

Comment ça marche ?

Les transmissions radio-numériques

14 – Correction des erreurs

Par le radio-club F6KRK

Après avoir vu la détection des erreurs de transmission, nous allons maintenant aborder les principales méthodes employées pour les corriger.

Présentation du problème

Nous avons vu que les transmissions numériques se faisaient par blocs de données et qu'une seule erreur dans le bloc suffisait pour, soit l'ignorer, soit en demander la répétition. Dans le cas où peu d'erreurs surviennent, il peut être plus rentable d'ajouter quelques données supplémentaires pour corriger ces erreurs, en particulier pour une transmission du type "Radiodiffusion". On utilise alors des algorithmes de détection et de correction dont les performances se mesurent au rapport entre le nombre d'erreurs corrigées et le nombre d'octets ajoutés.

Les codes de Hamming

Un code correcteur d'erreur permet de reconstituer le message émis même si des erreurs (en nombre limité) ont altéré le message, constitué ici de bits $\{0,1\}$.

Chaque mot de longueur m est codé par un mot de longueur n avec $n > m$. Parmi les n bits du mot-codé, m bits reproduisent le mot-source, les $n-m$ autres sont les bits de correction. Le rendement est égal n/m et l'efficacité est égale à $(n-m) / m$.

Les codes de Hamming pour lesquels $n = 2^k - 1$ et $m = n - k$, permettent de corriger **une erreur**. Avec $k=3$, nous avons le code 7-4, avec $k=4$, le code 15-11 et avec $k=5$ le code 31-26. Le rendement est respectivement de 57%, 73% et 84%. Mais bien sûr, plus m est grand et plus il risque d'y avoir plusieurs erreurs. Un bon compromis consiste à prendre $k=4$.

Les données informatiques sont organisées en blocs d'octets comportant 8 bits. Il est alors plus simple pour $k=4$ de prendre un code 12-8⁽¹⁾, composé d'un octet de donnée plus un quartet de contrôle. Le rendement passe de 73% à 67% ce qui n'est pas trop cher payé.

Exemple de mise en œuvre (code 12-8)

Elle comprend trois parties : l'encodage, le décodage et la correction éventuelle d'une erreur. La figure 1 montre l'organigramme de la routine d'encodage.

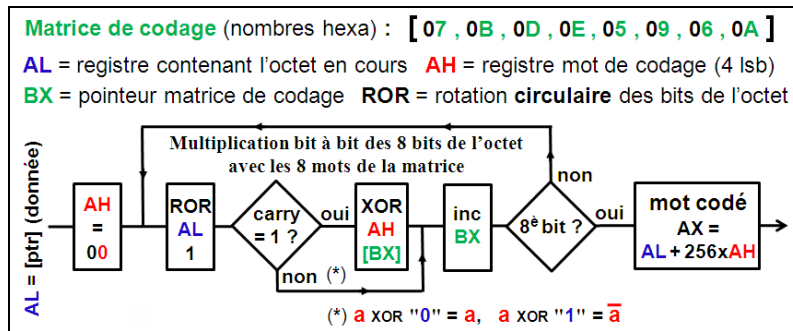


Figure 1 : Organigramme logiciel d'un encodage de Hamming {12,8}

Les habitués de la programmation sur micro-contrôleur n'auront aucun mal à implanter cet algorithme dans leur langage. Noter que le résultat est sur 12 bits car le deuxième quartet de AH reste toujours à zéro.

La figure 2 montre l'organigramme de la routine du décodage.

