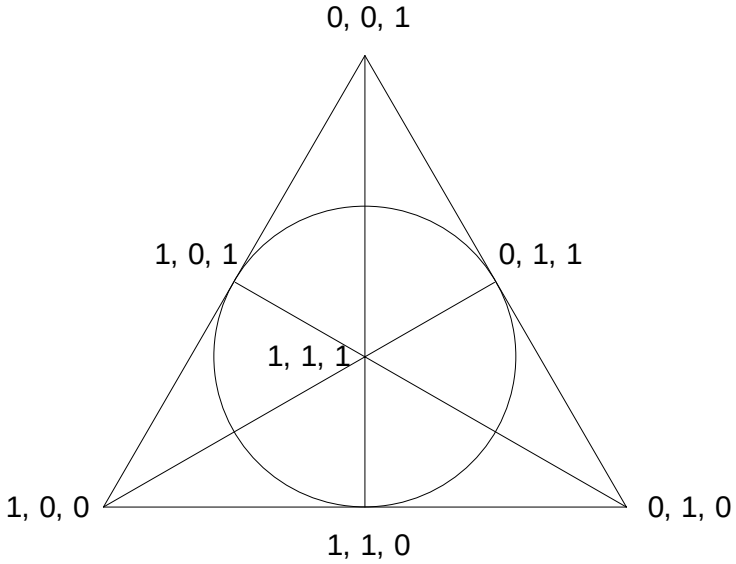
Atelier Math.en.jeans

Sujet : « Les codes correcteurs »

Codage linéaire de canal

Comment l'algèbre linéaire sur les corps finis permet-elle d'effectuer un codage et de détecter les erreurs de manière efficace en pratique ?



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

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

Elèves :

Amandine GOUMARD, Louciné SARKHANOVA – Seconde.
Adrien CHAIGNE, Benoît GOUPILLEAU, Artak SARKHANOV – Première S.
Amélie CHAMBET, Sylvain FAVRE, Elodie MALEPLATE, Damien PETIT – Terminale S.

Enseignants : Maryse COMBRADE, Cédric JOSSIER.

Chercheur : Lancelot PECQUET – Université de Poitiers (UMR CNRS 6086)

Lycée de l'image et du son - 303 avenue de Navarre – 16022 ANGOULEME CEDEX

Année scolaire 2004 – 2005

Sommaire.

I – Déroulement et objectifs de l'atelier.

L'objectif principal de cet atelier est de permettre à des élèves du Lycée d'être confrontés à une situation de recherche. Ceci ayant plusieurs objectifs pédagogiques :

- Aider les élèves à structurer une recherche de documents.
- Aider les élèves à trier les informations (utiles et inutiles).
- Faire travailler les élèves en groupes.
- Développer l'autonomie dans l'organisation du travail.

Pour ce faire, un chercheur de l'Université de Poitiers est venu présenter le **18 octobre 2004** aux élèves du Lycée un sujet de recherche portant sur les codes correcteurs. Son rôle a consisté à guider et à réorienter les élèves tout au long de l'année. Le relais a été passé aux enseignants de l'établissement, à raison d'une à deux heures hebdomadaires. Le rôle de l'enseignant a consisté à aider les élèves à comprendre les documents trouvés sur internet ou au C.D.I. Il a également donné un coup de pouce pour la présentation des différents documents. Le rôle des élèves étant alors celui de chercheurs ne savant pas à l'avance dans quelle direction se tourner et ils ont dû construire eux-mêmes les outils dont ils avaient besoin pour répondre à la problématique.

Un autre objectif de l'atelier est de faire rencontrer des élèves de plusieurs établissements et de travailler en groupe, à distance, sur des sujets parallèles et liés. Des exposés similaires ont été réalisés au Lycée du bois d'Amour de Poitiers et au Lycée Saint Joseph de Bressuire. Il a fallu que les élèves se rencontrent plusieurs fois pour mettre en commun le fruit de leurs découvertes.

La première rencontre a eu lieu au **Lycée du bois d'Amour de Poitiers le 4 janvier 2005**. La seconde fut mise en place au **laboratoire de mathématiques de l'Université de Poitiers le 22 février 2005**. Elle fut l'occasion d'une visite du laboratoire de recherche.

Le travail ainsi effectué a été exposé pendant **le congrès national MATh.en.JEANS à l'université de Jussieu à Paris les 1, 2 et 3 avril 2005**.

