CODAGE DE CANAL

CLASSIFIACTION des CODES

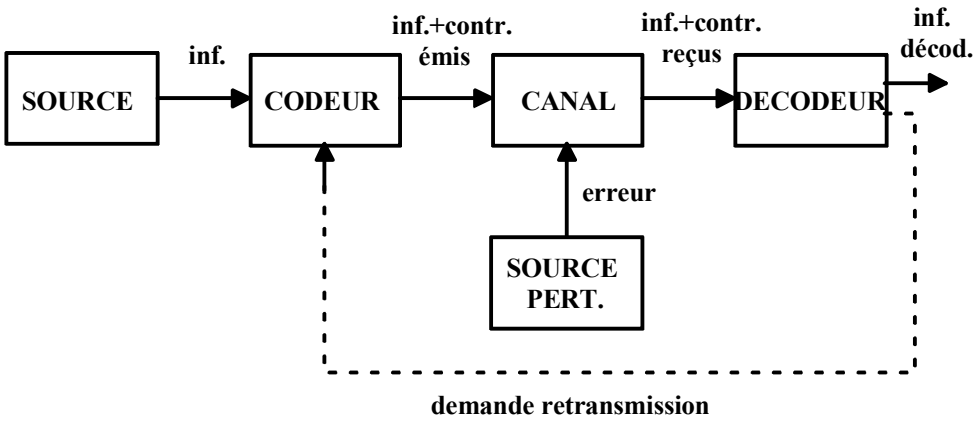
DETECTEURS et CORRECTEURS D'ERRORS

Dans le cas des **canaux à perturbations** le but du codage est de diminuer la probabilité d'erreur. Un tel codage implique l'augmentation de la redondance de la source primaire – c'est le prix à payer.

L'idée de base est la suivante : par codage on ajoute aux symboles de la source (**symboles** ou **bits d'information**) des **symboles** ou **bits de contrôle** (dans le cas des **codes binaires**, les symboles sont binaires et on les appelle **bits)**. Les bits de contrôle sont calculés de façon que le décodeur puisse vérifier si les symboles reçus sont erronés ou pas (**détection d'erreurs**), et, éventuellement, déterminer quelles sont les positions des bits erronés (**correction d'erreurs**).

Dans le cas des **codes détecteurs d'erreurs** :
- le décodeur doit déterminer si, dans une séquence donnée, il y a des bits erronés ;
- le décodeur doit demander la retransmission des séquences contenant des bits erronés, ce qui implique l'existence d'un canal inverse ;
- l'application doit 'accepter' un certain retard dû à la demande de retransmission et à la retransmission.

Dans le cas des **codes correcteurs d'erreurs** :
- le décodeur doit déterminer quelles sont les positions des bits erronés ;
- les bits erronés doivent être corrigés.



Les codes correcteurs d'erreurs nécessitent une redondance plus grande que les codes détecteurs d'erreurs, et la tâche du décodeur est beaucoup plus compliquée.

Classification des codes détecteurs et correcteurs d'erreurs :

- codes **en bloc**
- codes **convolutifs (convolutionnels)** ou **récurrents**.

Erreurs individuelles et paquets (ou rafales) d'erreurs

Si chaque bit transmis est affecté par des perturbations de façon indépendante, alors les erreurs qui apparaissent seront '**individuelles**', c'est à dire indépendantes les unes des autres. C'est ce qui se passe suite à l'action du bruit de fluctuation, dont un exemple est le bruit thermique.

Par contre, suite à l'action des perturbations d'impulsions ou des interruptions le long du canal de transmission les erreurs apparaissent groupées. C'est le cas des media de stockage (hard disc, CD), des transmissions de données à travers le réseau téléphonique, et aussi des communications mobiles. Dans ce cas on dit que les erreurs apparaissent en '**paquets**' ou '**rafales**'.

CODES EN BLOC

Les bits de ces codes sont groupés par **n** en des blocs, formant des **mots-code** de **longueur n**. Parmi ces **n** bits on a **m** bits d'information (générés par la source) et **r** bits de contrôle, ajoutés par le codeur.

Chaque bit d'information peut prendre une des valeurs 0 ou 1. La source peut générer n'importe quelle combinaison de **m** bits, donc il y a en tous $\mathbf{2^m}$ combinaisons possibles.

Les **r** bits de contrôle sont calculés par le codeur en fonction des **m** bits d'information, en utilisant certaines règles (plus précisément **r** équations) qui déterminent le code.

Le nombre de différentes combinaisons possibles de longueur **n** vaut $\mathbf{2^n}$. Mais parmi toutes ces combinaisons, seulement $\mathbf{2^m}$ représentent des mots-code, toutes les $2^n - 2^m$ autres combinaisons n'appartiennent pas au code.

