TP TS2 19 : TP CODAGE CANAL

1 OBJECTIF DU TP

L'objectif du TP est d'observer les améliorations apportées par les codes correcteurs d'erreur (ECCs pour Error Correcting Codes) sur les performances des transmissions numériques. Les performances sont exprimées en termes de BER (Bit Error Rate) et les simulations sont effectuées dans le cadre d'une transmission dans un canal AWGN (Additive White Gaussian Noise).
Quatre types de codes sont étudiés : les codes en blocs linéaires à base de matrice génératrice, les codes cycliques, les codes convolutifs et les codes de Reed-Solomon (RS). Le principe des simulations consiste à comparer les performances de deux chaînes de transmission : avec et sans ECCs. Ces performances sont évaluées en termes de BER en fonction du rapport E_b/N_0 où E_b désigne l'énergie moyenne reçue par bit et où $N_0/2$ désigne la densité spectrale de puissance du bruit additif blanc gaussien.
Compte-rendu : le compte-rendu de ce TP sera constitué des réponses aux questions, des codes MATLAB des programmes développés ainsi que des figures présentant les courbes de performances. Le compte-rendu sera envoyé par messagerie électronique, au format PDF.
Logiciel utilisé : MATLAB version 7.1 (ou version ultérieure)
Communications Toolbox version 3.2 (ou version ultérieure)

2 TRAVAIL A EFFECTUER

2.1 Code en bloc linéaire C(7,4)

2.1.1 Test sans bruit

Générer N_b bits avec la fonction `randint()` avec N_b=100.
Coder ces bits avec la fonction `encode()` en utilisant un code en bloc linéaire tel que k=4, n=7 et dont la matrice génératrice est la suivante :

```
G= [ 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];
```

Décoder la séquence à l'aide de la fonction `decode()` dûment paramétrée. Vérifier que la séquence émise correspond bien à la séquence décodée.

2.1.2 Test avec bruit

Refaire la même expérience mais en rajoutant un canal AWGN entre les étapes de codage et de décodage. Le canal sera simulé avec la fonction `bsc()`. La probabilité d'erreur binaire dans la fonction `bsc()` sera celle d'une transmission BPSK. La simulation sera effectuée pour un rapport E_b/N_0 allant de 0 à 7 dB (par pas de 1 dB). Des bits seront envoyés jusqu'à observer 100 erreurs.
Observer l'amélioration apportée par l'ECC sur la courbe de BER en traçant, sur deux courbes différentes, la probabilité d'erreur avec et sans codage. Par ailleurs, le résultat de la simulation sera confirmé par le tracé de la probabilité d'erreur théorique en présence de codage. Cette probabilité sera programmée à l'aide de l'instruction `bercoding()`. La valeur théorique du BER pour la chaîne non codée doit faire intervenir le facteur k/n. Commenter et conclure.

2.2 Code cyclique C(7,4)

Modifier le programme précédent de façon à remplacer le code en bloc linéaire par un code cyclique C(7,4). Identifier les lignes de codes à modifier, puis les fonctions MATLAB à utiliser. A l'issue des simulations, répondre à la question suivante : pour un BER cible de 10^{-2}, quel est le gain (en dB) sur le rapport E_b/N_0 apporté par l'ECC dans la chaîne codée par rapport à la chaîne non codée ?

2.3 Code convolutif C(3,1/2)

Reprendre la question précédente en l'adaptant à un code convolutif de rendement 1/2, de longueur de contrainte 3 et de polynômes générateur 5 et 7 (en octal). Puis adapter le tout au cas d'un code convolutif poinçonné de rendement 2/3 ayant comme code mère, le code convolutif précédent. Le poinçonnage est de la forme [1 1 ; 0 1]. Le canal ne sera plus un canal binaire symétrique mais un canal AWGN car il faut implanter un décodeur « soft-decision ».

2.4 Code Reed-Solomon RS(255,239)

Adapter le code MATLAB à un code RS(255,239) capable de corriger 8 symboles.