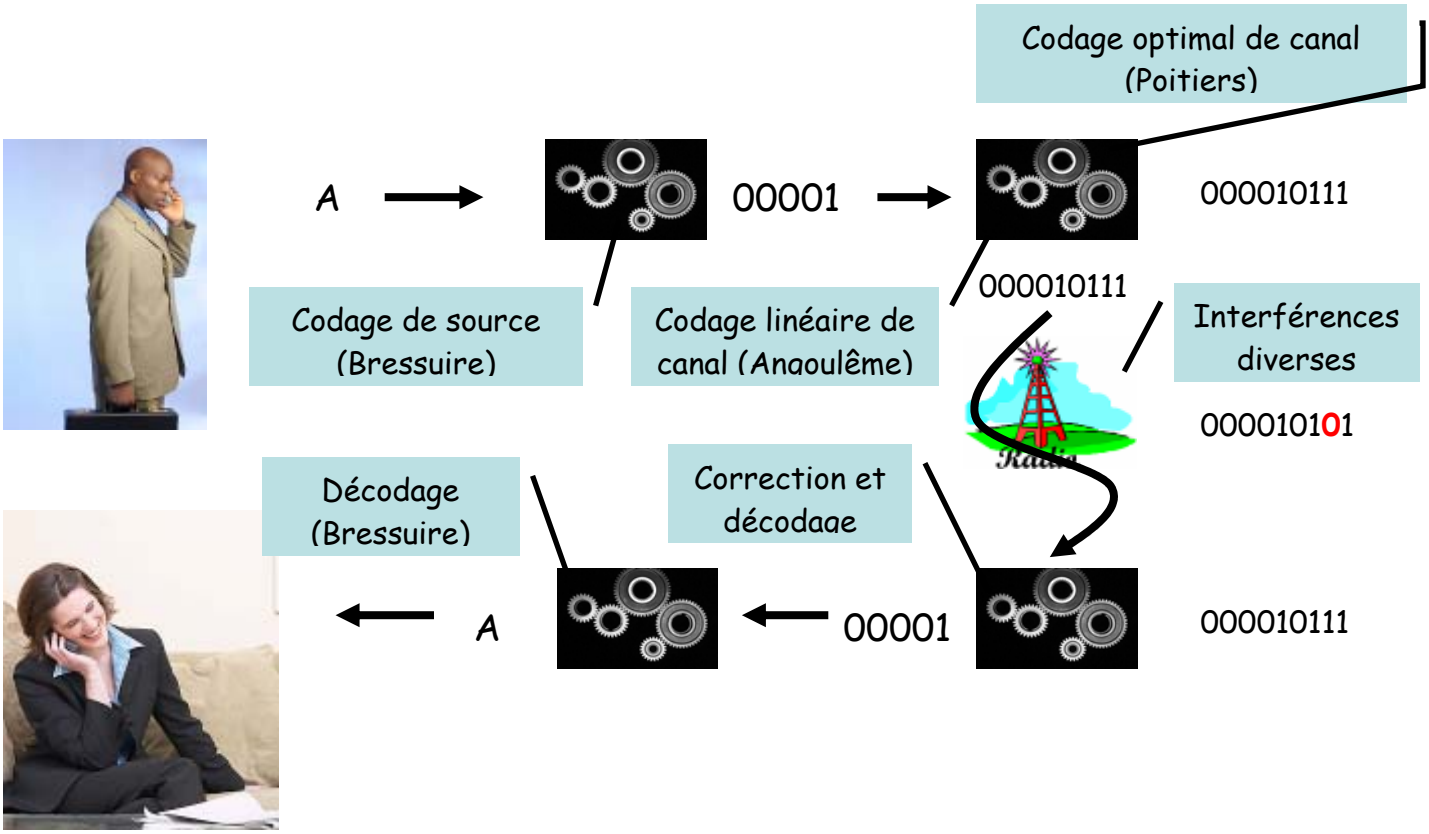
Ce compte rendu sera publié de manière nationale par l'association MATh.en.JEANS.

C. JOSSIER

II – Présentation du problème.

La problématique, « comment l'algèbre linéaire sur les corps finis permet-elle d'effectuer un codage et de détecter les erreurs de manière efficace en pratique ? », est plutôt vague. Pourtant, le codage et le décodage interviennent très souvent dans la vie quotidienne sans que nous nous en rendions compte : lors d'envoi d'un sms, d'un mail, lorsqu' une donnée nous provient d'un satellite… Si dans la problématique nous parlons des corps finis c'est parce que toutes les données sont codées en 0 et 1.

Dans l'exposé qui suivra lorsque nous parlerons de mots ce seront des mots binaires (composés de 0 et de 1) et les lettres qui composeront ce mot seront binaires elles aussi (une lettre vaut 0 ou 1).



Nous étions trois lycées à travailler sur les codes correcteurs d'erreur : le lycée Saint Joseph de Bressuire, le lycée du Bois d'Amour de Poitiers et notre lycée, celui de l'Image et du Son d'Angoulême. Le lycée de Bressuire travaillait sur la transformation d'un mot en langage binaire puis nous intervenions pour rallonger le mot binaire (de façon à détecter et corriger des erreurs éventuelles lors de l'envoi du message) et le lycée de Poitiers essayait de voir le rallongement optimal, l'allongement ne doit pas être trop grand, mais assez tout de même pour pouvoir corriger un maximum d'erreurs, afin que l'envoi ne nécessite pas de calcul trop long à effectuer car ceci prend du temps. Après, nous avons vu que nous pouvions corriger une erreur intervenue lors d'une interférence, puis nous décodions le mot rallongé en mot binaire plus court et le lycée de Bressuire décodait le mot binaire et le mot de départ était retrouvé !

Maintenant, vous allez voir en détail notre travail.

E. MALEPLATE

III – Corps fini F_2.

La suite de notre travail sera liée à des échanges d'informations par l'intermédiaire de canaux de transmission. Ces échanges ne sont traduits de manière mathématique que par deux valeurs 0 et 1. Nous aurons besoin tout au long de cet exposé de faire appel à un nouveau corps appelé corps fini à 2 éléments et noté F_2 qui sera composé des deux seuls éléments : 0 et 1.

On munit ce corps des opérations suivantes :

- Une addition (et donc une soustraction).
- Une multiplication (et donc une division).

Les règles de calcul dans ce corps à deux éléments sont les mêmes que dans le corps des réels saufs les deux cas grisés qui nous seront utiles pour effectuer les calculs par la suite :

$0 + 0 = 0$	$0 + 1 = 1$	$1 + 0 = 1$	**$1 + 1 = 0$**
$0 - 0 = 0$	**$0 - 1 = 1$**	$1 - 0 = 1$	$1 - 1 = 0$
$0 \times 0 = 0$	$0 \times 1 = 0$	$1 \times 0 = 0$	$1 \times 1 = 1$
$\frac{0}{0}$: indéfini	$\frac{0}{1} = 0$	$\frac{1}{0}$: indéfini	$\frac{1}{1} = 1$

Nous aurons également besoin de quelques notions concernant les matrices et les calculs qui peuvent être réalisés à l'aide de celles-ci.

A. GOUMARD

IV - Notion de matrice.

Une matrice est un tableau de valeurs composé d'un nombre n de lignes et p de colonnes. Son format est alors noté $n \times p$.