A cause des perturbations un ou plusieurs bits d'un mot-code peuvent être erronés pendant la transmission, donc le mot-code est 'transformé' en un **mot reçu** qui peut valoir n'importe quelle combinaison de n bits. Le décodeur vérifie facilement si le mot reçu appartient ou pas au code. S'il n'y appartient pas il y a deux possibilités : soit la retransmission est demandée (détection d'erreur), soit le décodeur détermine les positions des bits erronés et les corrige en les

inversant (correction d'erreur). On peut représenter les mots-code et les mots reçus par des vecteurs n-dimensionnels, les n coordonnés binaires de chaque vecteur valant les bits des mots respectifs. On peut écrire ces vecteurs sous forme matricielle (tout au long de ce chapitre les matrices seront notées par des lettres **en gras**) :

mots-code $\mathbf{c} = [\, c_1 \;\; c_2 \;\; \ldots \; c_n \,]$
mots reçus $\mathbf{c'} = [\, c_1' \; c_2' \; \ldots \; c_n' \,]$

L'effet des perturbations peut être représenté par l'addition modulo-2 (fonction logique OU EXCLUSIF, EX-OR) bit par bit d'un mot-code avec un **mot erreur** $\boldsymbol{\varepsilon} = [\, \varepsilon_1 \;\; \varepsilon_2 \;\; \ldots \; \varepsilon_n \,]$:

$$\mathbf{c'} = \mathbf{c} + \boldsymbol{\varepsilon}$$
$$\mathbf{c'} = [\, c_1+\varepsilon_1 \;\; c_2+\varepsilon_2 \;\; \ldots \; c_n+\varepsilon_n \,]$$

avec + signifiant partout dans ce chapitre (sauf indication contraire) l'addition modulo-2 :

+	0	1
0	0	1
1	1	0

Attention : le passage d'un côté à l'autre d'une équation à sommes modulo-2 se fait sans changement de signe :

$$a + b = c \quad \rightarrow \quad a = c + b$$

En effet

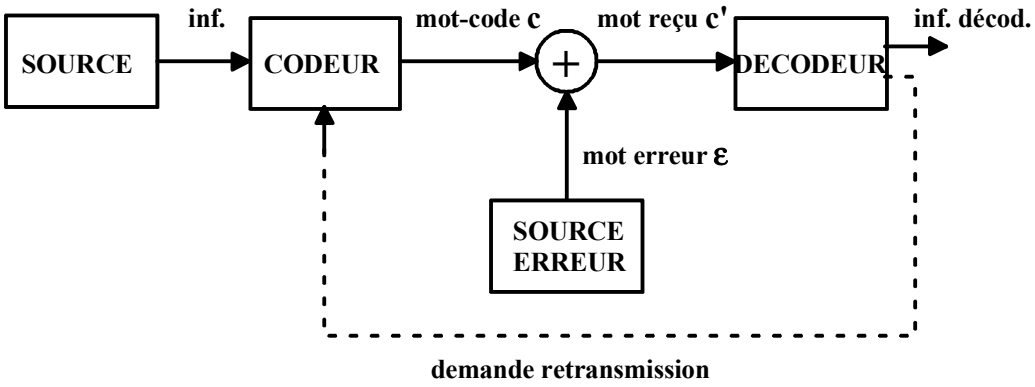
$$a + b + b = c + b,$$

$$a + 0 = c + b \quad \rightarrow \quad a = c + b$$

Les bits erronés du mot reçu se trouvent sur les positions où le mot erreur a des 1, car :

$$c_i' = c_i + 0 = c_i$$
$$c_i' = c_i + 1 = \overline{c_i}$$

Sur le schéma-bloc ci-dessous la source de perturbations est remplacée par une source des mots erreur, et le canal est remplacé par un EX-OR.



PROPRIETES DES CODES EN BLOC

Les codes en bloc sont linéaires si la somme (toujours modulo-2) de deux mots–code représente, elle aussi, un mot–code :

$\mathbf{c_a}$ et $\mathbf{c_b}$ mots-code → $\mathbf{c_a} + \mathbf{c_b}$ mot-code.

Il s'ensuit que le mot ayant tous les bits 0 est, également, un mot-code.

Distance Hamming et poids Hamming

La distance Hamming entre deux mots-code est définie comme le nombre des positions où les deux mots diffèrent. Pour la calculer il faut additionner modulo-2 bit par bit les deux mots-code est compter le nombre des 1 obtenus.

Exemple

La distance Hamming entre les mots 1011010 et 1001001 vaut 3 :

```
1011010 +
1001001
-----------
0010011
```

Le **poids Hamming** d'un mot-code vaut le nombre des bits 1 du mot. La **distance minimum** d'un code est la plus petite distance Hamming entre deux mots-code. Elle vaut le plus petit poids Hamming du code.

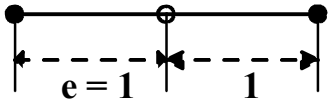
Distance minimum d'un code détecteur de e erreurs

Pour qu'un code puisse détecter un nombre de e erreurs il faut qu'en inversant e bits d'un mot-code il ne résulte un autre mot-code, ce qui implique une distance minimum d'au moins e + 1 (ici l'addition est 'normale') :

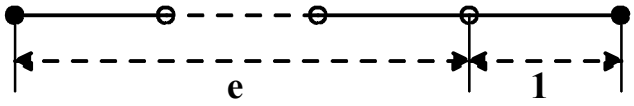
$$D_{min} \geq e + 1$$

En associant des points pleins aux mots-code et des points vides aux combinaisons non significatives, on peut donner une représentation spatiale de la distance Hamming.