Codage et Décodage des codes RS :

Deux quantités sont nécessaires pour la caractérisation du code : N et K, sachant que N s'écrit sous la forme 2^M-1.

La difficulté du codage et du décodage RS réside dans le fait que les instructions de codage et de décodage RS de MATLAB ont des entrées et des sorties au format d'éléments de GF(2^M). Il faut donc

- convertir les bits en éléments de GF(2^M) avant les étapes de codage et de décodage RS,
- convertir les éléments de GF(2^M) en bits après les étapes de codage et de décodage RS.

Par ailleurs, la conversion ne peut pas se faire directement et nécessite une étape intermédiaire où les bits et les éléments de GF(2^M) sont convertis en entiers.

Codage RS : pour le codage RS, le point de départ doit être un vecteur de (Nw x K x M) bits où Nw représente le nombre de mots de K symboles envoyés, les symboles étant codés sur M bits. La sortie de l'émetteur est, quant à elle, constituée d'un vecteur de (Nw x N x M) bits.

Procédure :

Générer un vecteur de bits de taille 1 x (Nw x K x M).

Reformater le vecteur pour en faire une matrice de (Nw x K) x M bits avec la fonction `reshape()`.

Convertir le vecteur de bits en un vecteur d'entiers de taille 1 x (Nw x K) avec la fonction `bi2de()`.

Convertir le vecteur de symboles en vecteur d'éléments de GF(2^M), de taille 1 x (Nw x K) avec l'instruction `gf()`.

Reformater le vecteur pour en faire une matrice de taille Nw x K avec `reshape()`.

Coder par la fonction `rsenc()` pour donner une matrice de taille Nw x N.

Reformater pour avoir un vecteur de taille 1 x (Nw x N) avec `reshape()`.

Convertir le vecteur d'éléments de GF(2^M) en un vecteur d'entiers, de taille 1 x (Nw x N) par la fonction `gf2dec()` et la ligne de code suivante :

```
sortie=gf2dec(entree,table_gf2dec);
```

Convertir les entiers en bits dans une matrice de taille (Nw x N) x M par la fonction `de2bi()`.

Former un vecteur de bits de taille 1 x (Nw x N x M) avec `reshape()`.

Décodage RS : pour le décodage RS, le point de départ doit être un vecteur de (Nw x N x M) bits. La sortie du récepteur est, quant à elle, constituée d'un vecteur de (Nw x K x M) bits.

Procédure :

Former une matrice de taille (Nw x N) x M avec `reshape()`.

Transformer les bits en un vecteur d'entiers de taille 1 x (Nw x N) avec l'instruction `bi2de()`.

Transformer le vecteur d'entiers en un vecteur d'éléments de GF(2^M) de taille 1 x (Nw x N) avec l'instruction `gf()`.

Reformater le vecteur en matrice de taille Nw x N avec `reshape()`.

Décoder avec la fonction `rsdec()` (la taille de sortie doit être Nw x K).

Reformater pour obtenir un vecteur de taille 1 x (Nw x K) avec `reshape()`.

Convertir en entier pour avoir un vecteur de taille 1 x (Nw x K) avec la fonction `gf2dec()`.

Convertir en une matrice de bits de taille (Nw x K) x M avec `de2bit()`.

Reformater pour obtenir un vecteur de taille 1 x (Nw x K x M) avec `reshape()`.

Les codes MATLAB de la fonction `gf2dec()` et pour obtenir le tableau `table_gf2dec` sont les suivants :

Code pour la fonction `gf2dec()` :

```
function a_dec=gf2dec(a,tabb)
a_dec(1:length(a))=0;
nzg = find(a~=0);
a_dec(nzg)=tabb(log(a(nzg))+1);
end
```

Code pour le tableau `table_gf2dec` :

```
gf_nonzero=gf([1:2^M-1],M);
expformat=log(gf_nonzero)+1;
for ii=1:2^M-1
    table_gf2dec(expformat(ii))=ii;
end
```