Exemple :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$: matrice 5×3

Nous pouvons définir plusieurs opérations à l'aide de matrices :

- La multiplication d'un élément du corps de référence (ici F_2) par une matrice.
- L'addition de deux matrices de **même format**.
- La multiplication d'une matrice **n** × **p** par une matrice **n'** × **p'** à condition que **p** = **n'**

Pour notre projet nous n'utiliserons qu'une de ces opérations : la multiplication d'une matrice $n \times p$ par un vecteur colonne $p \times 1$, nous obtiendrons un vecteur colonne $n \times 1$

Voici comment une telle multiplication s'effectue :

On prend l'exemple d'une multiplication d'une matrice 5×3 par un vecteur 3×1

Pour obtenir le premier élément du vecteur colonne 5×1, on multiplie le premier élément de la première ligne de la matrice par le premier élément du vecteur colonne 3×1. on ajoute à cela le produit du deuxième élément de la première ligne de la matrice par le

deuxième élément du vecteur colonne. On ajoute enfin le produit du troisième élément de la première ligne de la matrice par le troisième élément du vecteur colonne.

On procède de même avec les lignes suivantes de la matrices pour obtenir les 4 autres coordonnées du vecteur colonne 5×1 :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \times y_1 + 0 \times y_2 + 0 \times y_3 \\ 0 \times y_1 + 1 \times y_2 + 0 \times y_3 \\ 1 \times y_1 + 1 \times y_2 + 0 \times y_3 \\ 1 \times y_1 + 0 \times y_2 + 1 \times y_3 \\ 1 \times y_1 + 1 \times y_2 + 1 \times y_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_1+y_2 \\ y_1+y_3 \\ y_1+y_2+y_3 \end{pmatrix}$$

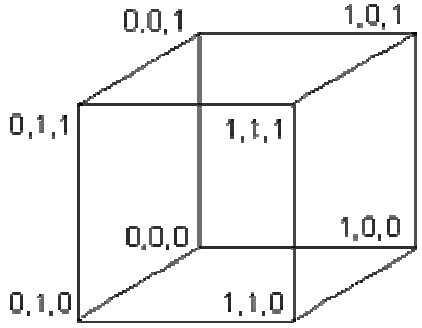
Ceci nous permettra de coder, décoder et corriger les informations de manière calculatoire. Traitons un exemple pour comprendre comment fabriquer un code correcteur.

L. SARKHANOVA

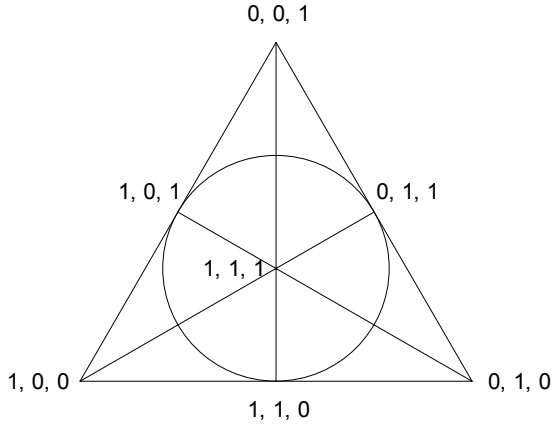
V – Un exemple de codage linéaire : [7,3,3] – code.

Après avoir découvert les notions matricielles utiles pour la suite, nous allons vous présenter un exemple de codage linéaire : le [7, 3, 3] – code. Il se nomme ainsi car nous partons d'un mot de 3 lettres (deuxième valeur) que nous allons coder en un mot de 7 lettres (première valeur) avec une distance minimale de 3 (dernière valeur) entre chaque mot de 7 lettres. Voir l'ANNEXE 1 pour les détails concernant la distance minimale d'un code.

Considérons tout d'abord le corps $(F_2)^3$ où se trouvent nos mots de 3 lettres. Il y a donc $2^3 = 8$ mots différents. On peut représenter ces 8 mots de la façon suivante :



On s'aperçoit que sur 8 plans constitués chacun de trois sommets du cube, la somme des coordonnées de 2 de ces points donne les coordonnées du 3ème point de ce plan. Cette particularité nous permet d'établir le schéma suivant :



Ce schéma conserve les propriétés du cube : sur une même ligne (le cercle inscrit également), la somme des coordonnées de deux points appartenants à la ligne donne les coordonnées du 3ème point de cette ligne.

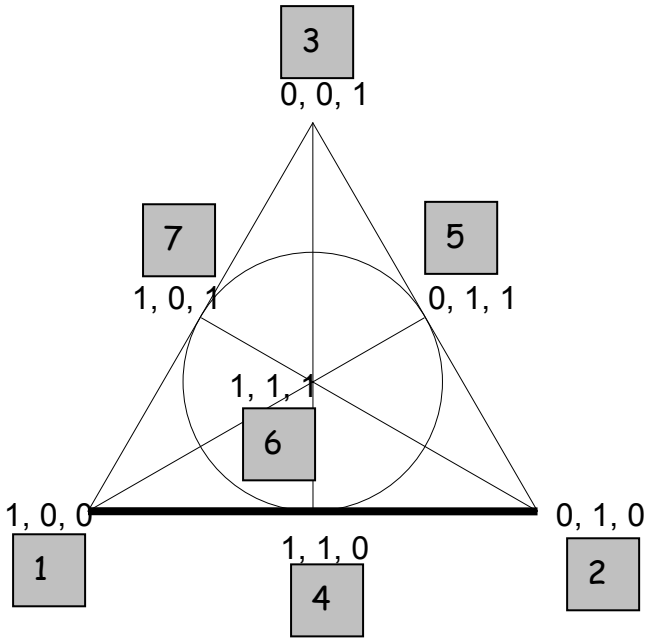
Prenons un exemple :

Considérons le cercle :

$$\begin{pmatrix}1\\0\\1\end{pmatrix} + \begin{pmatrix}0\\1\\1\end{pmatrix} = \begin{pmatrix}1+0\\0+1\\1+1\end{pmatrix} = \begin{pmatrix}1\\1\\0\end{pmatrix}$$

S. FAVRE

Numérotons chaque sommet de la figure.