Détection d'une erreur, e = 1, $D_{min} = e + 1 = 2$



Détection de e erreurs, $D_{min} = e + 1$



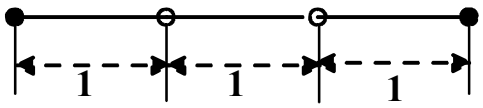
Distance minimum d'un code correcteur de e erreurs

La correction des erreurs implique que le décodeur décide de quel mot-code provient un mot reçu erroné. La décision du décodeur est prise en considérant que le mot reçu provient du mot-code se trouvant le plus près. Donc afin de pouvoir corriger e erreurs, il faut que le mot reçu se trouve à une distance e du mot-code duquel il provient et à une distance d'au moins e+1 de tous les autres mots-code, ce qui implique une distance minimum d'au moins 2e + 1 :

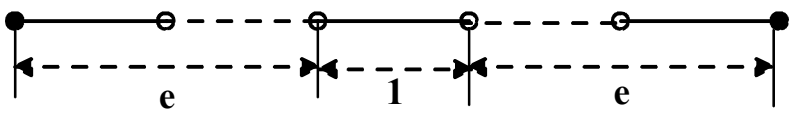
$$D_{min} \geq 2e + 1$$

TAUX D'ERREURS RESIDUELLES

Correction d'une erreur, $D_{min} = 3$



Correction de e erreurs, $D_{min} = 2e + 1$



Le **correcteur** (ou **syndrome**) est une combinaison de **r** bits que le décodeur calcule afin de détecter ou de corriger les e erreurs. Le principe du calcul du correcteur est le suivant :

- à partir des bits d'information reçus (corrects ou pas) le décodeur calcule les bits de contrôle qui leur correspondent ;
- le décodeur compare ces bits de contrôle qu'il a calculés avec les bits de contrôle reçus, en les additionnant bit par bit ;
- la combinaison (vecteur) obtenu représente le correcteur **s**.

On obtient un correcteur nul (tous les bits 0) seulement si le mot reçu est un mot-code, donc si :

- soit qu'on a reçu le mot-code qui a été transmis,
- soit que les erreurs intervenues ont transformé le mot-code transmis en un **autre mot-code**.

Le décodeur n'a pas les moyens de faire la distinction entre ces deux cas, donc il y a toujours une chance de 'louper' les erreurs. Autrement dit, on peut diminuer la probabilité d'erreur résiduelle, mais on ne peut pas l'annuler.

Pour la détection d'erreurs il suffit d'analyser le correcteur (nul ou pas), et demander le cas échéant la retransmission. Le décodeur ne détecte pas les erreurs seulement si le mot erreur est un mot-code (car dans ce cas-là le mot reçu est également mot-code). Il y a $\mathbf{2^m}$ telles possibilités (incluant l'erreur nulle), donc le **taux d'erreurs non détectables** vaut :

$$\mathbf{2^m : 2^n = 2^{m-n} = 2^{-r}}$$

La détection d'erreurs permet de contrôler une grande partie des erreurs possibles.

Pour la correction d'erreurs il faut pouvoir déterminer de quel mot-code provient le mot reçu. Ce qui implique qu'il doit exister une bijection (correspondance un à un) entre les correcteurs et les mots erreurs corrigibles. Etant donné que le correcteur a **r** bits, il y a $\mathbf{2^r}$ correcteurs différents, donc on ne peut corriger que $\mathbf{2^r}$ mots erreurs parmi les $\mathbf{2^n}$ possibles (l'erreur nulle est incluse). Le taux d'erreurs corrigibles vaut :

$$\mathbf{2^r : 2^n = 2^{r-n} = 2^{-m}}$$

et le **taux d'erreurs non corrigibles** vaut seulement $1 - \mathbf{2^{-m}}$. La correction d'erreurs ne permet de contrôler qu'une très petite partie des erreurs possibles.
Du point de vue du taux d'erreurs qui 's'échappent', il y a une grande différence entre les codes détecteurs et les codes correcteurs d'erreurs. Moralité ? Pourtant, comme on verra, les deux types des codes sont utilisés.

VERIFICATION DE PARITE

Le plus simple code détecteur d'erreurs est celui qui fait la vérification de parité. Aux m bits d'information on ajoute un seul bit de contrôle, donc n = m+1. Par exemple, pour m = 4, n = 5 :

$$\mathbf{c} = [\, c_1 \;\; c_2 \;\; c_3 \;\; c_4 \;\; c_5 \,] \quad \text{avec} \quad c_1 = c_2 + c_3 + c_4 + c_5$$

Le bit de contrôle c_1 est calculé de façon que le nombre des 1 dans le mot-code soit pair. Le décodeur reçoit le mot **c'**

$$\mathbf{c'} = [\, c_1' \;\; c_2' \;\; c_3' \;\; c_4' \;\; c_5' \,] = [\, c_1+\varepsilon_1 \;\; c_2+\varepsilon_2 \;\; \ldots \; c_5+\varepsilon_5 \,]$$

et calcule le correcteur **s** par une des deux méthodes équivalentes.

