
- Intérêt pour le stockage de données (disque dur, CD)
  - Récupération de données lorsque le support de stockage est défectueux.

Plan du cours

1. **Introduction**
   1. Contexte
   2. **Principe**
2. Codes en blocs linéaires
3. Codes convolutifs
4. Combinaisons de codes

Principe du codage

- Ajouter au message à transmettre des bits de redondance selon une loi donnée.
- Taux de codage R_c.
- Exemple : code à répétition C(3,1)
  - Si 0, alors 000. Si 1, alors 111.

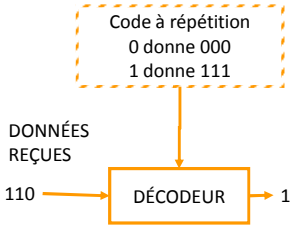
$$R_c = \frac{k}{n}$$

Principe du décodage

- Coder les données sans les décoder ne sert à rien.
- Décoder : tester si la loi de codage est respectée.
- Si la loi de codage est respectée, alors les données sont envoyées au bloc de traitement suivant.
- Exemple d'un code à répétition C(3,1) : si 000, alors 0. Si 111, alors 1.
- Si la loi de codage n'est pas respectée, il y a détection d'erreur(s).
- Exemple d'un code à répétition C(3,1) : il y a détection d'erreur dès lors que le bloc de trois bits reçus n'est ni 000, ni 111 (ex : 001).
- S'il y a détection d'erreur alors il y a correction et envoi au bloc de traitement suivant, ou suppression des données et demande de retransmission.

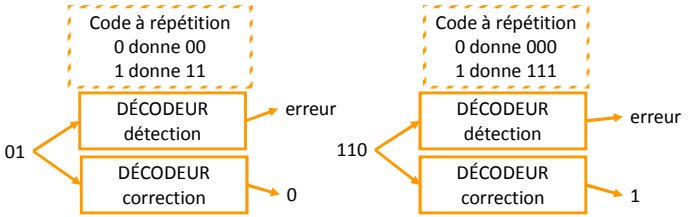
Exemple de correction d'erreur

- Détection d'erreur : le mot 110 est différent de 000 et de 111.
- Correction d'erreur : le mot reçu est plus proche de 111 que de 000. La sortie est donc 1.



Distinction entre détection et correction d'erreur

- Détection d'erreur : le décodeur détecte que le bloc de bits à traiter est erroné.
  - L'information produite est binaire : il y a ou il n'y a pas d'erreur (une ou plusieurs erreurs).
- Correction d'erreur : le décodeur sait où se trouvent les erreurs et les corrige.



Historique

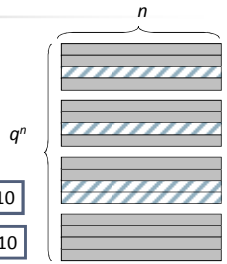
- 1948 : Shannon (théorie de l'information).
- 1950 : Hamming.
- 1950-1970 : codes en blocs et codes cycliques, BCH (Bose-Chaudhuri-Hocquenghem) et RS (Reed-Solomon).
- 1960-1970 : codes convolutifs (Fano, Forney, Viterbi).
- 1980 : modulations codées en treillis (Ungerboeck).
- 1990 : décodage itératif et turbo-codes (Berrou-Glavieux).
- 2000 : codes LDPC (Low Density Parity Check).

Plan du cours

1. Introduction
2. **Codes en blocs linéaires**
   1. Matrice génératrice et matrice de contrôle de parité
   2. Codes cycliques
   3. Décodage optimal soft-decision
   4. Décodage hard-decision
3. Codes convolutifs
4. Combinaisons de codes

Codes en blocs linéaires

- Code en bloc : ensemble de vecteurs de longueur n appelés mots de code.
- Les composantes d'un mot de code appartiennent à un alphabet à q symboles.
- Exemples :
  - Si q=2, alors les mots de code sont constitués de 0 et 1.
  - Si q =3, alors les mots de code sont constitués de 0,1 et 2.



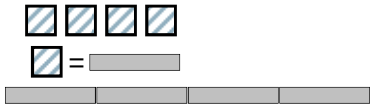
01010010

01210210

	code	composante	valeur
q=2	binaire	bit	{0,1}
q≠2	non binaire	symbole	{0,…,q-1}

Cas particulier : $q=2^b$ (b>1)

- Possibilité de représenter un symbole par b bits.
- Conséquence : un mot de code de N symboles peut se représenter par un mot de code binaire de bN bits.
- Exemple : si q = 4=2^2, alors mot de code constitué de 0,1,2 et 3.
  - Possibilité de représenter un symbole (0,1,2,3) par 2 bits (00 01 10 11).
  - Conséquence : mot de code de 8 symboles = mot de code binaire de 2x8 bits.



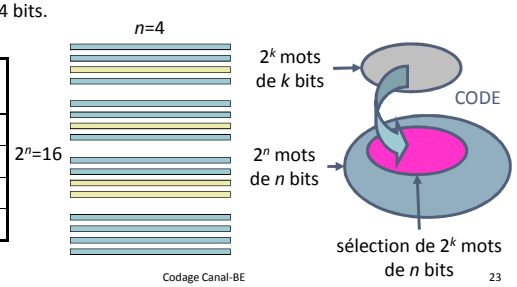
0 1 2 3 2 1 1 2

00 01 10 11 10 01 01 10

Définition d'un code C(n,k)

- Association de 2^k mots de code de n bits aux 2^k mots de données de k bits.
- Exemple C(4,2) : les 4 (2^2) mots de code sont choisis parmi les 16 (2^4) mots de 4 bits.

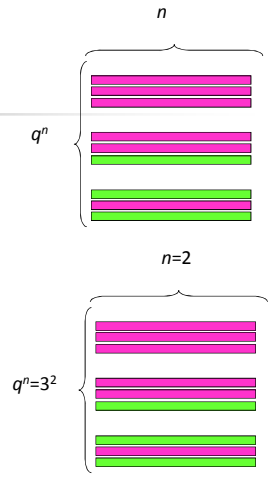
Mot de données	Mot de code
00	1010
01	0010
10	0110
11	1011



Généralisation aux codes non binaires

- Mots de code à n symboles (et non bits).
- Symboles : 0,1,2, …,q-1
- q^n mots de n symboles.
- Sélection de q^k mots pour former un code.
- Exemple C(2,1) q=3
  - 1 symbole en entrée : 0,1,2.
  - 2 symboles en sortie : (0,1,2) x (0,1,2).

Mot de données	Mot de code
0	12
1	20
2	22



Poids d'un mot : nombre d'éléments non nuls dans le mot

- Distribution des poids d'un code w : ensemble des poids d'un code.
- Existence de codes dont tous les mots de code ont le même poids (codes à poids fixe ou à poids constant).
- Paramètre important pour établir les performances de certains codes.
- Exemples : répartition possible des poids sur un code C(3,2).

$$000\ \ 001\ \ 011\ \ 111 \quad w = \{0,1,2,3\}$$
$$000\ \ 001\ \ 010\ \ 100 \quad w = \{0,1\}$$
$$110\ \ 101\ \ 011\ \ 111 \quad w = \{2,3\}$$

Opérations sur les mots de code et les mots de données

- Les composantes des vecteurs appartiennent à un alphabet à q symboles.
- Alphabet muni de l'addition modulo-q et de la multiplication modulo-q = corps de Galois d'ordre q, noté GF(q).
  - Corps de Galois = corps d'ordre fini.
  - Corps d'ordre fini = corps dont le nombre d'éléments (ordre) est fini.
- Exemple : GF(2) et GF(3).

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

×	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Distance minimale d_{min} d'un code

- Distance de Hamming : nombre d'éléments différents (bits, symboles) entre deux mots.
- Si les mots sont de taille n, la distance est comprise entre 0 et n.
- Distance de Hamming minimale entre deux mots de code : d_{min}.
- Exemple : C(3,2), d_{min}=2.

Mot de données	Mot de code
00	000
01	101
10	110
11	011

$$D([101],[110]) = 2$$

Plan du cours

1. Introduction
2. **Codes en blocs linéaires**
   1. **Matrice génératrice et matrice de contrôle de parité**
   2. Codes cycliques
   3. Décodage optimal soft-decision
   4. Décodage hard-decision
3. Codes convolutifs
4. Combinaisons de codes

Définitions et Notations

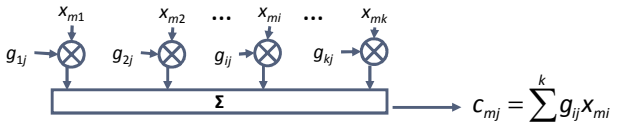
- Soit un code C(n,k).
- Mot de données $\mathbf{X}_m$ à k composantes.
- Mot de code $\mathbf{C}_m$ à n composantes.
- Pour construire un code C(n,k) valide, il faut que les 2^k n-uplets forment un sous-espace vectoriel de V_n, espace vectoriel des mots de n bits.

$$\mathbf{X}_m = [x_{m1} x_{m2} \ldots x_{mk}] \qquad \mathbf{C}_m = [c_{m1} c_{m2} \ldots c_{mk} c_{m(k+1)} \ldots c_{mn}]$$

$\mathbf{X}_m$ → CODEUR → $\mathbf{C}_m$

Génération de la $j^{\text{ième}}$ composante de $\mathbf{C}_m$

- Combinaison linéaire (CL) des k composantes du mot de données $\mathbf{X}_m$.
- Pour savoir quelles composantes de $\mathbf{X}_m$ participent à la CL : coefficients g_{ij} (0 ou 1).
- i de 1 à k pour les k composantes du mot de données
- j de 1 à n pour les n composantes du mot de code.
- m de 0 à 2^k-1 pour l'ensemble des mots de données.



$$c_{mj} = x_{m1} g_{1j} + x_{m2} g_{2j} + \cdots + x_{mk} g_{kj} \qquad j = 1,2,\cdots,n$$

Génération des n composantes de $\mathbf{C}_m$

- Pour n composantes, n combinaisons linéaires différentes.
- Le codeur est entièrement défini par n CLs.
- Matrice génératrice de rang k.
  - Rang d'une matrice : nombre maximal de vecteurs lignes (ou colonnes) linéairement indépendants.
- Les vecteurs $\mathbf{g}_i$ forment une base non unique.

$$\mathbf{C}_m = \mathbf{X}_m \mathbf{G}$$

$\mathbf{X}_m$ → CODEUR → $\mathbf{C}_m$

$$\mathbf{C}_m = x_{m1}\mathbf{g}_1 + x_{m2}\mathbf{g}_2 + \ldots + x_{mk}\mathbf{g}_k$$

$$1 \le i \le k$$

$$c_{mj} = \sum_{i=1}^{k} g_{ij} x_{mi} \qquad 1 \le j \le n$$

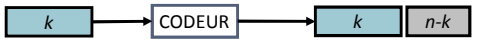
$$0 \le m \le 2^k - 1$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}$$

Matrice génératrice

Code systématique

- Un code est dit systématique si les k premiers bits du mot de code sont constitués par les k bits du mot de données.
- Ces k bits sont dits systématiques. Les (n-k) bits restants sont appelés bits de parité.
- Matrice génératrice pour code systématique : il est possible d'obtenir la matrice génératrice de la version systématique d'un code par opérations sur les lignes et permutations des colonnes.



