TP5 de l'UV SY26

Codes correcteurs d'erreurs

Introduction

L'objectif de ce TP est la mise en oeuvre du codage de canal pour simuler leur capacité de correction et en cerner les limites. Pour cela, nous utiliserons le logiciel MATLAB avec la "Communication Toolbox" et Simulink. Nous allons étudier les deux grandes familles de codeurs de canal :

- Les codes en bloc (linéaires ou cycliques), pour lesquels des blocs de k symboles d'information sont protégés indépendamment les uns des autres par m symboles de contrôle pour former des mots codes à n symboles.
- Les codes convolutifs, pour lesquels il n'y a pas de mots codes indépendants, mais où une fenêtre glissante de largeur m se déplace sur les symboles d'information et permet de coder un symbole d'information en fonction de m symboles précédents.

1 Codes en bloc

1.1 Eléments de théorie

Nous avons vu en cours que les codes linéaires sont définis par une matrice de contrôle $\mathbf{H}$. Si $\mathbf{c}$ est un mot code, alors $\mathbf{c}.\mathbf{H}^T = \mathbf{0}$. Cette opération, calculée sur le mot de code reçu $\mathbf{c}'$, permet de calculer le syndrome et de détecter les erreurs si $\mathbf{c}.\mathbf{H}^T = \mathbf{s} \neq \mathbf{0}$. Pour l'opération de codage, on peut utiliser la matrice génératrice $\mathbf{G}$ qui est liée à la matrice de contrôle $\mathbf{H}$ par la relation $\mathbf{G}.\mathbf{H}^T = \mathbf{0}$. $\mathbf{G}$ et $\mathbf{H}$ peuvent s'écrire sous leur forme canonique $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$ et $\mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_{n-k}]$.

On obtient le mot code $\mathbf{c}$ en fonction des caractères d'information $\mathbf{m}$: $\mathbf{c} = \mathbf{m}.\mathbf{G}$.

1.2 Programmation

Deux fonctions MATLAB `encode` et `decode` permettent d'implémenter des codeurs/décodeurs de canal basés sur une matrice de contrôle ou sur un polynôme générateur.

1.3 Codage de Hamming

Le codeur de Hamming permet de corriger une erreur parmi n. Le nombre de caractères de contrôle m est lié au nombre de caractères d'information k et de code n par les relations $n = 2^m - 1$ et $k = 2^m - m - 1$. Pour travailler avec les codes de Hamming, les fonctions MATLAB sont `hammgen`, `gen2par` et `htruthtb`.

- Quelles sont les matrices génératrice et de contrôle données par MATLAB pour un codeur de Hamming $(7, 4)$? Commentez la forme de la matrice $\mathbf{H}$. Quelles en sont les conséquences pour le décodage ?
- Quel est l'intérêt de la table retournée par la fonction `htruthtb` ? Comment va-t-on utiliser cette table ?
- Calculer, en utilisant $\mathbf{G}$, le mot de code associé au mot d'information $\mathbf{m} = [1010]$.

Question 1 : Ecrire la fonction `hamcode.m` qui retourne un mot code à partir d'un bloc d'information et de la matrice de contrôle $\mathbf{H}$. On remarque que les fonctions `mod` et `rem` permettent de faire l'opération modulo.

- Vérifier que votre fonction donne le même résultat que la fonction `encode` de MATLAB.

Question 2 : Ecrire la fonction `hamdecode.m` qui retourne le mot d'information corrigé à partir du mot code reçu et de la matrice de contrôle.

- Vérifier que votre fonction donne le même résultat que la fonction `decode` de MATLAB.

1.4 Codage BCH (Bose-Chaudhuri-Hocquenghem)

Les codes cycliques sont des codes en bloc particuliers tels que toute permutation circulaire sur les symboles d'un mot de code donne encore un mot du code. Ils sont définis par un polynôme générateur.

Les codes BCH sont un cas particulier des codes cycliques.

MATLAB donne les fonctions suivantes pour travailler avec les codes cycliques : `cyclpoly`, `cyclgen` et pour les codes BCH : `bchpoly`. Les fonction `encode` et `decode` sont toujours utilisables.

– Que retourne la fonction `bchpoly` ? En utilisant les fonction `bchpoly` et `cyclgen`, donnez la matrice de contrôle associée à un code BCH$(7, 4)$, et comparer la à celle donnée par `hammgen`.

Question 3 : En utilisant les fonctions déjà vues et les fonctions `randint`, `randbit` et `gfpretty`, regrouper dans un fichier de commande MATLAB `bchtest.m`, les instructions permettant de réaliser ce qui suit :

1. Récupérer le polynôme générateur BCH permettant de corriger 3 erreurs sur 15 bits ;
2. Générer une séquence binaire aléatoire de symboles d'information ;
3. Donner le mot du code associé ;
4. Ajouter trois erreurs aléatoires ;
5. Décoder le mot erroné ;
6. Extraire les symboles d'information et comparer avec les données transmises.

2 Codes convolutifs

2.1 Généralités

Les codes convolutifs sont des codes à mémoire qui utilisent les mots de code déjà reçus pour corriger le mot de code courant. Ils sont définis par trois paramètres (n, k, m). Le nombre de symboles de code transmis est n, lorsque k symboles d'information sont codés. m indique le nombre de symboles mémorisés par le codeur.