1. En vérifiant la parité du mot reçu :

$$\mathbf{s} = [c_1' + c_2' + c_3' + c_4' + c_5'] = [s_1]$$

2. En calculant le bit de contrôle c_1'' correspondant aux bits d'information reçus et en le comparant (par addition modulo-2) avec le bit de contrôle reçu c_1' :

$$c_1'' = c_2' + c_3' + c_4' + c_5'$$

$$\mathbf{s} = [c_1' + c_1''] = [c_1' + c_2' + c_3' + c_4' + c_5']$$

Suivant les deux méthodes on obtient le même correcteur qui vaut :

$$\mathbf{s} = [c_1' + c_2' + c_3' + c_4' + c_5'] = [c_1+\varepsilon_1 + c_2+\varepsilon_2 + c_3+\varepsilon_3 + c_4+\varepsilon_4 + c_5+\varepsilon_5] =$$

$$[c_1+c_2+c_3+c_4+c_5 + \varepsilon_1+\varepsilon_2+\varepsilon_3+\varepsilon_4+\varepsilon_5] = [0 + \varepsilon_1+\varepsilon_2+\varepsilon_3+\varepsilon_4+\varepsilon_5]$$

Donc le **correcteur est déterminé par le mot erreur** et il ne dépend pas du mot-code :

$$\mathbf{s} = [s_1] = [\varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 + \varepsilon_5]$$

Exercice

Soit le code ci-dessus. Donner tous les mots-code et déterminer la distance minimum du code.

Indiquer le type d'erreurs que le code peut détecter. Calculer le taux d'erreurs non détectées.
Calculer le correcteur pour 5 mots erreur de votre choix de poids 1, 2, 3, 4 et 5.

Solution :

LRC et VRC

Dans le cas où les données sont des caractères de 8 bits (octets) on fait souvent des vérifications de parité simples, comme LRC (Longitudinal Redundancy Check) et VRC (Vertical Redundancy Check).

Soit un mot-code contenant un nombre entier d'octets, suivis par 8 bits de contrôle :

$$c_{11} \dots c_{18}\, c_{21} \dots c_{28}\; c_{31} \dots c_{38} \dots\dots c_{lrc1} \dots c_{lrc8}$$

qu'on représente de la façon suivante :

c_{11}	c_{21}	c_{31}	…	c_{lrc1}
c_{12}	c_{22}	c_{32}	…	c_{lrc2}
c_{13}	c_{23}	c_{33}	…	c_{lrc3}
………				
c_{18}	c_{28}	c_{38}	…	c_{lrc8}

Les bits de contrôle sont calculés conformément à la relation :

$$c_{lrcj} = c_{1j} + c_{2j} + c_{3j} + \dots$$

donc ils effectuent une vérification de parité 'longitudinale' - LRC. Un tel code détecte toutes les erreurs impaires sur les bits j des octets, pour j de 1 à 8.

Si chaque caractère comprend seulement 7 bits d'information, on peut ajouter un bit de vérification de parité par octet, représentant le bit VRC.

c_{11}	c_{21}	c_{31}	…
c_{12}	c_{22}	c_{32}	…
c_{13}	c_{23}	c_{33}	…
………			
c_{17}	c_{27}	c_{37}	…
c_{1vrc}	c_{2vrc}	c_{3vrc}	…

Les bits de contrôle sont calculés conformément à la relation :

$$c_{ivrc} = c_{i1} + c_{i2} + c_{i3} + \ldots + c_{i7}$$

donc ils effectuent une vérification de parité 'verticale' - VRC. Un tel code détecte toutes les erreurs impaires dans chaque octet.

Il y a des codes qui font les deux types de vérification de parité - LRC et VRC (vérification de parité croisée).

$$\begin{matrix} c_{11} & c_{21} & c_{31} & \ldots & c_{lrc1} \\ c_{12} & c_{22} & c_{32} & \ldots & c_{lrc2} \\ c_{13} & c_{23} & c_{33} & \ldots & c_{lrc3} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ c_{1vrc} & c_{2vrc} & c_{3vrc} & \ldots & c_{lrc8} \end{matrix}$$

Dans ce cas le huitième bit du LRC vérifie la parité de tous les bits VRC :

$$c_{lrc8} = c_{1vrc} + c_{2vrc} + c_{3vrc} + \ldots$$

Ces codes permettent de corriger une erreur et de détecter toutes les autres combinaisons d'erreurs impaires sur les bits j des octets, pour j de 1 à 8, et des erreurs impaires dans chaque octet.