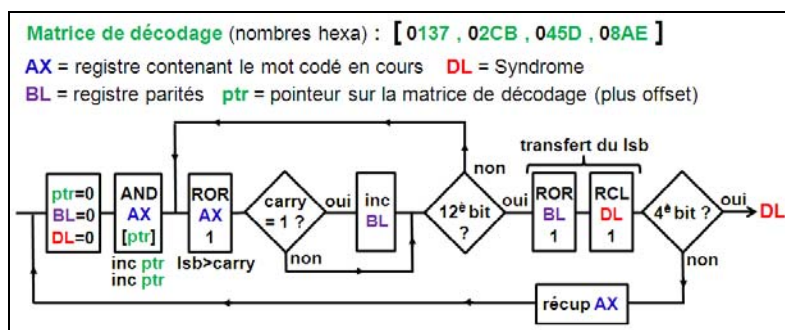


Figure 2 : Organigramme logiciel du décodage de Hamming {12,8}

A la sortie de la routine, le registre DL contient les 4 bits du syndrome. Si ce dernier vaut zéro, alors il n'y a aucune erreur dans le mot de 12 bits et les 8 lsb constituent l'octet de la donnée. Sinon, la valeur du syndrome va donner la position du bit erroné dans le mot de 12 bits. Après correction de l'erreur dans le fichier des données reçues, on effectue un second décodage et si le syndrome n'est toujours pas nul, c'est qu'il y a plus d'une erreur. Alors on remet le mot original dans le fichier en espérant que les erreurs soient dans les 4 bits du mot de codage. C'est le contrôle final du bloc à l'aide d'un CRC qui tranchera. La figure 3 montre l'organigramme logiciel de la correction d'erreur selon cet algorithme.

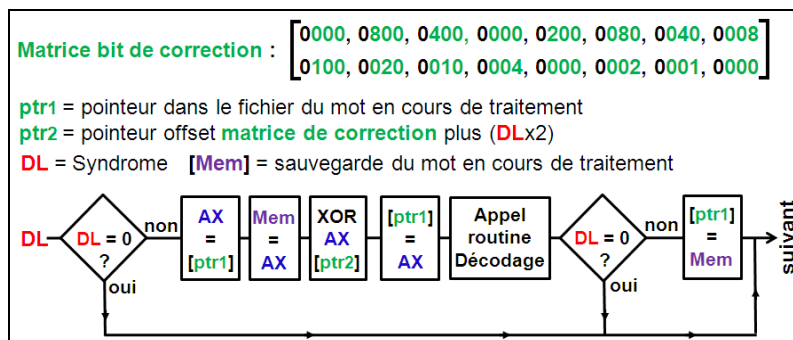


Figure 3 : Organigramme logiciel de la correction de Hamming {12,8}

Noter que "ptr" entre crochets signifie que le mot à traiter est à cet endroit dans la mémoire.

Codes de Reed-Solomon

Les codes de Reed (et) Solomon sont des codes par bloc de la forme $n = 2^m - 1 = k + 2t$. Ils prennent en entrée un bloc de données de taille k , chaque donnée étant un symbole de 2^m éléments. Généralement $m = 8$ (un octet). On ajoute à ce bloc $2t$ symboles de contrôle formant ainsi un bloc de sortie de n symboles (n octets). On voit alors que si les codes de Hamming codent des bits, ceux de Reed-Solomon codent des mots (octets).

Grâce à l'ajout des symboles de contrôle, ces codes permettent de corriger deux types d'erreurs :

- Les erreurs induisant une modification des données, où certains bits passent de la valeur 0 à la valeur 1, et vice-versa.

- Les erreurs provoquant des pertes d'informations aussi appelées "effacements" ⁽²⁾.

On note un codage de Reed-Solomon $RS(n,k)$ ou $RS(n,k,t)$. Il sait corriger t erreurs pour $2t$ symboles de contrôle. Exemple de code employé pour le DVB : $RS(204,188,8)$.

Nous ne nous étendons pas sur les algorithmes de codage et de décodage d'un code Reed-Solomon qui sont beaucoup plus complexes que ceux de Hamming.

Voyons les efficacités des deux systèmes en comparant le codage $RS\{204, 188\}$ du DVB et un codage de Hamming 15-11 :

- Rendement m/n pour Hamming, soit $11 / 15 = 0,733$, à comparer avec k/n pour Reed-Solomon, soit $188/204 = 0,922$.
- Efficacité de correction : $1/15 = 0,0667$ pour Hamming et $8/204 = 0,0392$ pour Reed-Solomon, dans le cas où il n'y a qu'une seule erreur dans 8 octets et $64/204 = 0,314$ dans le cas où il y a 8 erreurs dans 8 octets. En pratique ce cas a très peu de chances de se rencontrer.

