TP de codage canal
module TS226 – année 2018/2019

R. Tajan et P. Vallet

1 Objectifs et évaluation

L'objectif de ce TP de codage canal est de simuler à l'aide du logiciel Matlab plusieurs 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 doivent être au format NOM1_NOM2.pdf et les codes au format NOM1_NOM2.zip. Ces deux fichiers doivent être transmis à votre encadrant par Email au **plus tard deux semaines après la dernière séance de TP, sachant que la date exacte vous sera fixée par votre encadrant**.
Remarque : chaque jour de retard dans la remise des livrables sera sanctionné de 2 points en moins sur la note finale du TP.

2 Toolbox de communication

Vous trouverez les fichiers initiaux pour réaliser ce TP à l'adresse suivante `https://github.com/rtajan/TS226`. Ces fichiers s'appuient sur la toolbox de communications de Matlab afin de simuler efficacement la chaîne de communication. Tous les objets de la toolbox de communication ont des noms de la forme `comm.NomDeLObjet`. Par exemple vous pourrez remarquer les objets suivants :

- `comm.PSKModulator` : modulateur numérique M-PSK,
- `comm.PSKDemodulator` : démodulateur numérique M-PSK,
- `comm.AWGNChannel` : canal AWGN
- `comm.ErrorRate` : comptage des erreurs, calcul du TEB.
- `poly2treillis`
- `distspec`
- `comm.ConvolutionalEncoder`
- `comm.ViterbiDecoder`

— `comm.APPDecoder`
— `comm.RSEncoder`
— `comm.RSDecoder`
— `comm.BlockInterleaver`
— `comm.BlockDeinterleaver`
— `comm.MatrixInterleaver`
— `comm.MatrixDeinterleaver`

3 Code convolutifs

Dans cette première partie vous allez vous intéresser au cas des communications numériques codées à l'aide d'un code convolutif. L'architecture de communication à considérer est présentée sur la Figure 1.

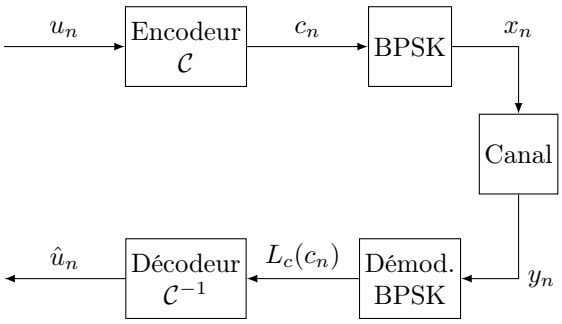

FIGURE 1 – Chaîne de transmission codée

3.1 Décodage de Viterbi

Dans cette partie nous allons étudier les performances des **codes convolutifs** lorsque le décodeur de Viterbi est utilisé. Pour effectuer cette comparaison, nous simulerons la transmission codée de messages binaires $\mathbf{u}$. Chaque message $\mathbf{u}$ est une trame composée de $N_p = 100$ **paquets** de $k = 330$ bits. Le vecteur $\mathbf{u}$ devra donc avoir une taille totale $K = 33000$. Vous paramètrerez les différents objets afin que le **treillis soit fermé**.

Le décodeur de Viterbi n'est pas à implémenter, vous utiliserez celui de la toolbox de communication de Matlab `comm.ViterbiDecoder` avec les propriétés suivantes :

— `'TrellisStructure'` : devra contenir une structure de treillis représentant le code.
— `'TracebackDepth'` : devra être choisie pour être toujours supérieure à 5 fois la longueur de contrainte du code étudié.
— `'TerminationMethod'` : type de fermeture du treillis (ici il doit être fermé).
— `'InputFormat'` : `'Unquantized'`, l'algorithme de Viterbi étudié travaillera sur des Logarithmes de rapports de vraisemblance (LLR) calculés par le démodulateur BPSK.

Pour une modulation BPSK, possédant des symboles équiprobables envoyés dans un canal sans mémoire, l'expression d'un LLR est donnée par la formule :

$$L_c(c_n) = \log \frac{\mathbb{P}(c_n = 0|y_n)}{\mathbb{P}(c_n = 1|y_n)} = \frac{2}{\sigma^2} y_n$$

où σ^2 est la variance du bruit supposée connue du récepteur. L'expression de ce LLR étant directement proportionnelle à y_n les performances de cet algorithme de Viterbi seront identiques à celles de l'algorithme vu en cours (travaillant directement sur y_n). Les tâches à effectuer pour cette partie sont les suivantes :

1. Pour les codes $(2,3)_8$, $(7,5)_8$, $(13,15)_8$ et $(133,171)_8$, faites un tableau récapitulatif explicitant : leur rendement **lorsque le treillis est ouvert**, leur rendement **lorsque le treillis est fermé**, leur mémoire, leur nombre d'états et leur distance minimale (obtenue à l'aide de la fonction `distspec`).
2. **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.
3. Sur le graphique de la question précédente relever le **gain de codage**. 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 $P_b = 10^{-5}$. Ajouter une colonne à votre tableau dans laquelle vous listerez ces résultats. Commenter le tableau obtenu.
4. 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 $\frac{E_b}{N_0}$. Ajouter une colonne à votre tableau dans laquelle vous listerez ces résultats.
5. Faire une synthèse du tableau avec les points forts et les points faibles des différents codes.

