# Codes détecteurs et correcteurs d'erreurs

*Travaux Dirigés de LO22/AI20*

Ce TD a pour objectif de se familiariser de mettre en pratique les notions vues en cours à propos des codes détecteurs et correcteurs d'erreurs.

## 1 Codes de Hamming

On se propose de généraliser le code [4,7,3] vu en cours. On rappelle que selon ce principe disposant d'un message de n bits au départ : $M_n$ $M_{n-1}$ ...$M_2$ $M_1$ on ajoute k bits de contrôle $C_kC_{k-1}$...$C_2C_1$, le message émis $E_NE_{N-1}$...$E_2E_1$comptant par conséquent N bits (N=n+k), les nombres n et k étant choisis de telle sorte que toute erreur simple puisse être corrigée. La distance de Hamming doit donc valoir au minimum 3 (un code de Hamming de distance 2D+1 étant capable de corriger D erreurs) : une différence d'un bit de message doit générer au minimum 2 différences dans les bits de contrôle.

Le principe adopté est donc le suivant : les bits de contrôles sont obtenus par addition modulo 2 (XOR) de bits de messages, chaque bit de message intervenant dans exactement une et une seule des combinaisons à 2, 3...k des bits de contrôle.

A l'arrivée on utilise les mêmes équations pour calculer les k bits de contrôles en fonction des n bits de message : Sous l'hypothèse des erreurs simples, si une seule équation est fausse, c'est nécessairement le bit de contrôle correspondant qui est faux. Sinon la combinaison précise à 2, 3...k des erreurs permet de retrouver le bit de message erroné.

1. En écrivant en fonction de k le nombre de combinaisons à au moins deux bits parmi k, montrer que le nombre maximal possible de bits du message vaut n=$2^k$-k-1 et que par conséquent la taille totale du message émis est N=$2^k$-1. La distance de Hamming étant de 3, il s'agit par conséquent d'un code [$2^k$-k-1, $2^k$-1,3] dont le code [4,7,3] vu en cours constitue le représentant pour k=3.

On rappelle le principe vu en cours :

— Les bits sont numérotés du bit de poids faible vers le bit de poids fort. Pour le code [4,7,3] le message avant addition des bits de contrôle est : $M_4M_3M_2M_1$.

— Le message émis intercale les bits de contrôle avec les bits de message en plaçant les bits de contrôle aux positions correspondant aux puissances de 2. Pour le code [4,7,3], le message émis est :

$$E_7E_6E_5E_4E_3E_2E_1 = M_4M_3M_2C_3M_2M_1C_2C_1$$

— On détermine quels $E_i$ participent à quels $C_j$ en écrivant i en binaire : pour le code [4,7,3] $M_3$=$E_6$ intervient dans $C_3$ et $C_2$ : 6 = $(110)_2$. Chaque $C_j$ est alors le XOR des bits identifiés comme devant participer à $C_j$ (hors $C_j$ lui-même).

2. Retrouver les équations permettant de calculer les $C_k$ à partir des $M_i$.
3. A l'arrivée on effectue la même opération sur les bits de message reçus et on effectue le XOR avec le bit de contrôle reçu correspondant (vaut donc 0 s'il n'y a pas eu d'erreur). Ecrire en fonction des numéros de bit du message global ($E_k$), l'expression de ces bits de contrôle. S'il y a eu une erreur, ces bits en donnent le numéro en binaire (sur 3 bits dans le cas du code [4,7,3].
4. On souhaite envoyer le message 1010 compléter le message émis 101_0__ .
5. On a reçu 1010110 : y a-t-il une erreur ? La corriger le cas échéant

On se propose maintenant d'étudier le cas où k=4, qui va donc être un code [11,15,3]

6. En appliquant le même principe que précédemment indiquer la structure sur 15 bits du message émis (position des 4 bits de contrôle).
7. L'expression de chacun des 4 bits de contrôle contient par conséquent 8 des 15 bits $E_k$. Donner pour chacun des 4 bits de contrôle la liste des numéros de bits k à prendre en compte.
8. En utilisant la règle précédente, ce message est il correct 101101011011011 ? Corriger l'erreur le cas échéant.