Forme de la matrice génératrice pour un code systématique

- $\mathbf{I}_k$ matrice identité et **P** matrice k x $(n\text{-}k)$.

$$\mathbf{G} = \left[\mathbf{I}_k \middle| \mathbf{P}\right] = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{11} & p_{12} & \cdots & p_{1(n-k)} \\ 0 & 1 & \ddots & \vdots & p_{21} & p_{22} & \cdots & p_{2(n-k)} \\ \vdots & \ddots & \ddots & 0 & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & p_{k1} & p_{k2} & \cdots & p_{k(n-k)} \end{bmatrix}$$

Intérêt des codes systématiques

- Le mot de données apparaît explicitement dans le mot de code.
- Le décodage est plus facile car il suffit de tronquer le mot reçu.
- Exemple : le code de l'exemple précédent est systématique.

$$\mathbf{X}_m\left[\mathbf{I}_k \;\; \mathbf{P}\right] = \left[\mathbf{X}_m \;\; \mathbf{X}_m\mathbf{P}\right]$$

Bits systématiques — Bits de parité

Exemple : code C(3,2) $G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

Mot de données	Mot de code
00	000
01	011
10	101
11	110

Exercice : code C(7,4)

- Soit le code C(7,4) de matrice génératrice C(7,4).
- Donner les expressions des éléments du mot de code c_{mj}, où j=1,…,n, en fonction des éléments du mot de données x_{mi}, où i=1,…,k.

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

Code dual d'un code C(n,k)

- Soit un code C(n,k) de matrice génératrice **G**.
- Code dual : code C(n,n-k) de matrice génératrice **H**.
- Propriété : tous les mots de code générés par **G** sont orthogonaux à ceux générés par **H**.
- Forme particulière de la matrice **H** si le code généré par la matrice **G** est systématique.

$$\mathbf{G}\mathbf{H}^T = \mathbf{0}$$

$$\mathbf{C}_m\mathbf{H}^T = \mathbf{0}$$

$$\mathbf{G} = \left[\mathbf{I}_k \middle| \mathbf{P}\right]$$

$$\mathbf{H} = \left[-\mathbf{P}^T \middle| \mathbf{I}_{n-k}\right]$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Capacité de détection d'un code C(*n*,*k*)

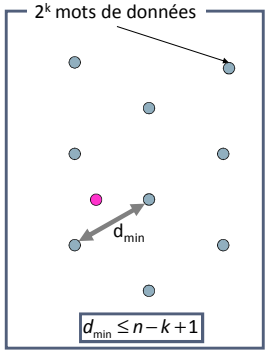
- Soit un code C(*n*,*k*) de distance minimale d_{min}.
- Capacité de détection d'un code : au plus d_{min}-1 erreurs.
- Détection : d_{min}-1 erreurs transforment un mot de code en un autre mot qui n'est pas un mot de code.

ÉVENTUELLEMENT

MOT DE CODE + $d \geq d_{min}$ ERREURS = MOT DE CODE

CERTAINEMENT

MOT DE CODE + $d < d_{min}$ ERREURS ≠ MOT DE CODE



Exemples

- C(3,2) :
  - d_{min}=2, capacité de détection : 1.
  - La réception de [010] implique au moins une erreur.
- C(5,1) :
  - d_{min}=5, capacité de détection : 4.
  - La réception de [00010] implique au moins une erreur.

Mot de données	Mot de code
00	000
01	011
10	101
11	110

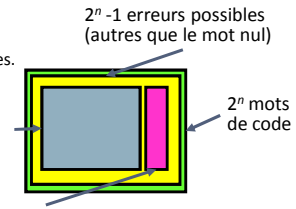
Mot de données	Mot de code
0	00000
1	11111

Calcul de la capacité de détection d'un code

- Code C(*n*,*k*) = 2^k mots de codes de longueur *n* parmi 2^n mots possibles.
- Lors d'une transmission, le mot de code $\mathbf{C}_m$ subit une altération **e** pour donner **Y**.
- Pour *n* grand, le nombre d'erreurs non détectables devient petit devant le nombre d'erreurs détectables.
- Exemple : C(7,4)
  - 16 mots de code, 15 erreurs non détectables.
  - 128-16=112 erreurs détectables

$\mathbf{Y} = \mathbf{C}_m + \mathbf{e}$

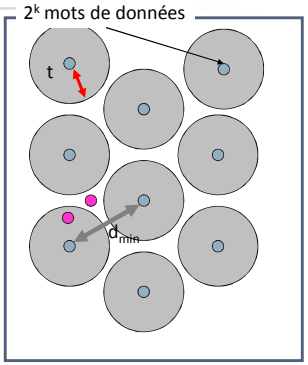
$$\frac{2^k - 1}{2^n - 2^k}$$



Capacité de correction d'un code C(*n*,*k*)

- Soit d_{min}, la distance minimale.
- Capacité de détection : d_{min} –1.
- Capacité de correction t : partie entière de la moitié de (d_{min}-1).
- Pour pouvoir corriger ces erreurs, il faut que le mot reçu soit suffisamment près d'un mot de code existant.

$$t = Ent\left[\frac{d_{min} - 1}{2}\right]$$



Exemples

- Code C(3,2), d_{min}=2 :
  - Réception de [010].
  - Détection d'une erreur.
  - Capacité de correction nulle.
  - Impossibilité de décider quel mot a été transmis : [000] ou [011] ou [110].
- Code à répétition C(3,1) :
  - Capacité de détection 2, capacité de correction 1.
  - Réception de [001] : détection d'une erreur et correction en [000].

Mot de données	Mot de code
00	000
01	101
10	110
11	011

Mot de données	Mot de code
0	000
1	111

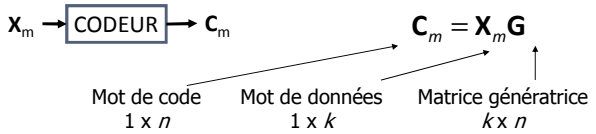
Limitation

- Exemple : code à répétition C(3,1).
- Transmission de [000] et réception de [011], détection d'une erreur et correction en [111].
- Conséquence :
  - en théorie, génération d'erreurs !
  - en pratique, cas correspondants à des conditions de transmission inexploitables.
- Les codes correcteurs ne peuvent fonctionner correctement que si le BER (Bit Error Rate) avant correction est inférieur à 10^{-1}.

Codes étendus et codes raccourcis

- Code étendu C(n+1,k) généré à partir d'un code C(n,k) : ajout d'un bit de parité (somme modulo 2 du mot de code) au mot de code généré par C(n,k) : 0 si nombre de 1 dans mot de code pair, 1 sinon.
- Propriété : si C(n,k) de distance minimale d_{min} impaire, alors C(n+1,k) de distance minimale d_{min}+1.
- Code raccourci C(n-l,k-l) généré à partir d'un code systématique de la forme C(n,k) : suppression des l premiers bits d'information (génération des mots de code par **G**) :
- Propriété : si le code C(n,k) est de distance minimale d_{min}, alors le code C(n-l,k-l) a au moins une distance minimale de d_{min}.
- Codes étendus et raccourcis utiles pour faire des multiples d'octets.

Conclusion sur les codes en blocs linéaires



- Limitations des codes en blocs linéaires
  - Stockage de la matrice (espace mémoire requis important),
  - Traitement temps réel impossible (traitement bloc).
- Alternative : codes cycliques.

De la forme [$\mathbf{I}_k$ **P**] pour les codes systématiques

Plan du cours

1. Introduction
2. **Codes en blocs linéaires**
   1. Matrice génératrice et matrice de contrôle de parité
   2. **Codes cycliques**
   3. Décodage optimal soft-decision
   4. Décodage hard-decision
3. Codes convolutifs
4. Combinaisons de codes

Codes cycliques

- Famille de codes linéaires ayant la propriété suivante : toute permutation circulaire d'un mot de code C donne un autre mot de code.
- Représentation polynomiale des codes cycliques
  - Associer à chaque mot de code, à n composantes, un polynôme C(p).
- Propriétés du polynôme :
  - Nombre de coefficients : n, de 0 à (n-1).
  - Degré inférieur ou égal à (n-1).
  - Codes binaires : coefficients dans {0,1}.

$\mathbf{C} = [c_{n-1} c_{n-2} \ldots c_1 c_0]$

$\mathbf{C}' = [c_{n-2} c_{n-3} \ldots c_0 c_{n-1}]$

$$\mathbf{C} = [c_{n-1} c_{n-2} \ldots c_1 c_0] \Rightarrow C(p) = c_{n-1} p^{n-1} + c_{n-2} p^{n-2} + \cdots + c_1 p + c_0$$

Exemples

$$\mathbf{C} = [1] \Rightarrow C(p) = 1$$
$$\mathbf{C} = [10] \Rightarrow C(p) = p$$
$$\mathbf{C} = [11] \Rightarrow C(p) = p + 1$$
$$\mathbf{C} = [1101] \Rightarrow C(p) = p^3 + p^2 + 1$$
$$\mathbf{C} = [1101101] \Rightarrow C(p) = p^6 + p^5 + p^3 + p^2 + 1$$

Génération de la permutation cyclique d'un mot de code

- Comment passer de **C** à $\mathbf{C}_1$? $\mathbf{C} = [c_{n-1} c_{n-2} \ldots c_1 c_0]$
- Idée 1 : $p\,C(p)$ mais $p\,C(p)$ ne peut pas être un mot de code car il est de degré n si $c_{n-1}=1$. $\mathbf{C}_1 = [c_{n-2} c_{n-3} \ldots c_0 c_{n-1}]$
- Idée 2 : division de $p\,C(p)$ par p^n+1

$$C(p) = c_{n-1} p^{n-1} + c_{n-2} p^{n-2} + \cdots + c_1 p + c_0$$
$$pC(p) = c_{n-1} p^n + c_{n-2} p^{n-1} + \cdots + c_1 p^2 + c_0 p$$
$$C_1(p) = c_{n-2} p^{n-1} + c_{n-3} p^{n-2} + \cdots + c_0 p + c_{n-1}$$
$$pC(p) = c_{n-1}(p^n + 1) + c_{n-2} p^{n-1} + \cdots + c_1 p^2 + c_0 p + c_{n-1}$$
$$pC(p) = c_{n-1}(p^n + 1) + C_1(p)$$

Génération des mots de code par permutation cyclique

- $\mathbf{C}_1$: reste de la division de $pC(p)$ par p^n+1.
- Généralisation :
  - Permutation de $\mathbf{C}_i$ de i éléments : reste de la division de $p^iC(p)$ par (p^n+1)

$$pC(p) = c_{n-1}\left(p^n+1\right) + C_1(p) \qquad \mathbf{C}_1 = \left[c_{n-2}c_{n-3}\ldots c_0 c_{n-1}\right]$$

$$p^iC(p) = Q(p)\left(p^n+1\right) + C_i(p) \qquad \mathbf{C}_i = \left[c_{n-i-1}c_{n-i-2}\ldots c_{n-i+1}c_{n-i}\right]$$

$$\mathbf{C} = \left[c_{n-1}c_{n-2}\ldots c_1 c_0\right]$$

Code cyclique et multiplication polynomiale

- Soit un polynôme générateur g(p) de degré (n-k), facteur de p^n+1 (g(p) divise p^n+1).
- Soit $X_m(p)$ le polynôme d'information à coder.
- Soit $C_m(p)$ le polynôme du mot de code.

$$g(p) = p^{n-k} + g_{n-k-1}p^{n-k-1} + \cdots + g_1 p + 1$$
$$X(p) = x_{k-1}p^{k-1} + x_{k-2}p^{k-2} + \cdots + x_1 p + x_0$$
$$C_m(p) = X_m(p)g(p)$$

$$p^7+1 = (p+1)\underbrace{\left(p^3+p^2+1\right)}_{g(p)}\underbrace{\left(p^3+p+1\right)}_{g_1(p)}$$

Exemple avec n=7 : avec p^7+1, possibilité de générer deux codes cycliques C(7,4).