4 Codes concatennés

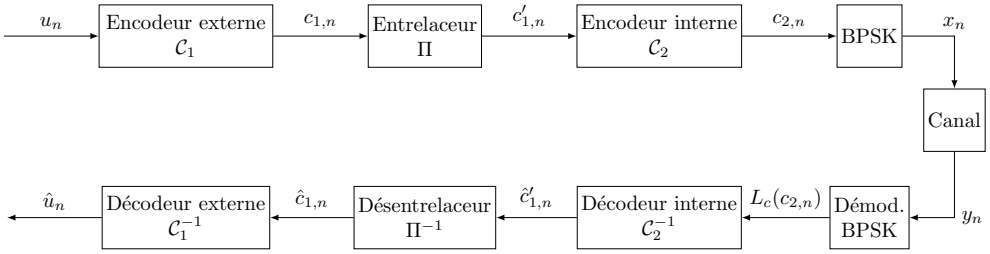

FIGURE 2 – Chaîne de transmission codée

Dans cette partie, on considère la concaténation d'un code de Reed Solomon $\mathcal{C}_1$ et d'un code et convolutif $\mathcal{C}_2$ représentée en Figure 2. Une trame de message **u** est toujours composée de 100 paquets de 330 bits. Chaque paquet est d'abord encodé par l'encodeur de $\mathcal{C}_1$ pour donner une séquence de 100 mots de codes RS. Cette trame est ensuite entrelacée puis ré-encodée par un code convolutif.

— Le code Reed Solomon considéré dans ce TP est le code RS(55,63,8). Il travaille sur un alphabet de 64 symboles composés de 6 bits. Ce code encode des paquets 55 symboles soit de $6 \times 55 = 330$ bits en des mots de codes de 63 symboles soit de $6 \times 63 = 378$ bits. Il est

(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 $3 * 10^{-6}$.

capable de corriger toute combinaison de moins de 4 erreurs symboles au sein d'un mot de code.

— Le code convolutif considéré est le code $(133, 171)_8$ non récursif et non systématique.

— L'entrelaceur considéré est de type ligne-colonne de 378 lignes et 100 colonnes. Le principe de fonctionnement est simple. Une matrice de 378 lignes et 100 colonnes est d'abord remplie par colonne avec les bits de $\mathbf{c}_1$. Cette matrice est ensuite lue en ligne pour donner $\mathbf{c}'_1$.

Les tâches à effectuer pour cette partie sont les suivantes :

6. **Sur un même graphique**, tracer le TEB en fonction de $\frac{E_b}{N_0}$ pour les stratégies de codage suivantes :
   — Code concaténé décrit en introduction de cette partie,
   — Code concaténé sans entrelaceur,
   — Code convolutif utilisé seul (résultat de la partie précédente),
   — Pas de codage (résultat fourni avec les fichiers de départ)

   Commenter votre résultat.
7. Pour chacune des stratégies proposées, vous calculerez les débits de sortie ainsi que les valeurs de $\frac{E_b}{N_0}$ en dB permettant de garantir un TEB inférieur à 10^{-5}. Faire le graphique du débit de réception en fonction du $\frac{E_b}{N_0}$ relevé. **Commenter votre résultat.**

5 Turbocodes parallèles

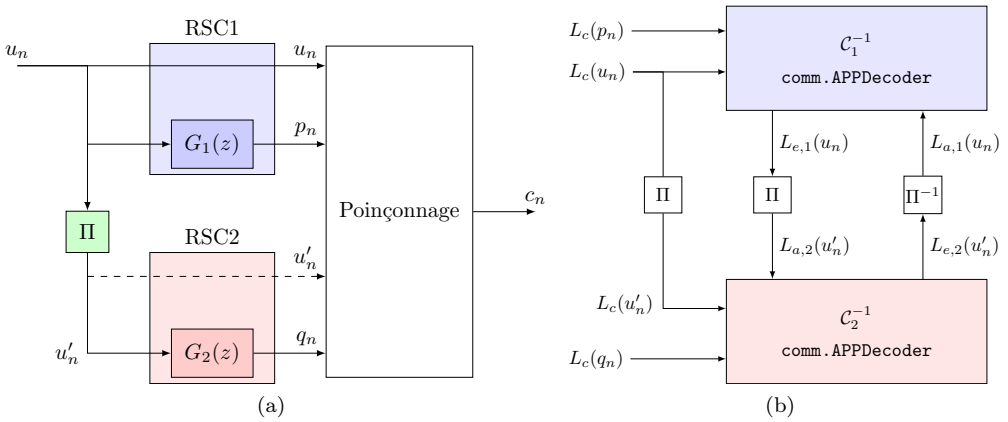

FIGURE 3 – Encodage et décodage des turbocodes

Dans cette partie, nous considérerons un **turbocode parallèle** de rendement $R = 1/2$, où RSC_1 et RSC_2 sont des encodeurs récursifs et systématiques du code $(7, 5)_8$. Les encodeurs et décodeurs sont respectivement donnés dans les Figures 3a et 3b. Les codes constituants étant de rendement $R_1 = R_2 = 1/2$, le rendement avant poinçonnage est de $1/4$. Afin de garantir $R = 1/2$, nous considèrerons la matrice de poinçonnage P suivante

