TP de codage canal
module TS226

M. Ellouze et R. Tajan

1 Objectifs et évaluation

L'objectif de ce TP de codage canal est de simuler à l'aide du logiciel Matlab des chaînes de communications numériques codées afin d'évaluer notamment leurs performances en termes de probabilité d'erreur binaire, en terme de rendement, mais aussi en terme de débit.

Les TP se font en **binôme ou monôme** et l'évaluation porte sur une note de rapport et une note de travail continu (en séances).
Concernant le rapport, il ne doit pas excéder **15 pages**, vous devez fournir un document scientifique et technique, qui doit présenter votre travail, vos choix techniques et dans lequel **tous les résultats obtenus doivent être interprétés et commentés**. Les codes Matlab doivent également être transmis à votre enseignant. Ils doivent pouvoir être compris rapidement. Cela passe par l'utilisation de commentaires. Les commentaires doivent permettre de répondre au moins à la question : que fait la ligne de code ? Une attention particulière doit être portée à la lisibilité du programme. Les rapports et les codes doivent être rendus par l'intermédiaire de la plateforme Thor.

Vous trouverez les fichiers initiaux pour réaliser ce TP à l'adresse suivante `https://github.com/rtajan/TS226`.

2 Code convolutifs

Dans ce TP, vous allez vous intéresser au cas des communications numériques codées à l'aide d'un code convolutif. Vous devrez implémenter, par vous-même, les fonctions d'encodage et de décodage des codes convolutifs. L'architecture de communication à considérer est présentée sur la Figure 1.

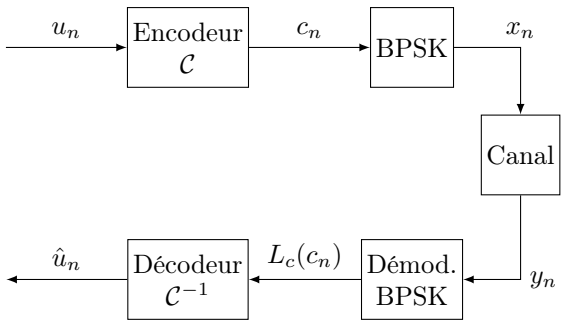

FIGURE 1 – Chaîne de transmission codée

2.1 Treillis

La communication toolbox de Matlab offre une structure de donnée permettant de représenter simplement les diagrammes d'états des codes convolutifs. Ces structures sont construites en utilisant la fonction `poly2trellis` de Matlab. Cette fonction prend en entrée une la longueur de contrainte $\nu = m + 1$ et le polynôme en notation octale sous forme de vecteur. Par exemple, la commande `poly2trellis(3,[7,5],7)` créant l'encodeur $(1, \frac{5}{7})_8$ renvoie la structure suivante :

— `numInputSymbols` : 2 (nombre de symboles possibles en entrée, 2^{n_b})

— `numOutputSymbols` : 4 (nombre de symboles possibles en sortie, 2^{n_s})

— `numStates` : 4 (nombre d'états possibles en sortie, 2^m)

— `nextStates` : tableau représentant la fonction suivante $\mathbf{s}_{n+1} = f(\mathbf{s}_n, u_n)$

État \ Entrée	0	1
0	0	2
1	2	0
2	3	1
3	1	3

TABLE 1 – Tableau représentant les états suivants à partir des états courants (lignes) et des symboles en entrée (colonnes).

— `outputs` : tableau représentant la fonction suivante $\mathbf{c}_n = g(\mathbf{s}_n, u_n)$

État \ Entrée	0	1
0	0	3
1	0	3
2	1	2
3	1	2

TABLE 2 – Tableau représentant les sorties à partir des états courants (lignes) et des symboles en entrée (colonnes).

2.2 Encodage (Matlab)

Écrire une fonction `cc_encode` ayant le prototype suivant `function c = cc_encode(u, trellis)`. Cette fonction possède 2 arguments en entrée :

— `u` : un vecteur de K symboles d'entrées.

— `trellis` : une structure représentant le treillis.

Cette fonction renvoie 1 sortie :

— `c` : le mot de code.

L'état initial sera systématiquement l'état 0 et le treillis devra être fermé. L étant le nombre de sections fermeture incluse et K le nombre de symboles du message. Comme le treillis est fermé, `c` devra être de taille $n_s L$.

2.3 Décodage de Viterbi (Matlab)

Écrire une fonction `viterbi_decode` ayant le prototype suivant `function u = viterbi_decode(y, trellis)`. Cette fonction renvoie `u` un vecteur de message de taille K.

— `y` : un vecteur de $n_s L$ observations du canal.

Astuces :

1. le choix de l'implémentation de l'algorithme de Viterbi est laissé libre. Cependant, il est souvent pratique de stocker les métriques de branches dans un tableau de taille $2^m \times (L+1)$.
2. vous pouvez utiliser l'outil `bertool` de Matlab pour tester votre décodeur.