Table de g(p)

- g(p)=p^3+p^2+1
- C(7,4)
- Exemples :
  - X(p)=1 donne C(p)=g(p)
  - X(p)=p donne C(p)=g(p).p
  - X(p)=p+1 donne C(p)=g(p)(p+1)=p^4+p^3+p+p^3+p^2+1=p^4+p^2+p+1

0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1
2	0	0	1	1	0	1	0
3	0	0	1	0	1	1	1
4	0	1	1	0	1	0	0
5	0	1	1	1	0	0	1
6	0	1	0	1	1	1	0
7	0	1	0	0	0	1	1
8	1	1	0	1	0	0	0
9	1	1	0	0	1	0	1
10	1	1	1	0	0	1	0
11	1	1	1	1	1	1	1
12	1	0	1	1	1	0	0
13	1	0	1	0	0	0	1
14	1	0	0	0	1	1	0
15	1	0	0	1	0	1	1

Lien avec les matrices génératrices

- La notion de code dual existe mais n'est pas utilisée.
- Matrice génératrice d'un code cyclique C(n,k) de polynôme g(p) constituée des lignes :
  - $p^{k-1}g(p)$
  - $p^{k-2}g(p)$
  - ...
  - $pg(p)$
  - $g(p)$

$$\mathbf{G} = \begin{bmatrix} 1 & g_{n-k-1} & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & & & & 0 \\ 0 & \cdots & 0 & 1 & g_{n-k-1} & \cdots & 1 \end{bmatrix}$$

Exemple : C(7,4)

$$g(p) = 1 + p^2 + p^3$$

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

Construction de codes systématiques

- Codes systématiques : *k* premiers bits constitués par les bits d'information.
- Méthode :
  - Multiplier le polynôme d'information X(*p*) par p^{n-k}.
  - Diviser p^{n-k}X(*p*) par *g*(*p*).
  - Ajouter le reste de la division, noté *r*(*p*), à p^{n-k}X(*p*).

$$p^{n-k}X(p) = Q(p)g(p) + r(p)$$
$$p^{n-k}X(p) + r(p) = Q(p)g(p)$$

Exemple : C(7,4) et $g(p) = p^3+p+1$

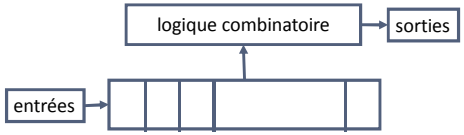
- Mot à coder **X**=[1001]
- X(*p*)=p^3+1
- p^{n-k}X(*p*)= p^6+p^3
- Le reste de la division de p^{n-k}X(*p*) par g*(p)* donne les bits de parité.
- **g**=[1011].

Mise en œuvre de codes cycliques

- Utilisation de registres à décalages.
- Registre : case mémoire accessible par le processeur sans temps d'accès (de taille 64 bits pour les processeurs dits 64 bits).
- Registre à décalage : registre de taille fixe dans lequel les bits sont décalés à chaque coup d'horloge.
- Intérêt des registres à décalages
  - Utiles pour le traitement de flux de donnés : peuvent traiter des flux ininterrompus de bits.
  - Utiles pour la mise en œuvre d'applications temps réel : un bit sortant pour un bit entrant.

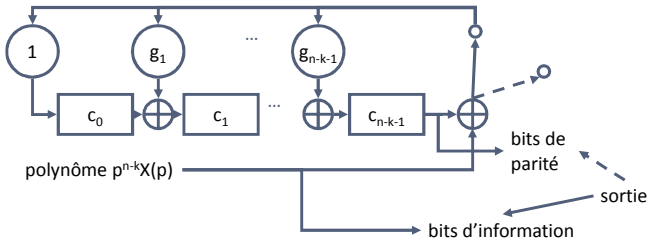
Étapes de codage

- Multiplier le polynôme d'information X(p) par p^{n-k} : décalage de registres.
- Diviser p^{n-k}X(p) par $g(p)$: seule opération non triviale à réaliser.
- Ajouter le reste de la division, noté $r(p)$, à p^{n-k}X(p) : remplir les cases du registre.



Codeur cyclique

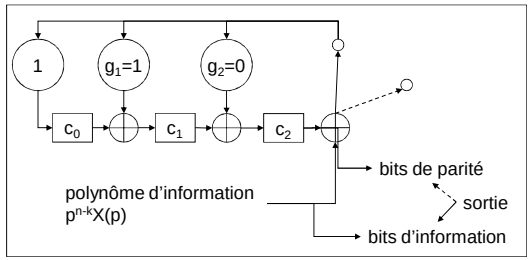
- Basculement des interrupteurs après passage des k bits d'information.



Exemple C(7,4)

$g(p) = 1 + p + p^3$

Entrée : 0110
Sortie : 0110001

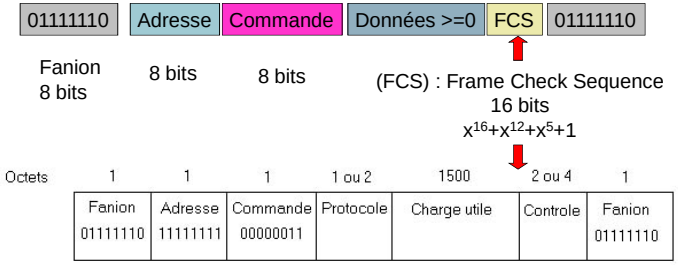


$c_0c_1c_2$
000
000
110
101
100

Codes cycliques dans les protocoles de communication

- Protection des unités de données de protocoles – PDUs (paquets, trames)
  - Les codes permettent de détecter des erreurs même si les champs de commande (en-tête) sont cohérents et conformes au protocole.
- Inconvénient : les codes cycliques définis avec n et k fixes alors que les PDUs sont de tailles variables.
- Implantation : utilisation d'un codeur C(n,k) systématique avec (n-k) bits de redondance.
- Quelle que soit la taille de la PDU, le nombre de bits de parité sera toujours le même : n-k.
- Exemples : paquet IP, trame Ethernet.
  - Avec un CRC de 16 bits, 16 bits de redondance seront produits, quelle que soit la taille du bloc à traiter.

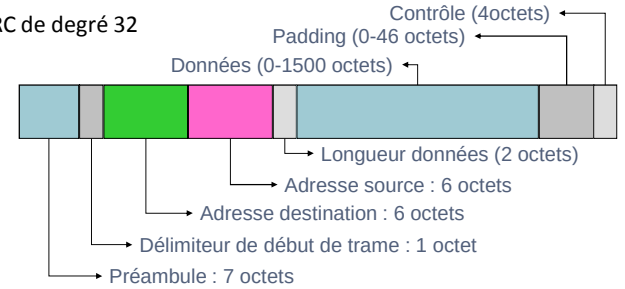
Exemples : trames HDLC et PPP



Exemple : trame Ethernet

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

- CRC de degré 32



Capacité de détection

- Si l'entrée est constituée de blocs de K bits, la sortie est constituée de blocs de $K+(n-k)$ bits.
- Le rapport entre le nombre d'erreurs non détectées et le nombre d'erreurs détectées tend vers $1/[2^{(n-k)}]$ lorsque K tend vers l'infini.

$$\frac{2^K-1}{2^{K+n-k}-2^K} = \frac{1-\frac{1}{2^K}}{2^{n-k}-1} \underset{K\to+\infty}{\longrightarrow} \frac{1}{2^{n-k}-1} \approx \frac{1}{2^{n-k}}$$

Codes BCH

- Limitation des codes cycliques C(n,k) : pour un code C(n,k) quelconque, il faut calculer toutes les distances entre les mots de code pour avoir d_{min} et en déduire les capacités de détection et de correction.
- Optimisation possible : définir des codes pour lesquels le d_{min} est connu a priori.
- Solution : codes BCH (Bose-Chaudhuri-Hocquenghem).
- Paramètres des codes BCH binaires
  - k bits d'information en entrée
  - n bits de code à la sortie
  - m entier supérieur ou égal à 3
  - t entier représentant le pouvoir de correction du code.

$$n = 2^m - 1$$
$$n - k \le mt$$
$$d_{min} = 2t + 1$$

- Propriété importante : les codes BCH portent dans la définition même de n et k, leur capacité de correction.

Représentation des polynômes générateurs des codes BCH

- Forme compacte d'énoncer les coefficients du polynôme : notation en octal
  - 1 chiffre = 3 bits
  - bit le plus à gauche = coefficient de degré le plus grand
- Tables existantes pour *m* compris entre 3 et 8, donc *n* compris entre 7 et 255 (Cf. Proakis).
- Exemple : C(15,5)
  - 2467 en octal = 010 100 110 111 en binaire

$$g(p) = p^{10} + p^8 + p^5 + p^4 + p^2 + p + 1$$

n	*k*	*t*	*g*(*p*)
7	4	1	13
15	11	1	23
	7	2	721
	5	3	2467
31	26	1	45
	21	2	3551 …
…			
255	247	1	435561

Conclusion sur les codes cycliques

$$C_m(p) = X_m(p)g(p)$$

Mot de code de degré <= *n* -1

Mot de données de degré <= *nk*-1

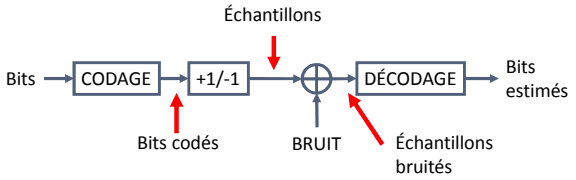
Polynôme générateur de degré <= *n* -*k*

Plan du cours

1. Introduction
2. **Codes en blocs linéaires**
   1. Matrice génératrice et matrice de contrôle de parité
   2. Codes cycliques
   3. **Décodage optimal soft-decision**
   4. Décodage hard-decision
3. Codes convolutifs
4. Combinaisons de codes

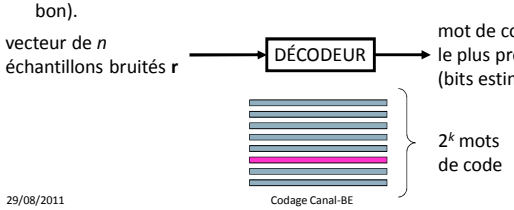
Modèle de la chaîne de transmission

- Bits équiprobables.
- Canal AWGN (Additive White Gaussian Noise) : ajout d'une source de bruit blanc gaussien, indépendant du signal émis, de PSD (Power Spectral Density) $N_0/2$.