Exemple

Soit le code à vérification de parité croisée :

```
0 1 0 1 0 | 0
1 0 1 1 0 | 1
1 1 1 0 1 | 0
0 1 0 1 0 | 0
1 1 0 0 0 | 0
1 0 1 0 1 | 1
0 1 1 1 1 | 0
---------- | --
0 1 0 0 1 | 0
```

La séquence émise :

011011001011101101100110110100100010011101000100

Soit la séquence reçue ayant une erreur :

0110110010111011011**1**011011010010001001110100 0100

Les vérifications de parité effectuées par le décodeur indiquent une ligne erronée et une colonne erronée.

0 1 0 1 0 0
1 0 1 1 0 1
1 1 1 0 1 0
0 1 **1** 1 0 0 ← ligne 4 erronée
1 1 0 0 0 0
1 0 1 0 1 1
0 1 1 1 1 0
0 1 0 0 1 0
↑
colonne 3 erronée

Le décodeur corrige l'erreur en inversant (1→0) le bit qui se trouve à l'intersection de la ligne 4 avec la colonne 3.

Exercice

Pour le code de l'exemple ci-dessus analyser les résultats des vérifications de parité effectuées par le décodeur dans le cas de différentes combinaisons de deux erreurs.

MATRICE DE CONTROLE

La matrice de contrôle **H** permet de calculer côté codeur, à partir des bits d'information, les bits de contrôle correspondants, et côté décodeur, à partir du mot reçu, le correcteur.

Etant donné qu'il y a m bits de contrôle, leur calcul nécessite un système de m équations. On définit la matrice de contrôle de dimension r×n (r lignes et n colonnes) :

$$\mathbf{H}=\begin{bmatrix} h_{11} & \ldots\ldots & h_{1n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots \\ h_{r1} & \ldots\ldots & h_{rn} \end{bmatrix}$$

Le système de m équations permettant de calculer les bits de contrôle est :

$$\begin{bmatrix} h_{11} & \ldots\ldots & h_{1n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots \\ h_{r1} & \ldots\ldots & h_{rn} \end{bmatrix} \times \begin{bmatrix} c_1 \\ . \\ . \\ . \\ c_n \end{bmatrix} = \mathbf{0}$$

ou

$$\mathbf{H} \times \mathbf{c}^T = \mathbf{0}$$

Sous forme explicite, le système de m équations est :

$$h_{11}c_1 + \ldots + h_{1n}c_n = 0$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$h_{r1}c_1 + \ldots + h_{rn}c_n = 0$$

Le décodeur calcule le correcteur **s** à l'aide du système de m équations :

$$\mathbf{s} = \begin{bmatrix} h_{11} & \ldots\ldots & h_{1n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots \\ h_{r1} & \ldots\ldots & h_{rn} \end{bmatrix} \times \begin{bmatrix} c_1' \\ . \\ . \\ . \\ c_n' \end{bmatrix} = \begin{bmatrix} s_1 \\ : \\ s_m \end{bmatrix}$$

ou

$$\mathbf{s} = \mathbf{H} \times \mathbf{c'}^{T}$$

et sous forme explicite, le système de m équations est :

$$s_1 = h_{11}c_1' + \ldots + h_{1n}\, c_n'$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$s_m = h_{r1}c_1' + \ldots + h_{rn}\, c_n'$$

Etant donné que $\mathbf{c'} = \mathbf{c} + \boldsymbol{\varepsilon}$

$$\mathbf{s} = \mathbf{H} \times \mathbf{c'}^{T} = \mathbf{H} \times (\mathbf{c} + \boldsymbol{\varepsilon})^{T} = \mathbf{H} \times \boldsymbol{\varepsilon}^{T}$$

et, comme attendu, le **correcteur est déterminé par le mot erreur** et il ne dépend pas du mot-code.

Exercice

Montrer que si $\mathbf{c_a}$ et $\mathbf{c_b}$ sont des mots-code, $\mathbf{c_a} + \mathbf{c_b}$ est aussi un mot-code.

Solution

MATRICE GENERATRICE

On peut définir également la matrice génératrice **G** d'un code. C'est une matrice de dimension m×n, orthogonale par rapport à la matrice de contrôle **H**. Ses lignes représentent des mots-code indépendants (aucune ligne de la matrice n'est une combinaison linéaire d'autres lignes).

Il est évident (pourquoi ?) que :

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

Une façon de déterminer la matrice génératrice **G** est de choisir ses lignes parmi les mots-code ayant chacun un seul bit 1 sur les k premières positions, les 1 étant décalés d'une ligne à l'autre :

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & \ldots & 0 & \ldots\ldots\ldots \\ 0 & 1 & \ldots & 0 & \ldots\ldots\ldots \\ \multicolumn{5}{c}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\ 0 & 0 & \ldots & 1 & \ldots\ldots\ldots \end{bmatrix}$$

1 2 m n

Les lignes de cette matrice sont, évidemment, indépendantes. Tous les 2^m mots-codes représentent des combinaisons linéaires des lignes de la matrice génératrice.