Globalement, on peut dire que **statistiquement**, les codes de Reed-Solomon sont plus performants, d'autant qu'ils permettent également de corriger $2t$ effacements.

Entrelacement des données

Quel que soit le codage employé, pour éviter que des rafales d'erreurs aient une taille supérieure à la possibilité du code, on effectue un entrelacement des bits après codage et un désentrelacement avant le décodage, le but étant de mélanger les bits pour « casser » les paquets d'erreurs.

On peut utiliser un entrelacement convolutif performant mais complexe à mettre en œuvre, ou un entrelacement par bloc, tel celui que nous allons voir sur la figure 4 pour un code de Hamming 12-8.

| | | Code correcteur | | | | Donnée | | | | | | | |
|---------------|-----|-----------------|-----|----|----|--------|----|----|----|----|----|----|----|
| | | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| Mots transmis | b11 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | b10 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | b9 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| | b8 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| | b7 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| | b6 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| | b5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| | b4 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| | b3 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | b2 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| | b1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| | b0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

Figure 4 : Principe de l'entrelacement par bloc

N-B : Les codes correcteurs de la figure 4 sont fantaisistes.

Nous voyons que pour les deux groupes d'erreurs de deux et trois bits consécutifs, il n'y a plus qu'une seule erreur par mot après désentrelacement et elle peut être corrigée. Ici, nous avons un bloc brut de 18 octets pour 12 octets de données (rendement 66,7 %).

Double codage (concaténation), avant et après entrelacement

Prenons la figure 4 et appliquons le code de Hamming aux octets transmis. Dans ce cas là, il n'y aura plus que 8 octets utiles et le rendement sera de 50%. Mais nous aurons alors deux corrections des erreurs, l'une avant entrelacement et l'autre après. Si cette méthode n'est pas très efficace avec un code de Hamming, elle l'est beaucoup plus avec deux codes différents de Reed-Solomon (code **CIRC** pour **Cross Interleaved Reed-Solomon Code**). Par exemple pour l'encodage des CD, on code une première fois avec un code $C1 = RS\{28, 24\}$, ensuite on entrelace, puis on code à nouveau les données entrelacées avec un code $C2 = RS\{32, 28\}$ (rendement global = 75%).

Codes convolutionnels

Le principe des codes convolutionnels, inventés par Peter Elias en 1954, est non plus de découper le message en blocs finis, mais de le considérer comme une séquence semi-infinie de symboles (bits) qui passe à travers une succession de registres à décalage, dont le nombre est appelé "mémoire du code" (v). En pratique on limite v à 8 car la puissance de calcul demandée par le décodage croît comme 2^v .

Nous avons sur la figure 5 un exemple simple avec un registre à décalage à deux étages. Pour chaque bit en entrée, le codeur fournit 2 bits en sortie. Le rendement est donc de 50%.

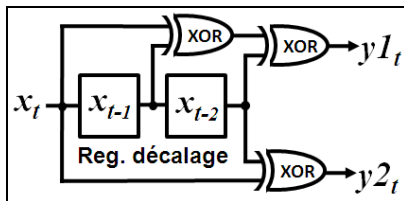


Figure 5 : Exemple de schéma logique d'un codeur convolutionnel non systématique à deux étages

Rappel : Une porte XOR réalise une addition modulo 2.

La figure 6 montre le fonctionnement d'un tel codeur pour un bloc de 4 bits ⁽³⁾.

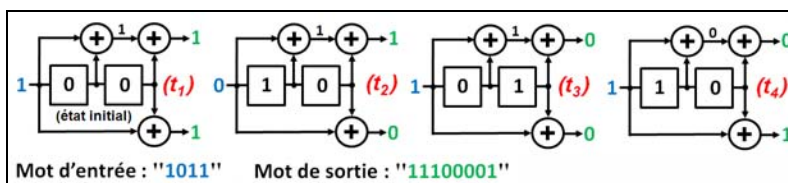


Figure 6 : Fonctionnement du codeur de la figure 5

Pour chaque bit de donnée, le codeur fournit v bits de code (le contenu du registre pour le bit de rang k).