A partir de cette notation, nous allons créer 7 mots de 7 lettres binaires (0 ou 1). Un mot sera représenté par $x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7$.

x_i sera égal à 1 si la droite passe par ce sommet et à 0 si la droite n'y passe pas.

Pour comprendre plus facilement, prenons un exemple (droite en gras sur le dessin) :

11O1OO**O**

Nous passons par le sommet 2
donc x_2 = 1

Nous ne passons pas par le sommet 7
donc x_7 = 0

De même, pour toutes les droites ainsi que pour le **cercle**. Nous obtenons les mots :

1101000 ; 0110100 ; 0011010 ; 0001101 ; 1000110 ; 0100011 ; 1010001

A. CHAMBET

Nous allons compléter ces mots et former au total 16 mots de 7 lettres en procédant de la manière suivante :

- Ajoutons aux 7 mots ci-dessus le mot 0000000
- Prenons tous les mots « inverses » : pour chaque mot, transformons chaque 1 en 0 et chaque 0 en 1.

Nous obtenons alors les 16 mots de 7 lettres suivants :

0000000	1111111
1101000	0010111
0110100	1001011
0011010	1100101
0001101	1110010
1000110	0111001
0100011	1011100
1010001	0101110

Ces Mots ont une particularité intéressante pour la suite de notre exposé : ils ont tous au moins une distance de Hamming de 3, la distance minimale d vaut 3, on pourra donc détecter et corriger :

$$e = \frac{d - 1}{2} = \frac{3 - 1}{2} = 1 \text{ erreur}$$

Voir l'ANNEXE 1 pour les informations concernant la distance de Hamming et la distance minimale d'un code.

Essayons de voir comment l'on peut s'y prendre.

A. GOUMARD

VI – Matrice génératrice du [7,3,3] – code.

Nous sommes dans un ensembles de mots de 7 lettres (7 coordonnées) qui sont à une distance de Hamming d'au moins 3 (d = 3) formés à partir de mots initiaux de 3 lettres (3 coordonnées).

Soit $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ un mot initial que l'on veut transformer en un mot de 7 coordonnées :

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}$$

On va donc chercher une matrice permettant de transformer les mots de 3 lettres en des mots de 7 lettres. Cette matrice est appelée matrice géneratrice. On la multiplie par le vecteur colonne initial (mot de 3 lettres) et on obtient le vecteur colonne final de 7 coordonnées (mot de 7 lettres), la matrice recherchée aura donc 3 lignes et 7 colonnes.

Pour faciliter le décodage (voir la fin de la partie sur la matrice de parité du [7,3,3] – code), on pose $x_1 = y_1$; $x_2 = y_2$ et $x_3 = y_3$. On peut déjà écrire le début de la matrice génératrice, en effet, comme on la vu dans la partie concernant les matrices :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \times y_1 + 0 \times y_2 + 0 \times y_3 \\ 0 \times y_1 + 1 \times y_2 + 0 \times y_3 \\ 0 \times y_1 + 0 \times y_2 + 1 \times y_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Les autres coordonnées : x_4 ; x_5 ; x_6 et x_7 seront exprimées en fonction de y_1 , y_2 et y_3. Les équations linéaires qui restent à notre disposition sont : $y_1 + y_2$; $y_1 + y_3$; $y_2 + y_3$ et $y_1 + y_2 + y_3$. Pour trouver le bon ordre, il faut essayer de trouver, parmi les mots de 7 lettres qui sont à notre disposition, 8 mots dont les 4 dernières coordonnées vérifient toutes les mêmes équations. Utilisons le tableau suivant :

3 lettres	7 lettres	Vérification
000	0000000	0+0 = 0
100	1001011	1+0 = 1
010	0101110	0+1 = 1
001	0010111	0+0 = 0
110	1100101	1+1 = 0
101	1011100	1+0 = 1
011	0111001	0+1 = 1
111	1110010	1+1 = 0

On remarque que la somme binaire des 2 premières coordonnées des mots de 3 lettres correspond toujours à la 4ème coordonnée des mots de 7 lettres. On sait donc à présent que $x_4 = y_1 + y_2$ En procédant de même, on démontre que $x_5 = y_2 + y_3$; $x_6 = y_1 + y_2 + y_3$ et $x_7 = y_1 + y_3$.

On peut maintenant compléter la matrice génératrice :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}$$

Cette matrice génératrice permet de transformer, de manière calculatoire, un mot de 3 lettres en un mot de 7 lettres que l'on transmet par l'intermédiaire d'un canal de transmission. Pendant cette transmission, une erreur éventuelle peut surgir dans ce dernier. Comment la détecter et la corriger ?

A. SARKHANOV

VII – Matrice de parité du [7,3,3] – code.

Rôle :

La matrice de parité permet de vérifier s'il n'y a pas d'erreur (et une seule car on ne sait pas en corriger plus) qui a pu se glisser lors de l'envoi.

Comment déterminer celle du [7, 3, 3]-code ?

Lorsque l'on reçoit un mot de sept lettres, ses quatre dernières coordonnées doivent vérifier les quatre équations définies ci avant.