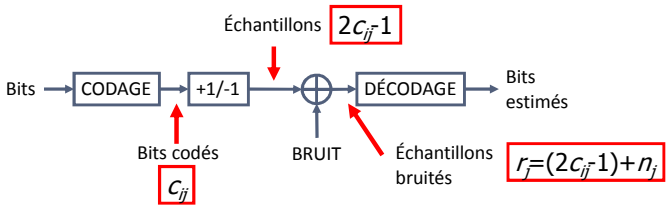
Décodage optimal soft-decision

- Décodage soft-decision = décodage à partir des valeurs des échantillons reçus et non pas sur des estimations de bits (0 ou 1).
- Rôle du décodeur : à chaque bloc de *n* bits, décider quel est le mot de code qui a été émis et en déduire le mot de données de *k* bits.
- Le décodeur optimal est celui qui maximise la probabilité de choisir le bon mot de code (ou de minimiser la probabilité de ne pas choisir le bon).



Modèle des échantillons reçus

- Pour chaque mot $\mathbf{C}_i$, les composantes c_{ij} du vecteur sont des 0 et des 1 (j=1…n).
- Les échantillons n_j sont des variables aléatoires gaussiennes, indépendantes,centrées, de variance $N_0/2$.



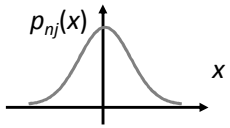
Densité de probabilité des échantillons de bruit

- La densité de probabilité du vecteur **n** des *n* échantillons de bruit est une gaussienne à *n* dimensions.

$\mathbf{n}=[n_0 \dots n_n]$

$$p(\mathbf{n})=\prod_{j=1}^{n} p(n_j)=\prod_{j=1}^{n}\frac{1}{\sqrt{2\pi\frac{N_0}{2}}}\exp\left\{-\frac{n_j^2}{2\frac{N_0}{2}}\right\}$$

$$=(\pi N_0)^{-\frac{n}{2}}\exp\left\{-\sum_{j=1}^{n}\frac{n_j^2}{N_0}\right\}$$



Densité de probabilité gaussienne du vecteur **r**

- La densité de probabilité du vecteur **r** des *n* échantillons reçus est une gaussienne à *n* dimensions.
- Hypothèse : le mot $\mathbf{C}_i$ est émis.
- La densité de probabilité de **r** est conditionnée par le symbole émis.

$\mathbf{r}=[r_0 \dots r_n]$

$$r_j=(2c_{ij}-1)+n_j \Leftrightarrow n_j=r_j-(2c_{ij}-1)$$

Expression de la densité de probabilité $p(\mathbf{r}|\mathbf{C}_i)$

- La densité de probabilité du vecteur des échantillons reçus est conditionnée par le mot de code émis $\mathbf{C}_i$.

Indépendance des échantillons de bruit

$$p(\mathbf{r}|\mathbf{C}_i) = \prod_{j=1}^{n} p(r_j|c_{ij}) = (\pi N_0)^{-\frac{n}{2}} \exp\left\{-\sum_{j=1}^{n} \frac{[r_j - (2c_{ij}-1)]^2}{N_0}\right\}$$

Rôle du décodeur

- Entrée : vecteur $\mathbf{r}$ des n échantillons bruités.
- Sortie : estimation du mot de code émis (puis décodage pour restituer le mot de donnée émis).
- Rôle du décodeur : décider quel mot de code a été émis en fonction des observations disponibles : $\mathbf{r}$.
- Critère de décision : le mot de code estimé est, parmi tous les $\mathbf{C}_i$, celui qui a la probabilité d'occurrence la plus forte (celui qui est le plus probable) en fonction des observations réalisées $\mathbf{r}$.
- Le mot de code estimé est donc celui qui maximise la probabilité $P[\mathbf{C}_i|\mathbf{r}]$.

$$\hat{\mathbf{C}} = \underset{\mathbf{C}_i}{\operatorname{argmax}} P[\mathbf{C}_i|\mathbf{r}]$$

Estimateurs

- Estimateur MAP (Maximum A Posteriori)
  - L'estimateur du maximum a posteriori choisit la probabilité maximale après avoir reçu les observations.
  - Procédure : calculer la distribution $P[\mathbf{C}_i|\mathbf{r}]$ et sélectionner le mot de code qui donne la plus grande valeur de $P[\mathbf{C}_i|\mathbf{r}]$.
- Estimateur du maximum de vraisemblance
  - L'estimateur du maximum de vraisemblance cherche le mot $\mathbf{C}_i$ qui maximise la densité de probabilité des échantillons reçus.
  - Estimateur MLSE : maximum likelihood sequence estimator.

$$\hat{\mathbf{C}} = \underset{\mathbf{C}_i}{\operatorname{argmax}} P[\mathbf{C}_i|\mathbf{r}] \qquad \hat{\mathbf{C}} = \underset{\mathbf{C}_i}{\operatorname{argmax}}\, p(\mathbf{r}|\mathbf{C}_i)$$

Équivalence des estimateurs ML et MAP

- Utilisation de la règle de Bayes.
- Hypothèses :
  - les mots sont équiprobables à l'émission : $P[\mathbf{C}_i]$=cste,
  - les observations ne dépendent pas d'un i particulier.
- Estimateur ML et MAP équivalents.
- Raisonnement à partir de l'estimateur ML car estimateur asymptotiquement sans biais et efficace.

$$\hat{\mathbf{C}} = \underset{\mathbf{C}_i}{\operatorname{argmax}} P[\mathbf{C}_i|\mathbf{r}] \qquad P[\mathbf{C}_i|\mathbf{r}] = \frac{p(\mathbf{r}|\mathbf{C}_i)P[\mathbf{C}_i]}{p(\mathbf{r})} \qquad \hat{\mathbf{C}} = \underset{\mathbf{C}_i}{\operatorname{argmax}}\, p(\mathbf{r}|\mathbf{C}_i)$$

Densité de probabilité p(**r**|**C**$_i$)

- Densité de probabilité de **r** | **C**$_i$: gaussienne à *n* dimensions.
- Échantillons indépendants donc densité de probabilité du vecteur = produit des densités de probabilité sur les échantillons.

$$p(\mathbf{r}|\mathbf{C}_i)=\prod_{j=1}^{n}p(r_j|c_{ij})=(\pi N_0)^{-\frac{n}{2}}\exp\left\{-\sum_{j=1}^{n}\frac{\left[r_j-(2c_{ij}-1)\right]^2}{N_0}\right\}$$

Maximisation de la probabilité conditionnelle

$$\begin{aligned}\underset{\mathbf{c}_i}{\text{argmax}}\,p(\mathbf{r}|\mathbf{C}_i)&=\underset{\mathbf{c}_i}{\text{argmax}}\left\{(\pi N_0)^{-\frac{n}{2}}\exp\left\{-\sum_{j=1}^{n}\frac{\left[r_j-(2c_{ij}-1)\right]^2}{N_0}\right\}\right\}\\&=\underset{\mathbf{c}_i}{\text{argmax}}\left\{-\frac{n}{2}(\pi N_0)-\frac{1}{N_0}\sum_{j=1}^{n}\left[r_j-(2c_{ij}-1)\right]^2\right\}\\&=\underset{\mathbf{c}_i}{\text{argmax}}\left\{-\frac{1}{N_0}\sum_{j=1}^{n}\left[r_j-(2c_{ij}-1)\right]^2\right\}\\&=\underset{\mathbf{c}_i}{\text{argmin}}\left\{\sum_{j=1}^{n}\left[r_j-(2c_{ij}-1)\right]^2\right\}=\underset{\mathbf{c}_i}{\text{argmin}}D^2(\mathbf{r},\mathbf{C}_i)\end{aligned}$$

Estimateur MLSE

- L'estimateur MLSE cherche la distance euclidienne minimale entre les *n* échantillons reçus et les *n* éléments des 2^k mots de codes.
- Circuit de décision
  - Calcul de 2^k distances euclidiennes D(**r**,**C**$_i$) i=1,..,2^k.
  - Sélection du mot de code **C**$_i$ qui donne la distance la plus petite.

$$\hat{\mathbf{C}}=\underset{\mathbf{c}_i}{\text{argmax}}\,p(\mathbf{r}|\mathbf{C}_i)=\underset{\mathbf{c}_i}{\text{argmin}}\left\{\sum_{j=1}^{n}\left[r_j-(2c_{ij}-1)\right]^2\right\}=\underset{\mathbf{c}_i}{\text{argmin}}D^2(\mathbf{r},\mathbf{C}_i)$$

Circuit de décision – amélioration

- Principe : chercher le mot le code le plus corrélé avec le mot reçu.
- Comparaison de la séquence de *n* valeurs r_j aux 2^k mots de codes possibles par les métriques CM_i *i*=1,..,2^k.
- Sélection du mot de code qui donne la métrique la plus grande.

$$\underset{\mathbf{c}_i}{\text{argmax}}\,p(\mathbf{r}|\mathbf{C}_i)=\underset{\mathbf{c}_i}{\text{argmax}}\,CM_i \qquad CM_i=\sum_{j=1}^{n}(2c_{ij}-1)r_j$$

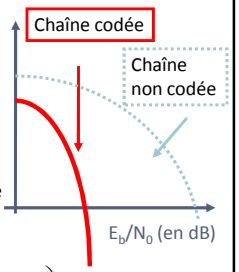
$$\underset{\mathbf{c}_i}{\text{argmin}}\left\{\sum_{j=1}^{n}\left[r_j-(2c_{ij}-1)\right]^2\right\}=\underset{\mathbf{c}_i}{\text{argmax}}\left\{\sum_{j=1}^{n}2r_j(2c_{ij}-1)\right\}$$

Limitation

- Méthode simple mais très coûteuse en temps de calcul dès que k>10 (2^{10} comparaisons à chaque mot de n bits reçu).
- Réduction possible du nombre d'opérations par algorithme de Viterbi (voir partie sur les codes convolutionnels).