CODES SYSTEMATIQUES

Dans les codes systématiques les bits de contrôle se trouvent groupés sur les positions 1 … r et les bits d'information se trouvent groupés sur les positions r+1 … n. La séparation des bits d'information reçus d'un code systématique est très aisée.

$$\mathbf{c} = [\, c_1 \;\; c_2 \;\; \ldots \; c_m \; c_{m+1} \;\; c_{m+2} \;\; \ldots \; c_n \,]$$

$$|\ldots \text{contrôle} \ldots| \quad \ldots \text{information} \ldots|$$

r n-r = m

Dans les codes non systématiques les bits d'information et de contrôle ne sont pas groupés.

CODE HAMMING CORRECTEUR D'UNE ERREUR

Le code Hamming non systématique correcteur d'une erreur est un code ayant deux particularités :

- les bits de contrôle se trouvent sur les positions 2^i (2^0, 2^1, 2^2, etc) ;
- les colonnes de la matrice **H** représentent en binaire les indices des respectives colonnes.

Ces deux propriétés ont les conséquences suivantes :

- le calcul des bits de contrôle est très simple, chacun dépendant seulement des bits d'information ;
- le correcteur représente en binaire la position erronée.

La distance minimum doit satisfaire la condition :

$$D_{min} = 2e + 1 = 3$$

Pour la correction d'une erreur, le nombre des correcteurs différents doit dépasser de 1 le nombre des différents mots-erreur corrigibles :

$$2^r \geq n + 1$$

Pourquoi ?

Exemple

Soit $m = 4 \rightarrow 2^r \geq 4 + r + 1 \rightarrow 2^3 = 4 + 3 + 1 \rightarrow r = 3,\ n = 7.$

$\mathbf{c} = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7]$ avec les bits de contrôle c_1, c_2, c_4.

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

Codage :

$$\mathbf{H} \times \mathbf{c}^T = \mathbf{0}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \mathbf{0}$$

$c_4 + c_5 + c_6 + c_7 = 0 \rightarrow c_4 = c_5 + c_6 + c_7$

$c_2 + c_3 + c_6 + c_7 = 0 \rightarrow c_2 = c_3 + c_6 + c_7$

$c_1 + c_3 + c_5 + c_7 = 0 \rightarrow c_1 = c_3 + c_5 + c_7$

Soit : $c_7 = 1,\ c_6 = 1,\ c_5 = 0,\ c_3 = 0.$

On obtient les bits de contrôle : $c_4 = 0,\ c_2 = 0,\ c_1 = 1$

et le mot-code :

$\mathbf{c} = [1\ 0\ 0\ 0\ 0\ 1\ 1]$

Décodage :

$$\mathbf{s} = \mathbf{H} \times \mathbf{c'}^{T}$$

$$\mathbf{s}=\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c_1' \\ c_2' \\ c_3' \\ c_4' \\ c_5' \\ c_6' \\ c_7' \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

$$s_1 = c_4' + c_5' + c_6' + c_7'$$

$$s_2 = c_2' + c_3' + c_6' + c_7'$$

$$s_3 = c_1' + c_3' + c_5' + c_7'$$

Si le mot erreur a un seul bit 1, soit $\varepsilon_i = 1$, comme

$$\mathbf{s} = \mathbf{H} \times \mathbf{c'}^{T} = \mathbf{H} \times (\mathbf{c} + \boldsymbol{\varepsilon})^{T} = \mathbf{H} \times \boldsymbol{\varepsilon}^{T}$$

on obtient :

$$\mathbf{s} = \begin{bmatrix} h_{i1} \\ h_{i2} \\ h_{i3} \end{bmatrix}$$

donc le correcteur vaut la colonne i et il indique en binaire la position erronée.

Soit le mot reçu : $\mathbf{c'} = [1\ 0\ 1\ 0\ 0\ 1\ 1]$

On calcule le correcteur : $s_1 = 0$, $s_2 = 1$, $s_3 = 1$

$$\mathbf{s} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Donc c'est le bit sur la position 3 qui est erroné. Le mot corrigé est :

$$\mathbf{c} = [1\ 0\ \mathbf{0}\ 0\ 0\ 1\ 1]$$

Exercice

Donner le schéma du codeur et du décodeur du code Hamming ci-dessus, en tenant compte du fait que les mots sont transmis en série en commençant par c_7.

Exercice

Soit le code Hamming non systématique correcteur d'une erreur de l'exemple précédent (m = 4 et r = 3).

1. Donner tous les mots-code et déterminer la distance minimum du code.

2. Compléter la matrice génératrice du code :

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

3. Soit le mot reçu **c'** = [1011011]. Calculer le correcteur **s**, indiquer la position erronée et corriger l'erreur.

4. On transmet le mot-code **c** = [1111111], dont les bits se trouvant sur les positions 4 et 5 seront erronés pendant la transmission. Déterminer le mot reçu **c'**, le correcteur **s**, et le mot corrigé en conformité avec le correcteur. Comparer le mot corrigé avec le mot-code d'origine. Interpréter le résultat de la comparaison. Quelle est la conclusion ?