En effet, nous avions :

$x_4 = y_1 + y_2$ $\qquad$ $x_4 = x_1 + x_2$

$x_5 = y_2 + y_3$ $\qquad$ $x_5 = x_2 + x_3$

$x_6 = y_1 + y_2 + y_3$ $\quad \rightarrow \quad$ $x_6 = x_1 + x_2 + x_3$

$x_7 = y_1 + y_3$ $\qquad$ $x_7 = x_1 + x_3$

On transpose x_4, x_5, x_6, x_7, ce qui donne :

$x_1 + x_2 - x_4 = 0$ $\qquad$ $x_1 + x_2 + (0\text{-}1)x_4 = 0$

$x_2 + x_3 - x_5 = 0$ $\qquad$ $x_2 + x_3 + (0\text{-}1)x_5 = 0$

$x_1 + x_2 + x_3 - x_6 = 0$ $\quad$ ce qui peut se réécrire : $\quad x_1 + x_2 + x_3 + (0\text{-}1)x_6 = 0$

$x_1 + x_3 - x_7 = 0$ $\qquad$ $x_1 + x_3 + (0\text{-}1)x_7 = 0$

Or **0 – 1 = 1** car nous travaillons dans le corps F_2 composé seulement de 0 et de 1 (voir partie III – Corps fini F_2)

Donc $x_1 + x_2 + x_4 = 0$ (1)

$x_2 + x_3 + x_5 = 0$ (2)

$x_1 + x_2 + x_3 + x_6 = 0$ (3)

$x_1 + x_3 + x_7 = 0$ (4)

On réécrit alors ces équations sous forme matricielle, ce qui formera la matrice de parité.

Pour ce faire, sur chaque ligne correspondant à une équation, on écrit chaque coordonnée par ordre croissant jusque 7 sous forme de « 0 » ou de « 1 ».

Pour savoir lequel écrire, rien de plus simple : on place un « 0 » lorsque la coordonnée n'existe pas et un « 1 » lorsqu'elle y est.

Prenons un exemple : sur la première ligne on placera 1101000 car l'équation (1) comporte x_1, x_2, x_4 donc on écrit un « 1 » à la première, deuxième et quatrième place et des « 0 » aux places restantes.

On fait de même avec les autres équations et au final et on obtient la matrice de parité suivante :

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

Si le mot de sept lettres reçu ne contient pas d'erreur c'est-à-dire figurant parmi les 8 mots du code, le résultat de la multiplication par ce mot donnera : $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

Mais une erreur peut apparaître pendant la transmission.

<u>Comment peut-on alors la détecter puis la corriger ?</u>

Par exemple, on reçoit le mot : 1011110 (qui n'appartient donc pas au code).

On peut soit effectuer la multiplication matricielle ou remplacer dans les quatre équations précédentes les coordonnées correspondantes au mot reçu.

D'après l'équation (1), on obtient 1 + 0 + 1 = 0 (car on utilise le corps F_2). Donc il n'y a pas d'erreur. De même pour l'équation (2) et (4).

Mais si l'on s'attarde sur la troisième équation 1 + 0 + 1 + 1 = 1 ≠ 0. Cette équation contient donc une erreur. Or si l'on observe bien, seule la coordonnée 6 n'apparaît que dans cette équation erronée, les autres coordonnées apparaissant dans d'autres équations. Donc l'erreur se situe sur x_6. On remplace alors le « 1 » de la sixième coordonnée en « 0 ». Le mot était donc 1011100.

Il ne nous reste plus qu'à retrouver le mot initial (3 lettres).

Pour y parvenir, on sait que $y_1 = x_1$; $y_2 = x_2$ et $y_3 = x_3$ donc il suffit de supprimer les quatre dernières coordonnées du mot de sept lettres. Pour cet exemple le mot de trois lettres envoyé était donc 101 !!

D. PETIT

VIII – [7,4,3] – code.

On veut augmenter le nombre de lettres des mots initiaux, tout en gardant une même distance minimale égale à 3. On pourra donc corriger 1 erreur.

On peut augmenter le nombre de lettres des mots initiaux, car pour corriger 1 erreur sur des mots de 7 lettres, 3 équations suffisent à notre matrice de parité. En effet, s'il y a 1 erreur, on a 7 cas : 1 erreur sur x_1 ; x_2 ; x_3 ; x_4 ;x_5 ; x_6 ; x_7. En comptant le cas ou il n'y a pas d'erreur, cela nous fait 8 cas possibles. Comme $2^3 = 8$, il nous faut dans notre corps F_2 au moins trois lettres de plus pour chaque mot donc 3 équations de parité au minimum.

Avec 4 lettres aux mots initiaux, on a 2^4 soit 16 mots de 4 lettres. Or avec le [7,3,3] – code, on avait 16 mots de 7 lettres à une distance minimale de 3. On essaye donc de faire correspondre les mots initiaux de 4 lettres avec les 4 premières lettres des mots de 7 lettres.

Mot de 4 lettres	Mot de 7 lettres	Mot de 4 lettres	Mot de 7 lettres
0000	0000000	0110	0110100
1000	1000110	0101	0101110
0100	0100011	0011	0011010
0010	0010111	1110	1110010
0001	0001101	1101	1101000
1100	1100101	1011	1011100
1010	1010001	0111	0111001
1001	1001011	1111	1111111

La correspondance fonctionne. On a donc $x_1 = y_1$; $x_2 = y_2$; $x_3 = y_3$; $x_4 = y_4$ (les y_i correspondent aux lettres des mots initiaux et les x_i aux lettres des mots de 7 lettres).

Il nous reste donc à exprimer les 3 dernières coordonnées x_5, x_6, x_7 en fonction des 4 premières. On répertorie toutes les équations linéaires possibles avec y_1, y_2, y_3, y_4. On a donc :

$$y_1+y_2 \; ; \; y_1+y_3 \; ; \; y_1+y_4 \; ; \; y_2+y_3 \; ; \; y_2+y_4 \; ; \; y_3+y_4 \; ; \; y_1+y_2+y_3 \; ; \; y_1+y_2+y_4 \; ;$$
$$y_1+y_3+y_4 \; ; \; y_2+y_3+y_4 \; ; \; y_1+y_2+y_3+y_4$$

A l'aide d'un tableur, on compare les résultats de chaque équation avec les valeurs correspondantes des x_5, x_6 et x_7. On obtient ainsi le tableau de l'ANNEXE 2.