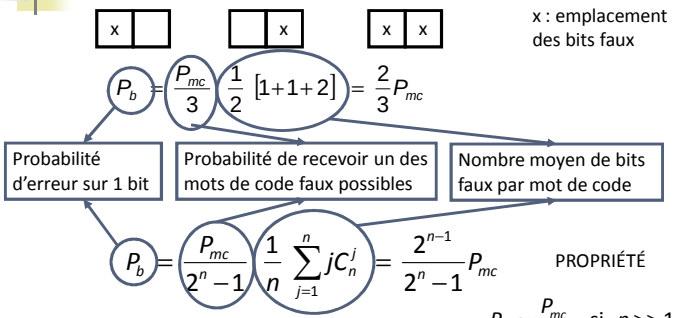
Probabilité d'erreur sur un mot de code

- En théorie : utilisation de la répartition des poids dans les mots de codes.
- En pratique : utilisation d'une borne supérieure indépendante de la répartition des poids.
- Exemple : BPSK
- À forts SNR par bit donné, la probabilité d'erreur pour une chaîne codée est plus faible que pour une chaîne non codée.
- À faibles SNR par bit, c'est l'inverse.



$$P_{non\_cod} < \frac{1}{2}\exp\left(-\frac{E_b}{N_0}\right) \quad P_{cod} < \exp\left(-\frac{E_b}{N_0}R_c d_{\min} + k\ln 2\right) \quad E_b^{(c)}k = nE_b^{(nc)}$$

Différence entre erreur sur un bit et erreur sur un mot



Conclusion sur le décodage «soft decision»

- Décodage = estimation du mot de code émis = mot de code le plus proche du mot reçu (distance euclidienne).
- Amélioration par mesure de corrélation.

Plan du cours

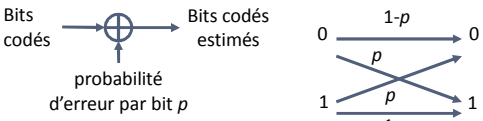
1. Introduction
2. **Codes en blocs linéaires**
   1. Matrice génératrice et matrice de contrôle de parité
   2. Codes cycliques
   3. Décodage optimal soft-decision
   4. **Décodage hard-decision**
3. Codes convolutifs
4. Combinaisons de codes

Décodage hard-decision

- Seule modification du récepteur : une décision est prise sur chaque échantillon du vecteur **r** avant d'entrer dans le bloc de décodage.
- Décodage par distance minimale.
- Entrée : mot de *n* bits.
- Décodage : comparaison entre le mot de code reçu et les 2^k mots de code possibles et sélection du mot de code le plus proche du mot reçu au sens de la distance de Hamming.

Canal BSC

- BSC = Binary Symetric Channel.
- E_c = énergie transmise par bit codé.
- $N_0/2$ = densité spectrale de puissance du bruit.



$$p = \frac{1}{2} erfc\left(\sqrt{\frac{E_c}{N_0}}\right)$$

Propriété du décodeur

- Décodage optimal au sens de la probabilité d'erreur sur un mot de code (maximum de vraisemblance).
- Inconvénient : 2^k comparaisons.
- Amélioration : méthode du syndrome.

Décodage par syndrome et *Look-up Table* (Table de vérification)

- $\mathbf{C}_m$, le vecteur des bits codés émis
- **Y**, le vecteur des bits codés estimés
- **H**, la matrice de contrôle de parité
- **S**, le syndrome
- **e**, un vecteur d'erreur dont les composantes sont
  - 1 s'il y a une erreur
  - 0 s'il n'y en a pas.

$$\mathbf{Y} = \mathbf{C}_m + \mathbf{e}$$

$$\mathbf{S} = \mathbf{Y}\mathbf{H}^T = (\mathbf{C}_m + \mathbf{e})\mathbf{H}^T$$
$$= \mathbf{e}\mathbf{H}^T$$

Décodage par syndrome

- Décodage par syndrome :
  - calcul du syndrome
  - repérage de l'erreur correspondante
  - correction du mot reçu
- Intérêt du décodage par syndrome
  - Le vecteur **S** (de taille *n-k*) a des composantes
    - nulles lorsque l'équation de contrôle de parité est satisfaite
    - non nulles sinon
  - Donc, 2^{n-k}-1 erreurs détectées.

$$\mathbf{Y} = \mathbf{C}_m + \mathbf{e}_i$$

$$\mathbf{S} = \mathbf{Y}\mathbf{H}^T$$
$$= (\mathbf{C}_m + \mathbf{e}_i)\mathbf{H}^T$$
$$= \mathbf{e}_i\mathbf{H}^T$$

$$\hat{\mathbf{C}}_m = \mathbf{Y} \oplus \mathbf{e}_i$$

Exemple : code C(7,4)

type d' erreur	syndrome
1000000	100
0100000	010
0010000	001
0001000	110
0000100	011
0000010	111
0000001	101

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

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

Exemple : code C(7,4)

- Réception : [1001111] / Syndrome : [011]
- Erreur correspondante : [0000100]
- Correction = Réception + Erreur correspondante = [1001011].
- Limitation du décodage par syndrome : un syndrome nul signifie qu'il n'y a pas d'erreur détectable.
- Exemple : si vecteur d'erreur = mot de code
  - mot reçu = mot de code + mot de code = mot de code.
  - Syndrome (mot de code)=0 donc pas d'erreur détectée.

Décodage des codes cycliques

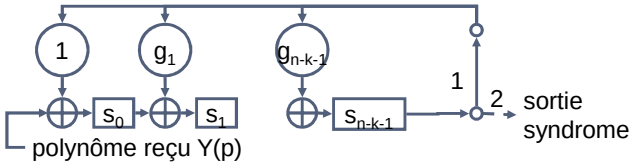
- Calcul du syndrome par division polynomiale.
- Soient les polynômes C(*p*),Y(*p*) et e(*p*) associés respectivement au mot de code **C**, au mot reçu **Y** et à l'erreur **e**.
- Procédure :
  - Division de Y(*p*) par le polynôme générateur g(*p*).
  - Le reste de la division R(*p*), de degré inférieur ou égal à *n-k*-1, représente le polynôme du syndrome.
  - Le syndrome est nul si Y(*p*) est un mot de code.

$$Y(p) = C(p) + e(p)$$
$$= X(p)g(p) + e(p)$$

$$Y(p) = Q(p)g(p) + R(p)$$

Division de Y(p) par g(p)

1. Les *n* bits du mot de code reçu **Y** sont passés dans le circuit (interrupteur en position 1).
2. Les *n-k* registres contiennent les bits du syndrome qui sont acheminés vers la sortie (interrupteur en position 2).



Look-up table

- Après calcul du syndrome : recherche de l'erreur correspondante $\mathbf{e}_c$ dans la table de vérification (look-up table) puis correction.
- Solution utilisée tant que *n-k*<20.
- Sinon, utilisation de codes BCH dont la complexité calculatoire du décodage est mois importante.

$$\hat{\mathbf{C}} = \mathbf{Y} \oplus \mathbf{e}_c$$

Dernière étape : décodage du mot de code

- 1ère idée : par une table donnant la correspondance entre les 2^k mots de codes et 2^k mots de données (pas d'opération matricielle disponible).
- Inconvénient : il faut stocker la table.
- 2ème idée : tirer profit de la propriété des codes systématiques (décodage = troncature du mot de code).

Remarque : protocoles de niveau supérieur ou égal à 2

- Pour le CSMA/CD ou IP, le calcul du syndrome se fait en temps réel : la détection d'erreur sur une trame ou un paquet est immédiate.
- En revanche, le temps de correction est ici rédhibitoire (recherche de la correspondance entre une valeur de syndrome et une forme d'erreur particulière).
- Conséquence : pas de correction dans les couches supérieures ou égale à 2 (choix de la retransmission).

Probabilité d'erreur en mode hard decision

- p est la probabilité d'erreur sur 1 bit.
- d_{min} la distance minimale du code.

- En pratique, le hard decision est moins performant que le soft decision.

$$P_{mc} \leq \left(2^k - 1\right)\left[4p(1-p)\right]^{\frac{d_{min}}{2}}$$

Importance de d_{min} dans le dimensionnement des codes

$$d_{min} \leq n - k + 1$$

- Pour les codes binaires, pas de codes pour atteindre la borne supérieure.
- Pour les codes non binaires, existence de codes (comme les codes RS) permettant d'atteindre la borne supérieure.

Codes non binaires

- Code en bloc non binaire : ensemble de mots de code dans lesquels les composantes appartiennent à un alphabet à q symboles.
- En pratique : $q=2^k$ (k bits donne un symbole).
- N : longueur du mot de code
- K : longueur du mot de donnée
- D_{min} : distance minimale du code

Codes RS

- Code RS (Reed Solomon) : code BCH primitif de longueur $N=q-1$ sur GF(q) où q est de la forme 2^k.
- Existence de tables pour les polynômes générateurs.

$$N = q - 1 = 2^k - 1$$
$$K = 1,2,3,\cdots,N-1$$
$$D_{\min} = N - K + 1$$
$$R_c = \frac{K}{N}$$
$$t = Ent\left[\frac{1}{2}(D_{\min} - 1)\right]$$
$$= Ent\left[\frac{1}{2}(N - K)\right]$$

Avantages des codes RS

- Correction d'erreurs groupées ou paquets d'erreurs (burst en anglais).
- Exemple : code RS(15,11), $15=2^4-1$
  - Pouvoir de correction t = 2 éléments quaternaires
  - Soit de 2 à 8 bits.
- Codes utilisés pour le stockage des données sur support CD et dans les systèmes de communications subissant des erreurs par rafales.

Exemple : DVB-S

- Utilisation d'un code raccourci RS(204,188,T=8) obtenu à partir d'un code original RS(255,239, T=8) appliqué à chaque groupe de 188 octets.

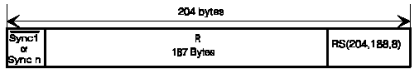


Figure 3c) Reed-Solomon RS (204,188, T=8) error protected packet

RS(204,188, T=8)

Paquets transport MPEG-2 (188 octets) → Ajout 51 octets à 0 (début de paquet) → RS (255, 239) → Suppression 51 octets (début de paquet) → Paquets transport MPEG-2 protégés (204 octets)

Conclusion sur le décodage «hard decision»

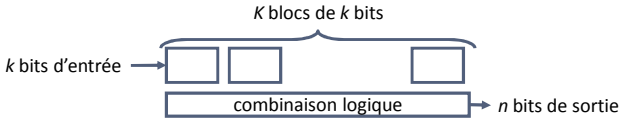
- Décodage = estimation du mot de code émis = mot de code le plus proche (distance de Hamming).
- Amélioration : décodage par syndrome.

Plan du cours

1. Introduction
2. Codes en blocs linéaires
3. **Codes convolutifs**
   1. Décodage optimal – algorithme de Viterbi
   2. Codes poinçonnés
4. Combinaisons de codes

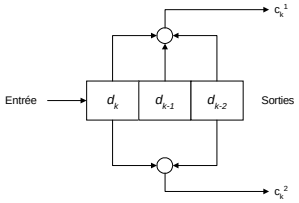
Codes convolutifs

- Principe du codage :
  - chaque bloc de *k* bits en entrée donne un bloc de *n* bits en sortie.
  - chaque bloc de *n* bits dépend des *K* blocs de *k* bits précédents.
- Caractéristiques des codes
  - Taux de codage : *k*/*n*
  - Longueur de contrainte : *K*
- Si les *k* bits d'information se retrouvent dans le bloc de *n* bits en sortie du codeur, le code est dit systématique.



