Communications Numériques TP 2

PolyTech IMFO3 - Basset - Tourancheau

1 Code (détecteur et) correcteur de Hamming

1. Quelle est la capacité de correction d'erreur d'un code de Hamming ?
2. Est-ce qu'un code de Hamming est linéaire ?
3. Quel est le surcoût r en bits, introduit par un code de Hamming, sur un mot de source de k bits ?
4. Donner l'efficacité maximum des codes de Hamming $[n, k]$ avec $n = k + r$ pour $r = 3, 4, 5$.
5. Donner la formule théorique de cette efficacité maximum.
6. Donner la limite de cette efficacité maximum lorsque $k \to \infty$
7. Dans un code de Hamming $[7, 4]$, soit le mot de source $m = m_1m_2m_3m_4$, on numérote en binaire de $(001)_2 = 1$ à $(111)_2 = 7$ les indices du message codé $x : x = x_{001}x_{010}x_{011}x_{100}x_{101}x_{110}x_{111}$. Les bits de contrôle $x_{001}, x_{010}, x_{100}$ se situe aux positions puissance de 2, ceux du message sont gardés dans le même ordre : $x = x_{001}x_{010}m_1x_{100}m_2m_3m_4$ Exprimer la valeurs des bits de parité en fonction de $m_1m_2m_3m_4$.
8. Comment détecte-t-on et corrige-t-on une erreur à la réception ? Illustrer par un exemple.
9. Donner la matrice génératrice pour ce code $[7, 4]$.
10. Donner la matrice de contrôle pour ce code $[7, 4]$.
11. Programmez un codeur $[n, k]$ pour des mots de source tels que $n + 1 = 2^{n-k}$.
12. Testez votre codeur sur les exemples de mots suivants :
    0000 ; 1111 ; 1011 ; 10101010101 ; 11110000101.
13. Programmez un décodeur-correcteur $[n, k]$ avec $n + 1 = 2^{n-k}$.
14. Testez votre décodeur sur les exemples de mots suivants :
    0000001 ; 0000010 ;0100000 ;1111111 ; 1011010 ; 101010101010101 ; 111100001010101.

2 Détection d'erreur CRC

1. Le code obtenu avec l'ajout d'un CRC est-il un code systématique ?
2. Qu'est ce qu'un code cyclique ?
3. Soit le message 10011101, Quelle est sa représentation polynomiale $D(x)$?
4. On veut "protéger" $D(x)$ avec le polynôme générateur $G(x) = x^4 + x^2 + x + 1$. Combien de bits va-t-on ajouter ?
5. Quels sont les bits à ajouter pour former le mot codé $C(x)$ à transmettre (donner votre calcul manuel) ?
6. Vérifiez que $C(x)$ est un multiple de $G(x)$ (donner votre calcul manuel).
7. G(x) permet de detecter toutes les erreurs de 1 bit, pourquoi ?
8. G(x) permet de detecter toutes les erreurs de 2 bits, pourquoi ?
9. G(x) permet de detecter toutes les erreurs d'un nombre impair de bits, pourquoi ?
10. G(x) permet de detecter toutes les erreurs en rafales d'un nombre inférieur à 4 bits, pourquoi ?
11. Ecrire un programme de calcul du code CRC pour un mot binaire D(x) et un pôlynome générateur G(x) donnés, il fournira C(x) en résultat.
12. Validez votre programme avec $G(x)$ et les mots de source : 11001101, 10101011, et 00110101.
13. Ecrire un programmes de vérification du CRC prenant en entrée un mot de code C(x) et un polynôme générateur G(x) et retournant la détection ou non d'erreur(s).
14. Validez votre programme avec $G(x)$ et les mots de code : 10001101000, 11101011101, et enfin les 3 mots de code générés dans la validation question 12.
15. Quelle est la complexité de votre algorithme de codage en nombre de XOR, c'est à dire le nombre d'opérations sur les mots, nécessaires pour faire le codage ?
16. Quelle est la complexité de votre algorithme de décodage en nombre de XOR, c'est à dire le nombre d'opérations sur les mots, nécessaires pour faire le décodage ?

3 Codage convolutif

1. Quelle est la différence entre les codes convolutifs et les codes en blocs linéaires ?
2. Quelle est l'efficacité lorsque k bits de mot de source donnent n bits de mot de code ?
3. Qu'est ce que la distance libre ? la profondeur de mémoire d'un code convolutif ? Expliquer brièvement quelle est la partie du message que l'on peut décoder au temps t ?
4. Soit $G_0 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$, donner au temps t, les équations des sorties du codeur v, v' en fonction de la source u_t et de ses précédentes valeurs indicée par leur profondeur de mémoire : u_{t-1}, u_{t-2} ici.
5. Pour l'exemple précédent. Soit m la profondeur de mémoire, donner l'automate d'états finis à 2^m états numérotés $u_{t-1}, \ldots, u_{t-m}$, avec ses étiquettes u_t/v_t sur les transitions où v_t est un mot de longueur 2.
6. Donner le treillis annoté pour $t = 0, ..., 8$ avec les étiquettes du poids des chemins w associés.
7. Soit $u[n] = 00101110$ le message à transmettre bit à bit successivement à $t = 0, 1, 2, \ldots$, quel est le message codé $v[n]$ correspondant ?
8. Quelle est l'efficacité de ce codage ?
9. Programmer ce codeur.
10. Soit $v[n] = 00\ 01\ 10\ 10\ 01\ 10\ 10\ 11$ un message reçu. A l'aide du diagramme d'état de la question précédente, construisez le treillis, et déterminez au mieux les erreurs. Donnez le message décodé correspondant.
11. Programmer le décodeur de Viterbi dans une implémentation à mémoire bornée, par exemple de taille 8 pas de temps.
12. Etablir un jeu de tests pour valider vos programmes, il contiendra $u[n] = 00101110$ pour valider.
13. Quelle est la distance libre de ce code ?
14. Combien d'erreur pourra-t-il corriger ?

4 Décodage de Viterbi en version non digitale

On souhaite appliquer le principe du décodeur de Viterbi sur un réseau en bande de base où les amplitudes du signal sont réelles, normalisée dans [0..1].

1. Modifiez votre programme décodeur de manière à pouvoir introduire le calcul de la distance Euclidienne pour les chemins à la place de la distance de Hamming.
2. Programmer un générateur aléatoire de valeurs en amplitude dans [0..1] et construire deux mots de code en amplitude.
3. Traduisez deux mots de code $a[n]$ en mots de code binaires $u[n]$. Pour ces deux mots de code introduire une ou deux erreurs binaire lorsque les valeurs sont proches de 0.5.
4. Décodez les mots de code binaires avec votre décodeur en version Hamming.
5. Décodez les mots de code en amplitude avec votre décodeur en version Euclidienne.
6. Obtenez-vous le même résultat ? Faites d'autres expériences et commentez.

5 Codes Reed-Solomon, LDPC et Turbocode, étude bibliographique

1. Donner le principe.
2. Dans quelles catégories se trouvent-t-ils ?
3. Comparer avec les codes étudiés.
4. Pour quel type d'erreurs sont-ils utilisés ?