On trouve donc :

$$x_5 = y_1+y_3+y_4$$
$$x_6 = y_1+y_2+y_3$$
$$x_7 = y_2+y_3+y_4$$

On a donc nos 7 équations : $x_1 = y_1$ $x_2 = y_2$ $x_3 = y_3$ $x_4 = y_4$
$x_5 = y_1+y_3+y_4$ $x_6 = y_1+y_2+y_3$ $x_7 = y_2+y_3+y_4$

On peut donc en déduire une matrice génératrice et de parité en faisant de même que pour le [7 ; 3 ; 3] – code.

Matrice génératrice : $\begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&0&1&1 \\ 1&1&1&0 \\ 0&1&1&1 \end{pmatrix}$ **matrice de parité** : $\begin{pmatrix} 1&0&1&1&1&0&0 \\ 1&1&1&0&0&1&0 \\ 0&1&1&1&0&0&1 \end{pmatrix}$

Dans la partie suivante, nous allons voir si l'on peut augmenter de nouveau à 5 le nombre de lettres des mots initiaux tout en gardant une distance minimale de 3 pour corriger 1 erreur.

B. GOUPILLEAU

IX – [9,5,3] – code.

Pour continuer à augmenter le nombre de lettres à coder, on passe au [9,5,3] – code. Cette fois les mots initiaux ont 5 coordonnées et on les allonge de 4 coordonnées. On ne peut pas faire moins de 4.

En effet pour des mots initiaux de k = 5 coordonnées il faut pouvoir détecter n + 1 cas : 1 erreur sur les n coordonnés et le cas sans erreurs (On suppose qu'il n'y a pas plus d'une erreur). Ceci correspond au nombre de résultats possibles après multiplication par la matrice de parité. Or pour un [7,5,3] – code on ajouterait 2 coordonnées donc 2 équations desquelles on déduit une matrice de parité à 2 lignes. On aura donc des résultats à 2 coordonnées, soit 2^2 = 4 résultats possibles. Or il faut détecter 7 + 1 = 8 cas, le [7,5,3] – code est donc impossible.

Il faut donc augmenter le nombre de coordonnées à ajouter tel que :

$n + 1 \leq 2^{n-k}$ ici k = 5 (5 lettres au départ), il faut résoudre $n + 1 \leq 2^{n-5}$

(cf. ANNEXE 3 pour la résolution de cette inéquation)

Pour le [8,5,3] – code on a :

$8 + 1 < 2^{8-5}$??

Or 9 > 8 IMPOSSIBLE

$\Rightarrow$ Ce code n'est pas possible

Pour le [9,5,3] – code on a :

$9 + 1 < 2^{9-5}$??

Or 10 < 16 POSSIBLE

$\Rightarrow$ L'allongement minimal est donc de 4 coordonnées

Il faut maintenant trouver les mots du code. Il y en a 2^5 = 32. Il y a des conditions à respecter pour choisir ces mots :

- le premier sera composé que de 0
- les 5 premières coordonnées seront ceux des mots de départ. On a ainsi $x_1 = y_1$, $x_2 = y_2$, $x_3 = y_3$, $x_4 = y_4$, $x_5 = y_5$
- la distance minimale entre 2 mots doit être de 3 de manière à pouvoir corriger 1 erreur

Il faut donc chercher 4 équations qui vérifient que la distance minimale soit de 3. On a recours à un tableur où l'on entre les 5 premières coordonnées puis on définit les autres par des équations à tester parmi les 26 possibles. Le tableur donne automatiquement par des fonctions bien choisies la distance entre 2 mots 2 à 2. Voir l'ANNEXE 4.

En testant différentes équations, on trouve la combinaison d'équation suivante qui vérifie nos conditions :

$x_6 = y_1 + y_2 + y_3 + y_4$

$x_7 = y_2 + y_3 + y_4 + y_5$

$x_8 = y_1 + y_3 + y_4 + y_5$

$x_9 = y_1 + y_2 + y_3 + y_5$

On a ainsi les 9 équations définissant les mots du code. On en déduit comme précédemment une matrice génératrice :

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

et une matrice de parité :

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

A. CHAIGNE

X – Conclusion et prolongements.

Ce travail nous a permis de comprendre comment on peut en pratique détecter et corriger des erreurs de transmission grâce à des outils mathématiques. Nous avons seulement eu le temps de construire quelques codes où nous pouvions détecter puis corriger une seule erreur. Ceci nous a tout de même aidé à comprendre comment fabriquer un code correcteur d'erreurs.

Les prolongements de ce travail sont nombreux nous pouvons citer quelques pistes :

- Pouvoir construire un code permettant de corriger 2 erreurs (ceci est en cours d'essai, les difficultés se multiplient du fait que le nombre de cas à envisager en beaucoup plus important.
- Comment allonger le moins possible les mots initiaux pour un nombre d'erreurs à corriger fixé : ceci a été abordé par le groupe de Poitiers qui a mis en évidence la borne supérieure de Hamming.
- Peut-on toujours trouver un code optimal pour k et d fixés ?

Ce sont des questions complètement ouvertes qui sont des pistes de recherche actuelles, questions sur lesquelles des chercheurs planchent tous les jours en France et partout dans le monde.

XI – Annexes.

ANNEXE 1

Soient $x = x_1x_2x_3\ldots x_{n-1}x_n$ et $y = y_1y_2y_3\ldots y_{n-1}y_n$ deux mots binaires de longueur n, on appelle la distance de Hamming entre x et y le nombre $\mathbf{d_H\ (x\ ;\ y)} = \sum_{i=1}^{n} |x_i - y_i|$.

On peut vérifier que ceci est une distance sur F_2.

Soit C un code (c'est à dire un ensemble de mots de longueur n), on appelle distance minimale du code C le nombre $\mathbf{d = min\ \{\ d_H\ (x\ ;\ y)\ /\ x \in C\ ;\ y \in C\ ;\ x \neq y\}}$.