Solution

CODE HAMMING CORRECTEUR D'UNE ERREUR ET DETECTEUR DE DEUX ERREURS

On ajoute un bit de contrôle c_0 sur la position 0. Dans ce cas, pour m = 4, r = 4 et n = 8 et le mot-code a 8 bits :

$$\mathbf{c} = [c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7]$$

On ajoute une ligne à la matrice **H**, correspondant à la vérification de parité de tous les bits :

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

Codage

A partir de $\mathbf{H} \times \mathbf{c}^T = \mathbf{0}$ on obtient pour les bits de contrôle c_4, c_2 et c_1 les mêmes expressions que dans le cas du code Hamming correcteur d'une erreur, et pour c_0 on obtient :

$$c_0 = c_1 + c_2 + c_3 + c_4 + c_5 + c_6 + c_7$$

donc c_0 est en effet un bit de parité qui permet au décodeur de faire la distinction entre une et deux erreurs.

Décodage

A partir de $\mathbf{s} = \mathbf{H} \times \mathbf{c'}^T$ on obtient un correcteur à quatre bits ; $s_1 \ldots s_3$ ont les mêmes expressions que dans le cas du code Hamming correcteur d'une erreur, et pour s_4 on obtient :

$$s_4 = c_0' + c_1' + c_2' + c_3' + c_4' + c_5' + c_6' + c_7'$$

$s_1\ s_2\ s_3$	s_4	décision
0	0	pas d'erreur
$\neq$ **0**	1	1 erreur, correction cf. $s_1\ s_2\ s_3$
$\neq$ **0**	0	2 erreurs, demande retransmission
0	1	c_0' est erroné

Exercice

Soit le code Hamming correcteur d'une erreur et détecteur de deux erreurs, avec m = 4 et r = 4. Soit les mots reçus :

$\mathbf{c_a}' = [1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1]$

$\mathbf{c_b}' = [1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1]$

$\mathbf{c_c}' = [0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0]$

$\mathbf{c_d}' = [1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0]$

Calculer les correcteurs correspondants et indiquer la décision du décodeur.

Solution

CODES CYCLIQUES

Les codes cycliques sont des codes en bloc ayant la propriété suivante : les mots obtenus par permutations cycliques des bits d'un mot-code sont, eux aussi des mots-code. Par exemple, si [1 0 0 1 0 1 1] est mot-code, [0 0 1 0 1 1 1] l'est aussi.

Il est convenable de représenter les mots d'un code cyclique sous forme des polynômes à coefficients binaires, ces derniers représentant les bits des mots. Par exemple un mot de n bits est représenté par un polynôme de degré n-1 :

$$c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + c_{n-3}x^{n-3} + \ldots + c_1x + c_0$$

Une permutation cyclique de i positions à gauche correspond à une multiplication par x^i, à condition que :

$$x^n + 1 = 0 \quad \text{ou} \quad x^n = 1$$

Par exemple pour i = 2 on a :

$$x^2c(x) = c_{n-3}x^{n-1} + c_{n-4}x^{n-2} + c_{n-5}x^{n-3} + \ldots + c_{n-1}x + c_{n-2}$$

et $x^2c(x)$ appartient aussi au code.

Un code cyclique avec **m** bits d'information et **r** bits de contrôle est généré par son **polynôme générateur g(x)** de degré **r** :

$$g(x) = x^r + g_{r-1}x^{r-1} + \ldots + g_1x + 1$$

Le polynôme générateur doit être primitif, c'est à dire il doit satisfaire les conditions :

- être diviseur de x^n+1, avec $n = 2^r - 1$, mais ne pas être diviseur de x^j+1 avec $j < n$;
- être irréductible.

Tous les polynômes générateurs ont $g_r = g_0 = 1$. Pourquoi ?

Tous les mots d'un code cyclique sont des multiples du polynôme générateur g(x) du code !

Par contre, tous les polynômes de degré ≤ r-1 qui ne sont pas multiples de g(x) n'appartiennent pas au code.
On peut définir le polynôme de contrôle h(x) d'un code cyclique :

$$h(x) = \frac{x^n + 1}{g(x)}$$

et comme

$$x^n + 1 = 0$$

on a :

$$h(x) \times g(x) = 0$$

Comme pour tout code bloc, on peut définir les matrice génératrice **G** (m×n) et de contrôle **H** (r×n) d'un code cyclique, matrices qu'on écrit à partir de g(x), respectivement de h(x), mais elles sont peu utilisées. La relation ci-dessus correspond à celle vue précédemment pour les codes bloc :

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

Exemple

Soit : m = 4, r = 3
$g(x) = x^3 + x^2 + 1$
$h(x) = x^7 + 1 \,/\, x^3 + x^2 + 1 = x^4 + x^3 + x^2 + 1$

$$\mathbf{G} = \begin{bmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ x^3g(x) \end{bmatrix} = \begin{bmatrix} 0001101 \\ 0011010 \\ 0110100 \\ 1101000 \end{bmatrix}$$

Les lignes de la matrice H représentent les polynômes h(x), xh(x), x^2h(x) écrits de gauche à droite, en commençant par le poids le plus faible :

$$\mathbf{H} = \begin{bmatrix} h(x) \\ xh(x) \\ x^2h(x) \end{bmatrix} = \begin{bmatrix} 1011100 \\ 0101110 \\ 0010111 \end{bmatrix}$$

Vérifiez que $\mathbf{H} \times \mathbf{G}^T = \mathbf{0}$.