Voici, par exemple, le schéma de deux codeurs (2, 1, 2) :

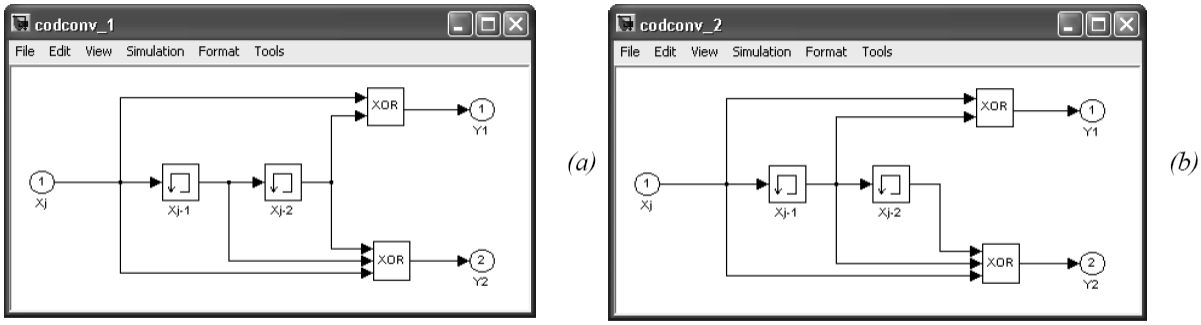

Figure 3 : Simulation de codeurs convolutifs (2, 1, 2)

Les trois paramètres (n, k, m) ne suffisent pas à décrire complètement un codeur convolutif. Il faut préciser sa fonction de transfert. Celle-ci peut être sous forme de fonction, de schéma-bloc ou de matrice de transfert qui elle-même peut être sous forme binaire, octale ou d'état. En utilisant MATLAB et SIMULINK, il est possible de passer d'une forme à l'autre.

La figure ci-dessous donne le schéma d'un codeur convolutif (3, 2, 2). Ce codeur est plus complexe que les précédents, car il code les symboles d'information deux par deux. Chaque fois que deux symboles entrent dans le codeur, trois symboles en sortent :

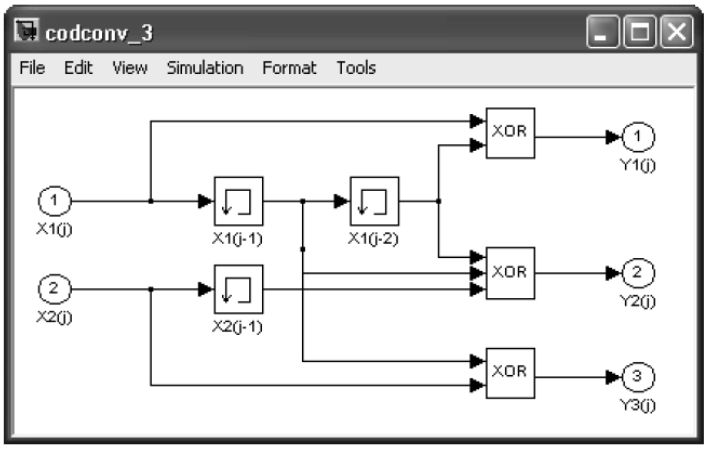

Figure 4 : Simulation d'un codeur convolutif (3, 2, 2)

La relation entre Y_j^1 et (X_j^1, X_j^2) est donnée par :

$$Y_j^1 = 1 \times X_j^1 + 0 \times X_{j-1}^1 + 1 \times X_{j-2}^1 + 0 \times X_j^2 + 0 \times X_{j-1}^2 + 0 \times X_{j-2}^2$$

Cette relation peut être mise sous la forme matricielle binaire ou octale suivante :

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

La première ligne donne les relations entre Y_j^1 et X_j^1 et la deuxième ligne entre Y_j^1 et X_j^2. La première colonne correspond à l'instant j, la deuxième à $(j-1)$ et la troisième à $(j-2)$.

On obtient une expression octale en sommant les lignes et en considérant que l'élément d'une colonne i est associée au poids 2^{i-1}.

Question 4 : Donner les équations reliant Y_j^2 et Y_j^3 aux entrées, ainsi que les deux matrices associées.

La matrice de transfert associée à un schéma de codage convolutif est la concaténation des matrices associées à chacune des sorties. Elle a donc une taille de k lignes et $(m+1) \times n$ colonnes. Cette matrice est nécessaire pour appeler les fonctions de codage et de décodage de MATLAB.

- Réaliser sous Simulink le codeur de la Figure 4. En utilisant les fonctions `sim2gen` et `oct2gen`, vérifier vos réponses à la question précédente.
- Réaliser sous Simulink le codeur de la Figure 3 (a). Les opérations de codage peuvent se faire soit avec les fonctions `encode` et `decode`, soit avec les fonctions `convenco` et `viterbi`. Les paramètres de la deuxième solution sont plus simples et mieux appropriés aux codeurs convolutifs. Dans tous les cas, nous avons besoin de la matrice de transfert associée à notre schéma-bloc. On peut la calculer avec la fonction `sim2gen`.

Question 5 :

1. Donner les équations d'entrées/sorties et la matrice de transfert du codeur.
2. Coder le message [11000100].
3. Introduire une erreur.
4. Décoder le message et le comparer avec le message transmis.

Remarque : La fonction `viterbi` qui implante l'algorithme de Viterbi doit être appelée ainsi : `viterbi(c,G,D,0,fig)` ; `G` est la matrice de transfert, `D` est l'intervalle de temps de recherche du chemin minimum, et `fig` est un numéro de figure dans laquelle sera tracé le treillis.