Prenons un exemple pour comprendre : nous allons déterminer la distance minimale du code défini par les 16 mots de 7 lettres que nous avons construit et nous calculerons la distance minimale de ce code.

Les mots sont :

0000000	1111111
1101000	0010111
0110100	1001011
0011010	1100101
0001101	1110010
1000110	0111001
0100011	1011100
1010001	0101110

Prenons par exemple les mots x = **0**011**0**1**0** et y = 1**0**1**0**0**0**1.

Comparons lettre à lettre le nombre de caractères différents (en gras). Il y a quatre caractères différents, on en déduit que $\mathbf{d_H\ (x\ ;\ y) = 4}$.

Dans le tableau ci-contre, nous indiquons la distance de Hamming entre deux mots de ce code.

0	0	0	0	0	0	0	0	3	3	3	3	3	3	7	4	4	4	4	4	4	4
1	1	0	1	0	0	0	3	0	4	4	4	4	4	4	7	3	3	3	3	3	3
0	1	1	0	1	0	0	3	4	0	4	4	4	4	4	3	7	3	3	3	3	3
0	0	1	1	0	1	0	3	4	4	0	4	4	4	4	3	3	7	3	3	3	3
0	0	0	1	1	0	1	3	4	4	4	0	4	4	4	3	3	3	7	3	3	3
1	0	0	0	1	1	0	3	4	4	4	4	0	4	4	3	3	3	3	7	3	3
0	1	0	0	0	1	1	3	4	4	4	4	4	0	4	3	3	3	3	3	7	3
1	0	1	0	0	0	1	3	4	4	4	4	4	4	0	3	3	3	3	3	3	7
1	1	1	1	1	1	1	7	4	4	4	4	4	4	4	0	3	3	3	3	3	3
0	0	1	0	1	1	1	4	7	3	3	3	3	3	3	3	0	4	4	4	4	4
1	0	0	1	0	1	1	4	3	7	3	3	3	3	3	3	4	0	4	4	4	4
1	1	0	0	1	0	1	4	3	3	7	3	3	3	3	3	4	4	0	4	4	4
1	1	1	0	0	1	0	4	3	3	3	7	3	3	3	3	4	4	4	0	4	4
0	1	1	1	0	0	1	4	3	3	3	3	7	3	3	3	4	4	4	4	0	4
1	0	1	1	1	0	0	4	3	3	3	3	3	7	3	3	4	4	4	4	4	0
0	1	0	1	1	1	0	4	3	3	3	3	3	3	7	3	4	4	4	4	4	4

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

On remarque une diagonale de 0 (un mot est à une distance de Hamming de 0 de lui même) et on remarque que la plus petite distance de Hamming entre deux mots différents vaut 3. On en déduit que la distance minimale de ce code est $\mathbf{d = 3}$.

Il existe un théorème qui dit que si la distance minimale d'un code est d, alors on peut détecter et corriger $e = \frac{d-1}{2}$ erreurs. Avec ces mots de 7 lettres, on peut fabriquer un code qui nous permettra de détecter puis de corriger e = 1 erreur .

ANNEXE 2

x1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x5	0	1	1	0	0	1	1	0	1	0	0	1	1	0	0	1
x6	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	1
x7	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1
y1+y2	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
y1+y3	0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0
y1+y4	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
y2+y3	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
y2+y4	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0
y3+y4	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
y1+y2+y3	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	1
y1+y2+y4	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1
y2+y3+y4	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1
y1+y3+y4	0	1	1	0	0	1	1	0	1	0	0	1	1	0	0	1
y1+y2+y3+y4	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0

x_1, x_2, x_3 et x_4 sont respectivement égaux à y_1, y_2, y_3 et y_4, les trois lignes suivantes correspondent aux valeurs que l'on avait trouvé en « fabriquant » nos 16 mots de sept lettres. Les onze dernières lignes ont été calculées par un tableur de façon à respecter les règles de calcul de F_2.

On remarque que les lignes grisées de la même teinte concordent exactement, ceci nous permet donc d'en déduire les trois dernières équations que l'on cherchait !

ANNEXE 3

On veut résoudre :

$n+1 \leq 2^{n-5}$

ou $2^{n-5}-(n+1) \geq 0$

on pose la suite $Un = 2^{n-5}-(n+1)$, $n \in N$

on cherche $Un \geq 0$ D'après le tableau ci-dessous, $n \geq 9$!

n	Un
0	-0,96875
1	-1,97375
2	-2,875
3	-3,75
4	-4,5
5	-5
6	-5
7	-4
8	-1
9	6

ANNEXE 4

X1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
X2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
X6=X1+X2+X3+X4	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0
X7=X2+X3+X4+X5	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
X8=X1+X3+X4+X5	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
X9=X1+X2+X3+X5	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	0	1	0	1	1	0	1	0