Représentation des codes

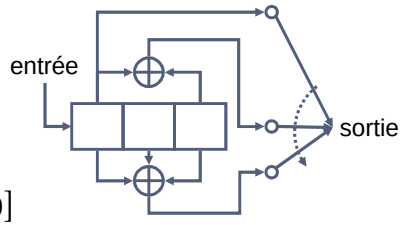
- *n* sorties.
- sortie = addition modulo 2 d'une sélection des Kk bits d'entrée.
- A chacune des *n* sorties correspond un vecteur de Kk éléments (fonction génératrice) :
  - 1 si l'élément participe à la somme,
  - 0 sinon.
- Exemple : codeur convolutif de rendement R_c=1/2 et de longueur de contrainte K = 3.
  - Entrée constituée par des blocs de k = 1.
  - Sortie constituée par des blocs de n = 2.



Exemple

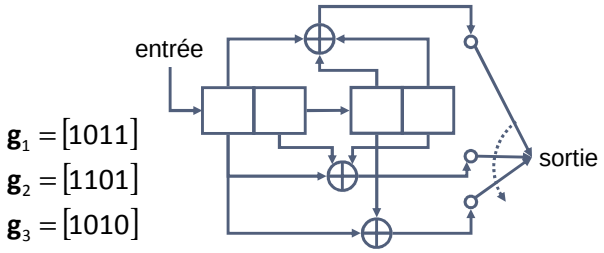
- Codeur 1/3
- *K*=3 // *k*=1 // *n*=3
- Registres initialisés à 0.
- Entrée 1 Sortie 111
- Entrée 0 Sortie 001
- Entrée 1 Sortie 100
- Fonctions génératrices $\mathbf{g}_1$, $\mathbf{g}_2$ et $\mathbf{g}_3$:

$$\mathbf{g}_1 = [100]$$
$$\mathbf{g}_2 = [101]$$
$$\mathbf{g}_3 = [111]$$



Exemple

- K=2, k=2, n=3



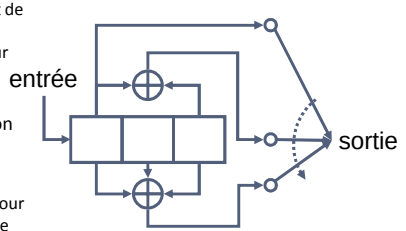
$\mathbf{g}_1 = [1011]$

$\mathbf{g}_2 = [1101]$

$\mathbf{g}_3 = [1010]$

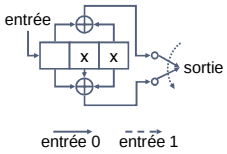
Remarques

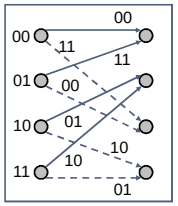
- Caractère convolutif du codeur :
  - La sortie du codeur peut être interprétée comme le produit de convolution entre l'entrée du codeur et la réponse du codeur définie par ses fonctions génératrices.
- Autres formes de représentation
  - Arbre
  - Diagramme d'état
  - Treillis : représentation utile pour l'algorithme Viterbi, algorithme de décodage le plus utilisé pour les codes convolutifs.



Treillis d'un code convolutif

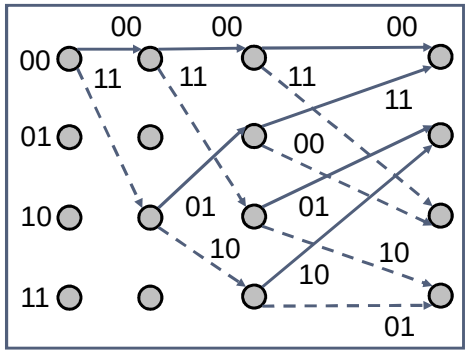
- 3 informations fournies :
  - Transitions d'un état du codeur vers un autre en fonction des entrées.
  - Valeurs des entrées et des sorties correspondantes.
- État du codeur : états des registres (hormis les cases mémoires réservées aux entrées).
- Exemple : codeur C(1/2, 3).





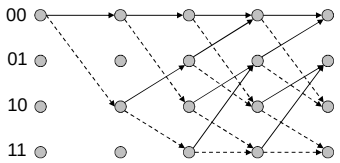
Régime transitoire et régime permanent

- Au bout de *K* transitions, le treillis atteint son régime permanent.



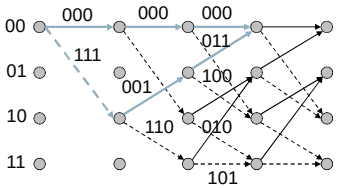
Généralisation

- Code C($k/n,K$).
- 2^k branches entrant dans chaque nœud.
- 2^k branches sortant de chaque nœud.
- $2^{k(K-1)}$ états possibles.



Distance libre minimale (Hamming)

- Distance minimale
  - le chemin tous-zéros
  - le chemin partant et revenant à l'état tout zéro en un nombre de minimal de transitions.
- Plus grande est la d_{min}, meilleur est le code.
- Existence de tables pour construire des codes à distance minimale la plus grande (cf. Proakis).
- Exemple : C(1/2,3), $d_{min} = 5$
  - fonctions génératrices (en octal) : 5 et 7



$d_{min}=6$

Plan du cours

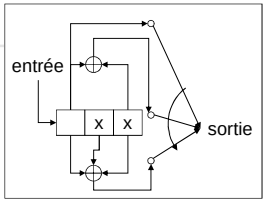
1. Introduction
2. Codes en blocs linéaires
3. **Codes convolutifs**
   1. **Décodage optimal – algorithme de Viterbi**
   2. Codes poinçonnés
4. Combinaisons de codes

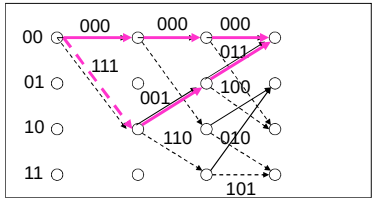
Décodeur optimal

- Estimateur séquentiel du maximum de vraisemblance MLSE : recherche, à travers le treillis, de la séquence la plus vraisemblable.
- Distance de Hamming ou euclidienne suivant qu'il s'agit de hard ou soft decision.

Explication du principe sur un exemple simple

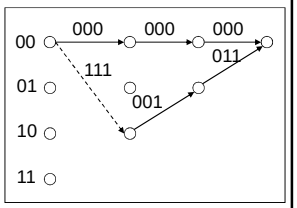
- Soit le codeur C(1/3,3).
- Il s'agit d'aller de l'état 00 à l'état 00 en trois transitions.
- Hypothèse : le récepteur sait que l'information envoyée permet d'aller de l'état 00 à l'état 00 en trois transitions.
- Enjeu : choisir entre les deux chemins possibles
  - Chemin i=0 : Entrée : 000 , Sortie : 000 000 000
  - Chemin i=1 : Entrée : 100 , , Sortie : 111 001 011





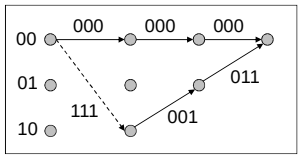
Démarche

- Soit la séquence reçue : 0,5 -0,5 0,33 -0,4 -0,3 -0,2 1,1 -0,2 -0,3
- Principe : comparer la séquence reçue aux séquences possibles grâce à des métriques.
  - Calculer des métriques transition par transition puis les accumuler pour former des métriques de chemin.
- Deux solutions pour les métriques
  - Soft Decision : distance euclidienne
  - Hard Decision : distance de Hamming



Données

- Soient r_{jm} les échantillons reçus
  - j représente l'indice de branche = 1,2,…,B
  - m représente le numéro du bit = 1,2,…,n
  - Échantillons réels utilisés par les décodeurs « soft decision ».
- Exemple pour B=3 et n=3 : 0,5 -0,5 0,33 -0,4 -0,3 -0,2 1,1 -0,2 -0,3
- Soient y_{jm} les décisions binaires sur r_{jm}
  - Valeurs binaires utilisées par les décodeurs « hard decision ».
- Exemple pour B=3 et n=3 : 101 000 100
- Soient les bits possibles $c_{jm}^{(i)}$
  - i représente le chemin possible



Méthode

$$PM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)}$$

- Calculer une métrique pour chaque chemin possible et choisir le chemin qui a la meilleure métrique.
- Les métriques de chemin s'obtiennent en additionnant les métriques sur chaque branche.

NA : Non Applicable

	Optimisation	Hard Decision	Soft Decision
Corrélation	Maximiser	NA	x
Distance euclidienne	Minimiser	NA	x
Distance de Hamming	Minimiser	x	NA

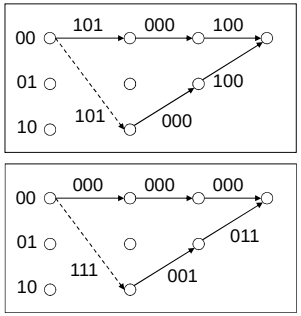
$$\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm}\left(2c_{jm}^{(i)} - 1\right) \qquad \mu_j^{(i)} = \sum_{m=1}^{n} \left[r_{jm} - \left(2c_{jm}^{(i)} - 1\right)\right]^2 \qquad \mu_j^{(i)} = \sum_{m=1}^{n} \overline{y_{jm} \oplus c_{jm}^{(i)}}$$

Exemple : C(1/3,3)

- Cas du hard decision

$$\mu_j^{(i)} = \sum_{m=1}^{3} \overline{y_{jm} \oplus c_{jm}^{(i)}}$$

$$PM^{(0)} = 2+0+1 = 3$$
$$PM^{(1)} = 1+1+3 = 5$$



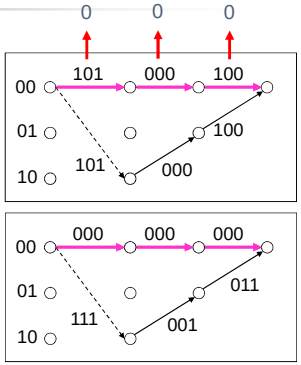
Critère de décision

- Une fois les métriques de chaque chemin calculées, il faut sélectionner le chemin qui a la meilleure métrique.
- Exemple : code C(1/3,3)
  - Si $PM^{(0)}=3$ et $PM^{(1)}=5$ et si la métrique est fondée sur le calcul de distance de Hamming, alors le meilleur chemin est le chemin 0.

$$\hat{i} = \arg\min_i \left[PM^{(i)} \right]$$

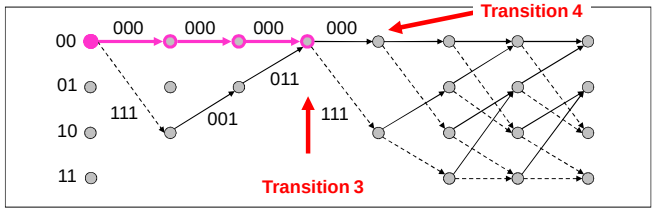
Dernière étape : le décodage

- Remonter le treillis de l'état d'arrivée à l'état de départ et re-parcourir le treillis vers la droite pour décoder les bits émis.
- Exemple : 000



Calcul des métriques de chemins à la 4ème transition

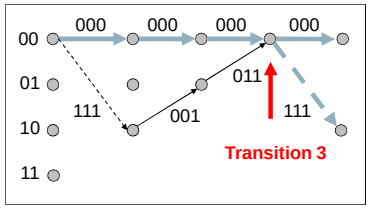
- Il y a 4 chemins possibles.
- Le fait de rajouter une transition ne modifie en rien la conclusion établie à la transition précédente.
- À partir de la transition 3, le chemin (1) peut être supprimé. Le chemin (0) est dit chemin survivant.