Algorithme de décodage Viterbi

Le décodage consiste à retrouver le bit d'origine du rang k qui a conduit au contenu du registre à cet instant. Ce contenu dépend aussi de tous les bits précédant le rang k ($k, k-1, k-2$, etc.) jusqu'à l'origine où le registre contenait v fois zéro. L'algorithme de Viterbi recherche tous les chemins possibles qui ont pu conduire au contenu actuel du registre en retenant le plus court. On refait l'opération à chaque fois, ce qui fait *grosso modo* augmenter les calculs

proportionnellement au carré du nombre de bits dans le bloc. Quand une erreur n'est pas corrigée, du fait même du principe, elle se transforme en une suite. C'est pourquoi, le codage convolutif n'est en général jamais employé seul, comme nous allons le voir plus loin.

Brève comparaison entre les codes

Voir dans le tableau 1 un résumé des performances pour un code de Hamming et un code convolutif.

| Code | Taille | Rendement | Taux erreur bit (IN = $5 \cdot 10^{-2}$) |
|-----------------|--------------------|----------------|--|
| Hamming | 4b data 3b code | 4 / 7 (57%) | $2,09 \cdot 10^{-2}$ |
| Code convolutif | bloc 128 bits | 1 / 2 (50%) | $1,54 \cdot 10^{-2}$ |
| | bloc 2048 bits | 1 / 2 (50%) | $1,24 \cdot 10^{-2}$ |

Tableau 1 : Performances comparées entre un code de Hamming et un code convolutif

N-B : Ces mesures ont été faites sur un fichier contenant une image cartographique N/B non compressée.

Pour terminer, on peut ajouter :

- Les codes cycliques comme les codes de Reed-Solomon, corrigent des erreurs arrivant par paquets ou par rafales (bursts).
- Les codes convolutionnels ont de bonnes performances au décodage et la faculté de corriger une plus grande quantité d'erreurs isolées.
- Avec des canaux très bruités (Hertziens), pour profiter des avantages des deux types de codages, on peut les concaténer avec un entrelacement entre les deux, comme montré sur la figure 7.

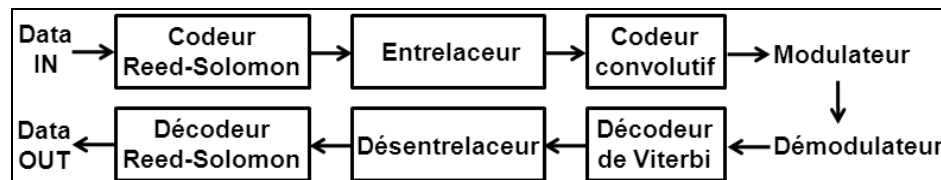


Figure 7 : Concaténation d'un code de Reed-Solomon et d'un code convolutif avec entrelacement entre les deux

Nous arrêterons là sur le sujet car le but était de comprendre succinctement le principe de la correction d'erreurs et pas d'étudier un modem HF.

Dans le prochain "comment ça marche", nous aborderons les systèmes de compressions de données pour la transmission de l'audio numérisée.

La Rubrique "Comment ça marche" est une activité collective du radio-club F6KRK (<http://www.f6krk.org>). Pour une correspondance technique concernant cette rubrique : "f5nb@orange.fr".

Notes :

- 1) Surtout quand le traitement est fait à l'aide d'un micro-contrôleur embarqué.

- 2) *Cet avantage vaut surtout pour les CD et les DVD. Pour la Radio on peut rencontrer le cas avec du 2FSK (RTTY). Prenons une démodulation avec deux détecteurs d'amplitude à F_1 et F_2 . Nous avons la table de vérité suivante : 00, 01, 10, 11 pour laquelle "01" = "0" et "10" = "1". Pour "00" et "11", On peut décider de les attribuer, soit à "0", soit à "1". Cela peut être fait avec un code Reed-Solomon, ou à l'aide d'un algorithme particulier.*
- 3) *En pratique les blocs varient d'une centaine d'octets à quelques milliers.*