CODAGE

Soit le polynôme i(x) correspondant aux m bits d'information :

$$i(x) = c_{n-1}x^{m-1} + c_{n-2}x^{m-2} + \ldots + c_{n-m+1}x + c_{n-m}$$

et soit le polynôme générateur g(x) de degré r :

$$g(x) = x^r + g_{r-1}x^{r-1} + \ldots + g_1x + 1$$

A partir de i(x) et g(x) on doit générer les mots-code, tous multiples de g(x).

La façon la plus simple n'est pas la meilleure : si on multiplie i(x) par g(x) on obtient, certes, c(x) multiple de g(x), mais on ne retrouve pas parmi ses coefficients les bits d'information explicites.

On peut faire mieux et générer un code systématique, dont les m bits d'information, groupés, sont les coefficients c_{n-1} à c_{n-m} et les r bits de contrôle, aussi groupés – les coefficients $c_{n-m-1}= c_{r-1}$ à c_0.

Voilà la méthode : on multiplie i(x) par $x^r = x^{n-m}$ et on ajoute au produit $i(x)\cdot x^r$ le reste r(x) de la division du même produit par g(x). En effet on peut écrire :

$$i(x)\cdot x^r = c_{n-1}x^{n-1} + \ldots + c_{n-m}x^{n-m}$$

$$i(x)\cdot x^r = q(x)\cdot g(x) + r(x) \qquad \text{où le degré de } r(x) < r$$

$$r(x) = c_{r-1}x^{r-1} + \ldots + c_0 = c_{n-m-1}\, x^{n-m-1} + \ldots + c_0$$

$$i(x)\cdot x^r + r(x) = q(x)\cdot g(x) \qquad \text{multiple de } g(x)$$

$$\mathbf{c(x) = i(x)\cdot x^r + r(x)}$$

C'est évident que c(x) est multiple de g(x). Suite à la multiplication, on retrouve les bits d'information comme coefficients de x^{n-1} … x^{n-m}, et les bits de contrôle (les coefficients du reste) comme les coefficients de x^{n-m-1} … x^0. Donc le mot-code a la structure suivante :

$$c(x) = c_{n-1}x^{n-1} + \ldots + c_{n-m}x^{n-m} + c_{n-m-1}x^{n-m-1} + \ldots + c_0$$

avec les bits d'information c_{n-1} … c_{n-m} et les bits de contrôle c_{n-m-1} … c_0.

DECODAGE

La première opération que le décodeur doit effectuer c'est le calcul du correcteur s(x) à partir du mot reçu c'(x).

$$c'(x) = c(x) + \varepsilon(x) = i'(x)\cdot x^r + r'(x)$$

Il y a deux méthodes équivalentes pour calculer le correcteur.

1. Calculer le reste de la division du mot reçu par g(x) :

$$s(x) = \text{reste } [c'(x)/g(x)]$$

2. Calculer les bits de contrôle correspondant aux bits d'information reçus, r''(x) et les additionner aux bits de contrôle reçus r'(x) :

$$r''(x) = \text{reste } [i'(x)\cdot x^r/g(x)]$$

$$s(x) = r'(x) + r''(x)$$

On peut facilement montrer que les deux méthodes mènent au même résultat.

En divisant c'(x) par g(x) on obtient :

$$s(x) = \text{reste } [c'(x)/g(x)] = \text{reste } [(i'(x)\cdot x^r + r'(x))/g(x)] =$$

$$= \text{reste } [i'(x)\cdot x^r/g(x)] + \text{reste } [r'(x)/g(x)] =$$

$$= \text{reste } [i'(x)\cdot x^r/g(x)] + r'(x) = r''(x) + r'(x)$$

En pratique on préfère la deuxième méthode, parce qu'elle permet d'utiliser la même structure pour le codeur et pour le décodeur.

La deuxième opération que le décodeur doit faire dépend du type de code.

Si le code est seulement détecteur d'erreurs, le décodeur prend la décision mot erroné / mot correct et demande ou pas la retransmission.

Si le code est correcteur d'erreurs, il faut que le décodeur fasse le passage du correcteur calculé à l'erreur corrigible. Il y a plusieurs possibilités : soit que le tableau des correspondances est mémorisé dans un ROM, soit qu'on détermine en temps réel l'erreur correspondant au correcteur calculé. Dans les deux cas la correction se fait en additionnant modulo-2 le mot reçu avec le mot erreur correspondant.