Conséquence du principe du chemin survivant

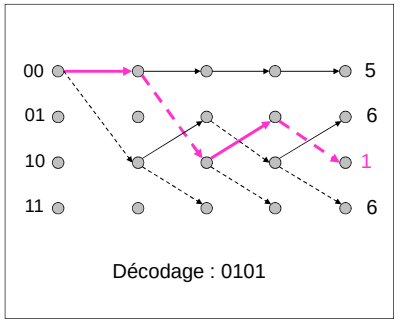
- La réduction du nombre de chemins réduit la complexité de l'algorithme.
- Exemple : seuls deux chemins sont à tester à la 4ème transition.
- Généralisation : à chaque transition, pour chaque état, ne conserver qu'un seul chemin.

Remarque : il faut que le treillis ait atteint son régime permanent (cad que tous les états soient atteints).



Décodage

- Après réception complète du bloc de données, sélection du chemin le plus probable et décodage.

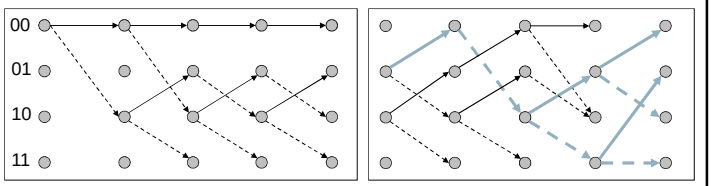


Algorithme de Viterbi : récapitulatif

- À chaque transition, calculer toutes les métriques de branches.
- Jusqu'à la transition K, calculer les métriques de tous les chemins.
- A la fin de la $K^{ième}$ transition, sélectionner les $2^{k(K-1)}$ chemins survivants (1 par état).
- À partir de la transition K+1,
  - calculer toutes les métriques de branches,
  - les ajouter aux métriques des chemins survivants,
  - et sélectionner les nouveaux chemins survivants.
- Après la réception complète du bloc de données, sélectionner le meilleur chemin survivant et en déduire les bits émis.
- Inconvénient de l'algorithme de Viterbi
  - Grand retard introduit au décodage sur de grands blocs de données.

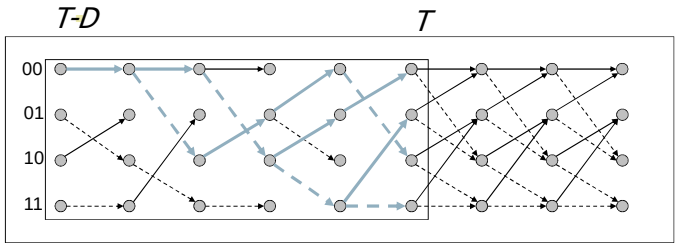
Observations empiriques

- En théorie : les chemins survivants peuvent provenir de plusieurs chemins.
- En pratique : les chemins survivants proviennent tous d'un même chemin (avec une probabilité voisine de 1).
- Conséquence : si tous les chemins survivants à la transition T proviennent d'un même chemin, il est possible de produire des bits de sortie avant même que le bloc de données ne soit complètement traité, cad à la transition T-D.



Méthode de décodage : fenêtre coulissante

- À la transition *T*, sortie du bit correspondant à la transition *T-D*.
- Valeur de *D* empirique : 5*K*.
- Dégradations négligeables sur les performances.

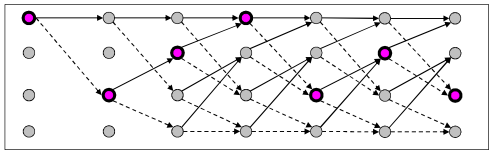


Limitation

- Si *k* bits en entrée et longueur de contrainte *K*, alors $2^{k(K-1)}$ états, donc $2^{k(K-1)}$ métriques et 2^k métriques calculées pour chaque état.
- Complexité calculatoire augmente exponentiellement avec *k* et *K*.
- Algorithme de Viterbi réservé pour petites valeurs de *k* et *K* (quelques unités).
- Autres algorithmes de décodage
  - Séquentiels : Fano (adapté pour les grandes longueurs de contrainte), Stack, Feedback.
  - Algorithmes à sorties soft : BCJR (Bak, Cocke, Jelinek, Raviv), SOVA (Soft Output Viterbi Algorithm).

Algorithme BCJR

- Algorithme soft input (échantillons réels à l'entrée du décodeur) et soft output (sorties sous la forme de rapport de vraisemblance sur la valeur des bits).
- Algorithme de décodage de base des turbo-codes.



Plan du cours

1. Introduction
2. Codes en blocs linéaires
3. **Codes convolutifs**
   1. Décodage optimal – algorithme de Viterbi
   2. **Codes poinçonnés**
4. Combinaisons de codes

Codes à fort rendement

- Codes à fort taux de codage (rendement), du type (n-1)/n.
  - Exemple : ¾.
- Avantage : économie de bande passante (peu de bits de redondance ajoutés par bit d'information).
- Inconvénient des codes à fort rendement : implique grande complexité calculatoire du décodeur.
- Solution : construire des codes à fort rendement à partir de codes à faible rendement et en supprimant des bits à l'émission.
- Technique du poinçonnage
  - Génération de codes (n-1)/n par poinçonnage de codes 1/n.
- Avantage :
  - codes à fort rendement de type (n-1)/n,
  - décodage peu calculatoire des codes 1/n.

Opération de poinçonnage

- Suppression périodique de bits à la sortie du codeur.
- Code de départ : code 1/n.
- Période de poinçonnage P.
- P bits en entrée du codeur.
- nP bits en sortie.
- Utilisation d'une matrice de poinçonnage **P**
  - 1 si le bit est transmis
  - 0 sinon

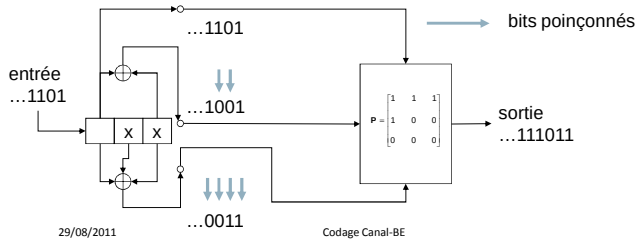
$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{21} & \cdots & p_{1P} \\ p_{21} & p_{22} & & p_{2P} \\ \vdots & \vdots & & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nP} \end{bmatrix}$$

- Suppression de N bits.
- Taux de la forme : $P/(nP-N)$ avec N entier de 0 à (n-1)P-1.
- Taux de la forme : $P/(P+M)$ avec M=1,2,…,(n-1)P.

Exemple

- Taux de codage visé : ¾
- Taux de codage de départ : 1/3
- n=3, P=3, nP=9, N=5
- Taux final : $P/(nP-N)$ = 3/(9-5)=3/4

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



Décodage des codes poinçonnés

- Rappel : l'intérêt des codes poinçonnés réside dans leur construction à partir d'un code de rendement 1/n.
- Objectif : utiliser le décodeur correspondant au code de rendement 1/n.
- Enjeu : remplacer les bits poinçonnés par des informations qui n'influencent pas les décisions prises par le décodeur.
  - Pas de rajout de 0 ou de 1.
- Indication : codage des bits par +1/-1 avant transmission dans le canal.
- Solution : rajouter des échantillons à 0 aux endroits où les bits ont été poinçonnés et utiliser Viterbi avec le treillis du code 1/n.

Propriétés des codes poinçonnés

- Distance minimale
  - Possibilité de chercher la matrice de poinçonnage qui donne la distance minimale d_{min} la plus grande.
  - La distance minimale d_{min} du code poinçonné est égale ou inférieure d'un bit à la distance minimale du code à fort taux de codage sans poinçonnage.
- Inconvénient
  - Une erreur dans un code poinçonné se propage plus longtemps qu'une erreur dans un code non poinçonné.
  - Conséquence : fenêtre d'observation plus longue que 5K.

Exemple : DVB-S

- Code de base convolutif de rendement ½ et de longueur de contrainte K=7.
- Peut être poinçonné pour donner des codes de rendement 2/3, ¾, 5/6 et 7/8.

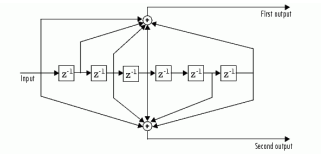


Table 2: Punctured code definition

Original code			Code rates									
			1/2		2/3		3/4		5/6		7/8	
K	G_1 (X)	G_2 (Y)	P	d_{free}	P	d_{free}	P	d_{free}	P	d_{free}	P	d_{free}
7	171_{OCT}	133_{OCT}	X: 1 Y: 1 I=X_1 Q=Y_1	10	X: 1 0 Y: 1 1 I=X_1 Y_2 Y_3 Q=Y_1 X_3 Y_4	6	X: 1 0 1 Y: 1 1 0 I=X_1 Y_2 Q=Y_1 X_3	5	X: 1 0 1 0 1 Y: 1 1 0 1 0 I=X_1 Y_2 Y_4 Q=Y_1 X_3 X_5	4	X: 1 0 0 0 1 0 1 Y: 1 1 1 1 0 1 0 I=X_1 Y_2 Y_4 Y_6 Q=Y_1 Y_3 X_5 X_7	3

NOTE: 1 = transmitted bit
0 = non transmitted bit

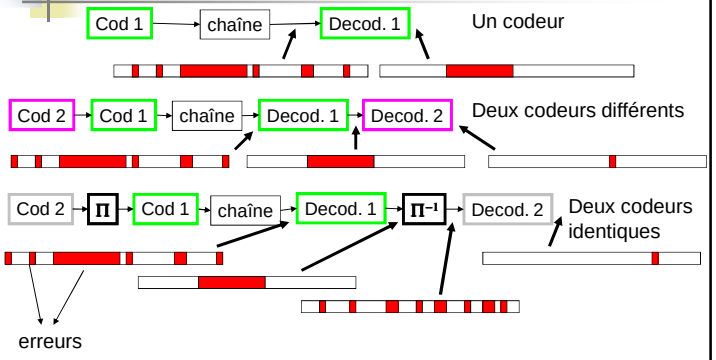
Plan du cours

1. Introduction
2. Codes en blocs linéaires
3. Codes convolutifs
4. **Combinaisons de codes**

Combinaisons de codes

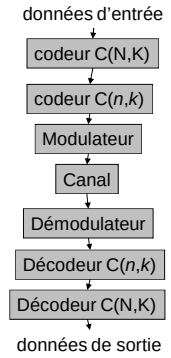
- Concaténation de codes
- Codage source canal conjoint
- HARQ (Hybrid Automatic Repeat Request)

Opérations sur les codes



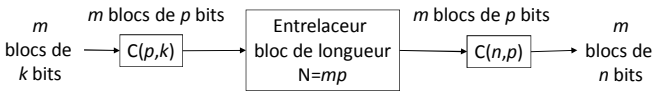
Concaténation de codes

- En pratique :
  - Code extérieur non binaire C(N,K),
  - Code intérieur binaire C(n,k).
- Blocs d'entrée de taille kK bits, séparés en K blocs (symboles) de k bits chacun et codés en N blocs de k bits chacun par C(N,K).
- Chacun des symboles de k bits est codé en mots de code de n bits C(n,k).
- Code résultant C(Nn,Kk).
- Taux de codage : Kk/Nn.
- Décodage hard decision pour les deux codeurs possibles.
- Décodage soft decision possible pour le code intérieur (si 2^k pas trop élevé).