$$P = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Dans le cours, nous avons vu que le décodage du Maximum a Posteriori binaire calcule le LLR suivant

$$L_1(u_n) = \log \frac{p(u_n = 0|\mathbf{y})}{p(u_n = 1|\mathbf{y})}.$$

Après calculs, on peut montrer que ce LLR peut s'écrire comme la somme du LLR a priori $L_{a,1}(u_n)$, du LLR canal $L_c(u_n)$ et d'un LLR appelé extrinsèque $L_{e,1}(u_n)$, c'est-à-dire :

$$L_1(u_n) = L_{a,1}(u_n) + L_c(u_n) + L_{e,1}(u_n).$$

La valeur $L_{e,1}(u_n)$ est ensuite fournie au décodeur $\mathcal{C}_2^{-1}$ pour qu'il effectue son décodage. Sur le schéma de principe de la Figure 3b, vous remarquerez aussi que $L_{a,1}(u_n)$ est en fait donné par $\mathcal{C}_2^{-1}$ et $L_c(u_n)$ est donné par le démodulateur. Les décodeurs $\mathcal{C}_1^{-1}$ et $\mathcal{C}_2^{-1}$ vont donc s'échanger des LLR de façon itérative avant le décodage définitif.

Une "itération" correspond aux étapes $\mathcal{C}_2^{-1} \rightarrow \mathcal{C}_1^{-1}$. En conséquence, I itérations de décodage correspondent à la séquence de décodage

$$\mathcal{C}_1^{-1} \rightarrow \underbrace{\mathcal{C}_2^{-1} \rightarrow \mathcal{C}_1^{-1} \rightarrow \cdots \rightarrow \mathcal{C}_2^{-1} \rightarrow \mathcal{C}_1^{-1}}_{\mathcal{C}_2^{-1}\rightarrow\mathcal{C}_1^{-1}\text{répété I fois}}$$

Une fois ces itérations effectuées, la décision est effectuée à partir du signe de $L_1(u_n)$.

Le décodeur MAP-bit implémenté sous Matlab est appelé `comm.APPDecoder`. Il calcule $L_{e,1}(u_n)$ à partir de $L_{a,1}(u_n)$ et $L_c(u_n)$. **Une petite subtilité est que le décodeur `comm.APPDecoder` considère un mapping BPSK $0 \rightarrow -1$, $1 \rightarrow 1$ il faudra donc penser à changer le signe des valeurs de $L_c(u_n)$ à son entrée ainsi qu'au moment de la décision.** Les autres paramètres de `comm.APPDecoder` à prendre en compte sont les suivants :

- `TrellisStructure` : devra contenir une structure de treillis représentant le code.
- `TerminationMethod` : type de fermeture du treillis (ici il doit être **ouvert**).
- `Algorithm` : `'True APP'`, on considèrera un algorithme Log-MAP.
- `CodedBitLLROutputPort` : mettre à `false`.

8. Implémenter un encodeur/décodeur turbocode **sans utiliser `comm.TurboEncoder` ni `comm.TurboDecoder`**. L'entrelaceur considéré sera un entrelaceur aléatoire généré obtenu par la commande `comm.BlockInterleaver`. Il est fortement recommandé de gérer le poinçonnage manuellement.
9. Tracer, **sur un même graphique**, le TEB en fonction de $\frac{E_b}{N_0}$ pour $I = 0$, $I = 3$ et $I = 8$ itérations. **Commenter votre résultat.**
10. Tracer, **sur le graphique précédent**, le TEB en fonction de $\frac{E_b}{N_0}$ pour $I = 0$, $I = 3$ et $I = 8$ itérations en changeant `Algorithm` pour `'Max'`. **Commenter votre résultat.**
11. Ajouter aux graphiques obtenus dans la Section 4 (graphiques de TEB et de débit) les performances du turbocode pour les décodeurs `'True APP'` et `'Max'` à 8 itérations. **Commenter votre résultat.** Un point intéressant à intégrer à votre discussion est que le théorème de Shannon pour le codage de canal énonce qu'une communication fiable est possible pour un rendement $R = \frac{1}{2}$ sur un canal AWGN avec un rapport signal à bruit de $\frac{E_b}{N_0} = 0dB$.

6 Contacts

— Romain Tajan - romain.tajan@ims-bordeaux.fr
— Pascal Vallet - pascal.vallet@ims-bordeaux.fr