X1	X2	X3	X4	X5	X6	X7	X8	X9																								
0	0	0	0	0	0	0	0	0	0	4	4	4	5	3	3	5	4	4	4	4	3	5	5	7	4	4	4	4	3	5	5	7	4	4	4	8	5	7	7	5
0	0	0	0	1	0	1	1	1	4	0	4	4	3	5	5	3	4	4	4	4	5	3	7	5	4	4	4	4	5	3	7	5	4	4	8	4	7	5	5	7
0	0	0	1	0	1	1	1	0	4	4	0	4	3	5	5	3	4	4	4	4	5	7	3	5	4	4	4	4	5	7	3	5	4	8	4	4	7	5	5	7
0	0	0	1	1	1	0	0	1	4	4	4	0	5	3	3	5	4	4	4	4	7	5	5	3	4	4	4	4	7	5	5	3	8	4	4	4	5	7	7	5
0	0	1	0	0	1	1	1	1	5	3	3	5	0	4	4	4	3	5	5	7	4	4	4	4	3	5	5	7	4	4	4	4	5	7	7	5	4	4	4	8
0	0	1	0	1	1	0	0	0	3	5	5	3	4	0	4	4	5	3	7	5	4	4	4	4	5	3	7	5	4	4	4	4	7	5	5	7	4	4	8	4
0	0	1	1	0	0	0	0	1	3	5	5	3	4	4	0	4	5	7	3	5	4	4	4	4	5	7	3	5	4	4	4	4	7	5	5	7	4	8	4	4
0	0	1	1	1	0	1	1	0	5	3	3	5	4	4	4	0	7	5	5	3	4	4	4	4	7	5	5	3	4	4	4	4	5	7	7	5	8	4	4	4
0	1	0	0	0	1	1	0	1	4	4	4	4	3	5	5	7	0	4	4	4	5	3	3	5	4	4	4	8	5	7	7	5	4	4	4	4	3	5	5	7
0	1	0	0	1	1	0	1	0	4	4	4	4	5	3	7	5	4	0	4	4	3	5	5	3	4	4	4	4	7	5	5	7	4	4	4	4	5	3	7	5
0	1	0	1	0	0	0	1	1	4	4	4	4	5	7	3	5	4	4	0	4	3	5	5	3	4	4	4	4	8	4	4	7	5	5	7	4	4	4	4	4
0	1	0	1	1	0	1	0	0	4	4	4	4	7	5	5	3	4	4	4	0	5	3	3	5	4	4	4	4	5	7	7	5	4	4	4	4	7	5	5	3
0	1	1	0	0	0	0	1	0	3	5	5	7	4	4	4	4	5	3	3	5	0	4	4	4	5	7	7	5	4	4	4	8	3	5	5	7	4	4	4	4
0	1	1	0	1	0	1	0	1	5	3	7	5	4	4	4	4	3	5	5	3	4	0	4	4	7	5	5	7	4	4	8	4	5	3	7	5	4	4	4	4
0	1	1	1	0	1	1	0	0	5	7	3	5	4	4	4	4	3	5	5	3	4	4	0	4	7	5	5	7	4	8	4	4	5	7	3	5	4	4	4	4
0	1	1	1	1	1	0	1	1	7	5	5	3	4	4	4	4	5	3	3	5	4	4	4	0	5	7	7	5	8	4	4	4	7	5	5	3	4	4	4	4
1	0	0	0	0	1	0	1	1	4	4	4	4	3	5	5	7	4	4	4	8	5	7	7	5	0	4	4	4	5	3	3	5	4	4	4	4	3	5	5	7
1	0	0	0	1	1	1	0	0	4	4	4	4	5	3	7	5	4	4	8	4	7	5	5	7	4	0	4	4	3	5	5	3	4	4	4	4	5	3	7	5
1	0	0	1	0	0	1	0	1	4	4	4	4	5	7	3	5	4	8	4	4	7	5	5	7	4	4	0	4	3	5	5	3	4	4	4	4	5	7	3	5
1	0	0	1	1	0	0	1	0	4	4	4	4	7	5	5	3	8	4	4	4	5	7	7	5	4	4	4	0	5	3	3	5	4	4	4	4	7	5	5	3
1	0	1	0	0	0	1	0	0	3	5	5	7	4	4	4	4	5	7	7	5	4	4	4	8	5	3	3	5	0	4	4	4	3	5	5	7	4	4	4	4
1	0	1	0	1	0	0	1	1	5	3	7	5	4	4	4	4	7	5	5	7	4	4	8	4	3	5	5	3	4	0	4	4	5	3	7	5	4	4	4	4
1	0	1	1	0	1	0	1	0	5	7	3	5	4	4	4	4	7	5	5	7	4	8	4	4	3	5	5	3	4	4	0	4	5	7	3	5	4	4	4	4
1	0	1	1	1	1	1	0	1	7	5	5	3	4	4	4	4	5	7	7	5	8	4	4	4	5	3	3	5	4	4	4	0	7	5	5	3	4	4	4	4
1	1	0	0	0	0	1	1	0	4	4	4	8	5	7	7	5	4	4	4	4	3	5	5	7	4	4	4	4	3	5	5	7	0	4	4	4	5	3	3	5
1	1	0	0	1	0	0	0	1	4	4	8	4	7	5	5	7	4	4	4	4	5	3	7	5	4	4	4	4	5	3	7	5	4	0	4	4	3	5	5	3
1	1	0	1	0	1	0	0	0	4	8	4	4	7	5	5	7	4	4	4	4	5	7	3	5	4	4	4	4	5	7	3	5	4	4	0	4	3	5	5	3
1	1	0	1	1	1	1	1	1	8	4	4	4	5	7	7	5	4	4	4	4	7	5	5	3	4	4	4	4	7	5	5	3	4	4	4	0	5	3	3	5
1	1	1	0	0	1	0	0	1	5	7	7	5	4	4	4	8	3	5	5	7	4	4	4	4	3	5	5	7	4	4	4	4	5	3	3	5	0	4	4	4
1	1	1	0	1	1	1	1	0	7	5	5	7	4	4	8	4	5	3	7	5	4	4	4	4	5	3	7	5	4	4	4	4	3	5	5	3	4	0	4	4
1	1	1	1	0	0	1	1	1	7	5	5	7	4	8	4	4	5	7	3	5	4	4	4	4	3	5	5	3	4	4	0	4	5	7	3	5	4	4	0	4
1	1	1	1	1	0	0	0	0	5	7	7	5	8	4	4	4	7	5	5	3	4	4	4	4	5	3	3	5	4	4	4	0	7	5	5	3	4	4	4	0

Ce tableau, obtenu après des essais successifs en changeant les 4 dernières coordonnées des mots, nous permet d'obtenir 32 mots de 9 lettres qui sont à une distance minimale de 3. Ceci nous permet de fabriquer un code où l'on peut détecter et corriger 1 erreur.