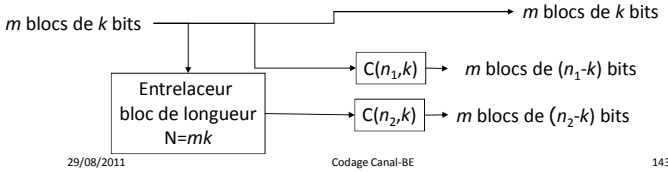


Combinaison série et parallèle

- taux de codage : k/n

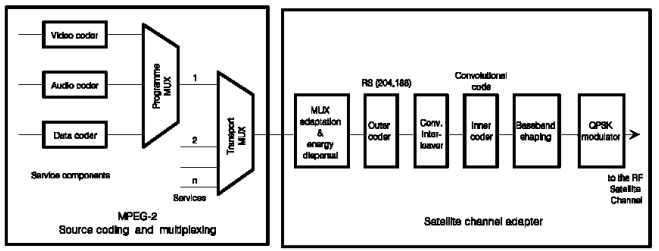


- taux de codage : $k/(n_1+n_2-k)$



Exemple : DVB-S

- Code RS et code convolutif séparés par un entrelaceur de type Forney.

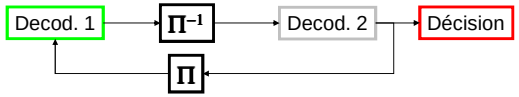


Décodage des concaténations série ou parallèle

- Décodage itératif (soft-in, soft-out) de type MAP :
  - Benedetto (1998) pour les codes série,
  - Berrou (1993) pour les codes parallèle.
- Famille des turbo-codes.

Turbo-Codes

- Utiliser les résultats du deuxième décodeur dans le premier codeur.
- Besoin de décodeurs produisant une information non binaire (rapport de vraisemblance) sur les bits : utilisation de codes convolutifs et d'un algorithme de décodage de type BCJR.
- Performances proches des limites théoriques de Shannon (exemple : code ½, $N=2^{16}$, 18 itérations, 10^{-5} pour SNR=0,6 dB).
- Limitations
  - Retard et complexité calculatoire compensés par les excellentes performances.
  - Meilleures performances avec une concaténation série pour des BER inférieurs à 10^{-2}.

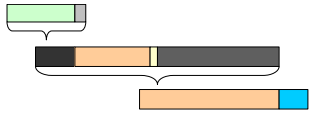


Codage source canal conjoint

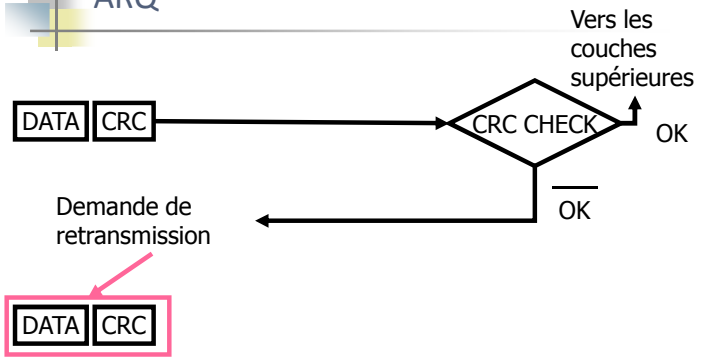
- Exemple : protection hiérarchique dans le GSM.
- Sortie du codeur de parole :
  - 260 bits toutes les 20 ms (soit débit de 13 kbit/s).
- Classement des 260 bits suivant leur criticité dans le rendu du signal de parole.
- Principe : mieux protéger les bits les plus sensibles.
- Résultat : 3 classes, de la plus sensible à la moins sensible : 50, 132 et 78 bits.

Protection hiérarchique

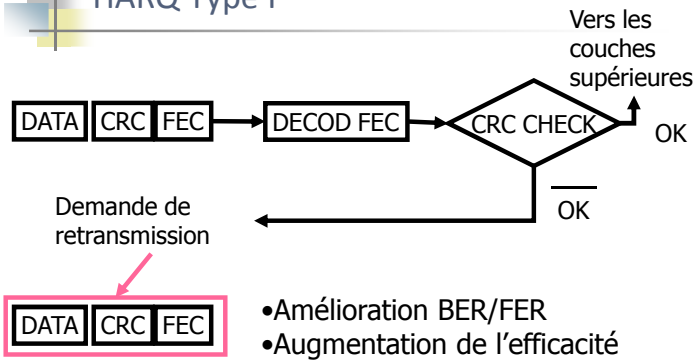
- La 1ère classe est codée par un CRC de 3 bits : 50+3=53.
- La 2ème classe (132 bits) est concaténée aux 53 bits (+ 4 bits), soit 189 bits.
- Le tout est codé par un code convolutif de rendement ½ : 378 bits à la sortie.
- La 3ème classe est concaténée (78 bits) sans protection.
- Total : 456 bits toutes les 20 ms (soit débit de 22,8 kbit/s).



ARQ

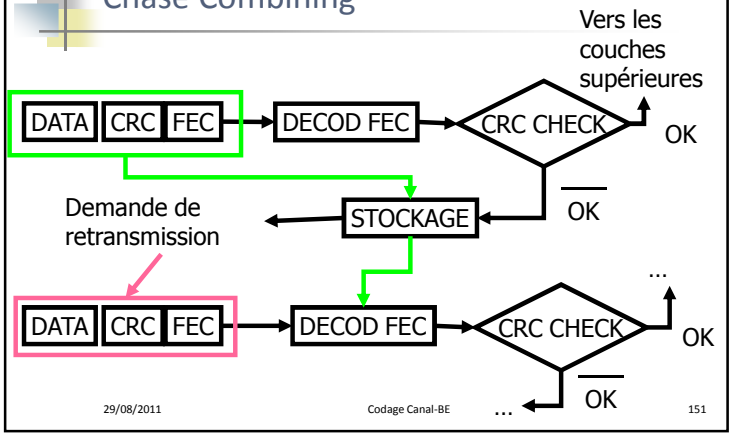


HARQ Type I

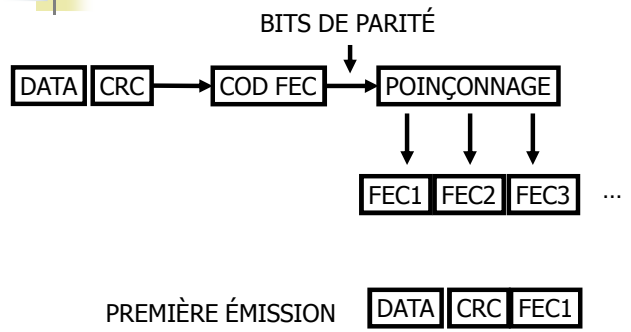


- Amélioration BER/FER
- Augmentation de l'efficacité spectrale

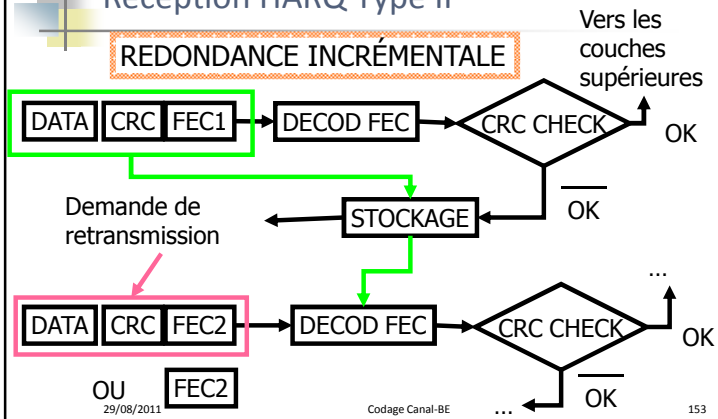
Amélioration : décodeurs « soft » et Chase Combining



Émission HARQ Type II



Réception HARQ Type II



Conclusion générale

- Codage canal : code correcteurs d'erreurs
- Familles de codes étudiées : codes en blocs linéaires, codes cycliques, codes convolutifs.
- Applications :
  - Transport de données
  - Stockage de données

1. Introduction
2. Codes en blocs linéaires
   1. Matrice génératrice et matrice de contrôle de parité
   2. Codes cycliques
   3. Décodage optimal soft-decision
   4. Décodage hard-decision
3. Codes convolutifs
   1. Décodage optimal – algorithme de Viterbi
   2. Codes poinçonnés
4. Combinaisons de codes

Annexe : rappels d'algèbre

- Rappels sur les corps, les groupes et les espaces vectoriels.

Rappel sur les corps

- Corps = ensemble G d'éléments muni de deux opérations (l'addition + et la multiplication .) vérifiant les propriétés suivantes :
  - L'ensemble G muni de l'addition + est un groupe commutatif.
  - L'élément neutre de l'addition est noté 0.
  - L'ensemble des éléments non-nuls de G muni de la multiplication . est un groupe commutatif.
  - L'élément neutre de la multiplication . est noté 1.
  - La multiplication est distributive par rapport à l'addition.

Rappel sur les groupes

- Un ensemble G sur lequel est défini une opération * est un groupe si
  - L'opération est associative,
  - G contient un élément e (appelé élément neutre de G),
  - Pour chaque élément a de G, il existe un élément *a*' de G appelé inverse de *a*.

$$a * e = e * a = a \qquad a * a' = a' * a = e$$

Espaces vectoriels

- Les mots de données et les mots de code appartiennent respectivement aux espaces vectoriels V_k et V_n. Ces espaces vectoriels sont munis de l'addition modulo-*q* et de la multiplication modulo-*q* sur un corps de Galois GF(*q*).
- Soit V un ensemble sur lequel est définie l'addition "+" et soit F un corps. La multiplication "." est définie entre les éléments de F et les éléments de V. Les éléments de V sont des vecteurs; ceux de F, des scalaires.
- V est un espace vectoriel sur le corps F si :
  - V muni de l'addition + est un groupe commutatif.
  - Pour chaque élément *a* dans F et chaque élément *v* dans V, *a*.*v* est un élément de V.
  - La multiplication est distributive par rapport à l'addition.
  - La multiplication est associative.
  - Si 1 est l'élément neutre pour la multiplication, alors 1.*v*=*v*.