2.4 Étude de l'impact de la mémoire du code (rapport)

Utiliser les codes développés précédemment pour réaliser une étude comparative de l'impact de la mémoire des codes convolutifs leurs performances. Pour cela, pour $K = 1024$, vous simulerez les performances de décodage des codes convolutifs suivants :

— $(2, 3)_8$

— $(5, 7)_8$

— $(13, 15)_8$

— $(133, 171)_8$

Sur un même graphique, tracer les taux d'erreur binaire (TEB) en fonction de $\frac{E_b}{N_0}$ [(1)] pour chacun des codes de la question précédente. Sur ce graphique, tracer aussi la courbe de la probabilité d'erreur binaire dans un scénario non-codé.

Pour chaque stratégie, relever le **gain de codage** sur les courbes obtenues. Le gain de codage est la différence en dB entre la valeur $\frac{E_b}{N_0}$ pour laquelle une stratégie non codée passe sous $P_b = 10^{-5}$ et la valeur $\frac{E_b}{N_0}$ pour laquelle une stratégie codée passe sous $P_b = 10^{-5}$.

Une autre caractéristique intéressante est le débit de sortie du récepteur. Ce débit est normalement déjà évalué dans par le script fourni. Pour chaque code, vous relèverez le débit lors du calcul de la dernière valeur de $\frac{E_b}{N_0}$.

Dans votre rapport, après avoir présenté les courbes de résultat, faites un tableau de synthèse récapitulatif explicitant pour chaque code :

— son rendement **lorsque le treillis est ouvert**

— sa mémoire et leur nombre d'états

— sa distance minimale (obtenue à l'aide de la fonction `distspec`)

— son gain de codage

— le débit de décodage

Réaliser votre étude comparative à l'aide du tracé du TEB en fonction de $\frac{E_b}{N_0}$ et du tableau récapitulatif.

(1). Les valeurs de $\frac{E_b}{N_0}$ iront de $-2dB$ à $10dB$ par pas de $0.5dB$. Le pas peut être ajusté en fonction des stratégies de codage. Il n'est pas nécessaire de simuler les points de $\frac{E_b}{N_0}$ ayant un TEB inférieur à $3 * 10^{-6}$.

2.5 Encodage récursif / encodage non récursif (rapport)

Tracer sur un même graphique les performances pour le décodage de Viterbi des codes convolutifs pour $K = 1024$ avec les encodeurs suivants :

— $(5, 7)_8$
— $(1, \frac{5}{7})_8$
— $(13, 15)_8$
— $(1, \frac{13}{15})_8$

Le décodage de Viterbi sera considéré treillis fermé. Dans votre rapport présentez les résultats obtenus, analysez les courbes et apportez une démonstration des résultats obtenus.

2.6 Prédiction des performances (Matlab + rapport)

Plusieurs méthodes existent pour déterminer les performances des codes convolutifs sans avoir à réaliser de longues simulations de Monte Carlo. Une de ces méthodes s'appelle la **méthode de l'impulsion** ; elle est donnée dans l'Algorithme 1.

Algorithme 1 Méthode de l'impulsion

Entrées : d_0 et d_1
Initialiser $\mathbf{v} = [0, 0, \ldots, 0]$ un vecteur contenant K éléments à 0.
Initialiser $\mathbf{x}_u = [0, 0, \ldots, 0]$ un vecteur contenant K éléments à 0.
Initialiser $\mathbf{y} = [1, 1, \ldots, 1]$ un vecteur contenant N éléments à 1.
Pour $l = 0$ à $K - 1$ **faire**
  $A = d_0 - 0.5$
  $\hat{\mathbf{x}}_u = \mathbf{x}_u$
  Tant que $\hat{\mathbf{x}}_u == \mathbf{x}_u$ et $A \leq d_1$ **faire**
    $A \leftarrow A + 1$
    $\mathbf{y} = [1, 1, \ldots, 1 - A, \ldots, 1]$, le terme $1-A$ étant sur la position du l^{ieme} bit systématique
    $\hat{\mathbf{x}}_u$ est obtenu en décodant $\mathbf{y}$
  Fin Tant que
  $v_i = \lfloor A \rfloor$, mise à jour du l^{eme} élément de $\mathbf{v}$.
Fin Pour
Soit $\mathcal{D}$ l'ensemble des entiers contenus dans $\mathbf{v}$.
Pour $d \in \mathcal{D}$, A_d est le nombre d'éléments de $\mathbf{v}$ égaux à d.
Une estimation du TEP est donnée par :

$$TEP \simeq \frac{1}{2} \sum_{d \in \mathcal{D}} A_d \text{erfc}\left(\sqrt{dR\frac{E_b}{N_0}}\right)$$

Implémenter la méthode de l'impulsion pour $d_0 = 1$ et $d_1 = 100$ et comparer les résultats obtenus aux courbes simulées précédemment.

3 Contacts

— Romain Tajan - romain.tajan@ims-bordeaux.fr
— Malek Ellouze - malek.ellouze@gmail.com