PROJET 2 TELECOMMUNIVATION ET TRAITEMENT DU SIGNAL

Introduction au codage canal

Étude d'un codage de type bloc et d'un codage convolutif
Comparaison chaine codée/chaine non codée

Première Année - Département Sciences du Numérique

2024-2025

1 Introduction

Soit la chaine de transmission présentée dans la figure suivante.

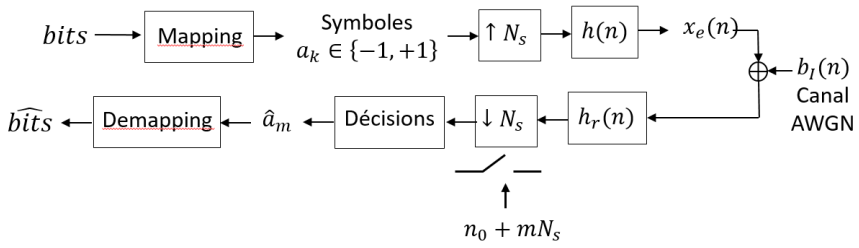

Figure 1: Chaîne de transmission passe-bas équivalente à la chaine de transmission sur porteuse de type BPSK.

Elle représente la chaine passe-bas équivalente associée à une transmission BPSK sur canal AWGN : $x_e(t)$ représente l'enveloppe complexe associée au signal modulé sur porteuse (voie en phase uniquement en BPSK $x_e(t) = I(t)$), tandis que $b_I(t)$ représente l'enveloppe complexe associée au bruit introduit par le canal de propagation (voie en phase uniquement également).

L'objectif de ce travail va être d'évaluer l'impact de l'ajout de codage canal à cette chaine de transmission, en termes d'efficacité spectrale et d'efficacité en puissance.

La figure suivante présente la chaine à implanter avec codage canal.

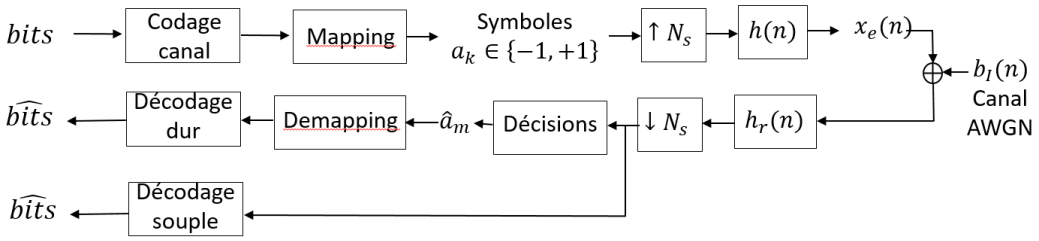

Figure 2: Chaîne de transmission passe-bas équivalente à une chaine de transmission sur porteuse codée, de type BPSK.

L'opération de codage canal consiste à ajouter, selon une loi donnée, de la redondance à l'information à transmettre afin de la protéger contre les erreurs de transmission. Afin de détecter les erreurs le décodeur teste si la loi de codage utilisée à l'émission (par le codeur) est respectée ou pas en réception.

Les codes peuvent être classés en deux grandes familles : les codes en blocs et les codes convolutifs.

Le décodage optimal se fait en utilisant le maximum de vraisemblance et pourra se faire de deux manières : décodage souple (soft decoding) ou décodage dur (hard decoding). Lorsque le décodage optimal est trop complexe à implanter on pourra utiliser des algorithmes sous optimaux.

Nous vous demanderons d'implanter un exemple de codage bloc (code de Hamming), avec décodage souple et dur par maximum de vraisemblance et un exemple de codage convolutif (code $(3, 1/2)$), avec décodage souple et dur en utilisant l'algorithme de Viterbi qui permet de réduire la complexité du décodage.

2 Codes en blocs linéaires

2.1 Définitions

Nous nous limiterons à des codes constitués d'éléments prenant leurs valeurs dans l'ensemble $\{0, 1\}$ noté F_2. Un **codage bloc** est alors une application g de F_2^k vers F_2^n qui associe à un **mot d'information** constitué de k éléments binaires : $\mathbf{i} = [i_0...i_{k-1}]$ un **mot de code** constitué de n éléments binaires : $\mathbf{c} = g(\mathbf{i}) = [c_0...c_{n-1}]$, avec $n > k$ pour satisfaire à la condition de redondance :

$$\begin{aligned} g : & F_2^k \to F_2^n \\ & \mathbf{i} \to \mathbf{c} = g(\mathbf{i}) \end{aligned}$$

$n-k$ représente le nombre d'**éléments de redondance** introduits par le code et $\eta = \frac{k}{n}$ est appelé **rendement du code**.
Un code bloc est dit linéaire si les 2^k blocs de n élémets binaires (mots de code) constituent un sous espace vectoriel de F_2^n de dimension k.
Les **opérations de codage** sont réalisées à partir des opérations OU exclusif ($\oplus$) et ET logique ($\times$) entre éléments binaires (voir tableau suivant). F_2 muni de ces deux opérations constitue un corps.

b_0	b_1	$b_0 \oplus b_1$	$b_0 \times b_1$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2.2 Principe du codage

L'application g étant linéaire, en exprimant $\mathbf{i}$ dans une base $(e_0, ...e_{k-1})$ de F_2^k : $\mathbf{i} = \sum_{p=0}^{k-1} i_p e_p$, on peut écrire

$$\mathbf{c} = g(\mathbf{i}) = \sum_{p=0}^{k-1} i_p g(e_p)$$

En exprimant $g(e_p)$ dans une base $(v_0, ...v_{n-1})$ de F_2^n : $g(e_p) = \sum_{q=0}^{n-1} g_{pq} v_q$, $g_{pq} \in F_2$, on obtient alors la matrice génératrice $k \times n$ du code :

$$G = \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ \cdots & & & \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix}$$

et tout mot de code $\mathbf{c}$ associé à $\mathbf{i}$ peut être obtenu par la relation matricielle suivante :

$$\mathbf{c} = \mathbf{i}G$$

2.3 Code systématique

La matrice G (non unique) est de rang k et il est donc toujours possible de l'écrire telle que :

$$G = [I_k \ P]$$

où I_k est la matrice identité $k \times k$ et P une matrice $k \times n$ utilisée pour calculer les $n-k$ bits de redondance. $\mathbf{c}$ peut alors s'écrire

$$\mathbf{c} = [\mathbf{i} \ P]$$

et le code est dit systématique : les bits d'information apparaissent "en clair" dans le mot de code. Ce sont les k bits de poids fort.

2.4 Principe du décodage

Le décodage optimal se fait en utilisant une règle du maximum de vraisemblance : à partir d'un mot de code reçu **c'**, nous déciderons du mot de code émis $\widehat{\mathbf{c}}$ (et donc des bits d'information associés) qui est le plus vraisemblable parmi tous les mots de code $\mathbf{c}$ possibles du dictionnaire utilisé :

$$\widehat{\mathbf{c}} = \underset{c}{argmax} \ P[c'/c]$$

Dans un contexte de canal à bruit additif Gaussien, le mot de code le plus vraisemblable correspond également au mot de code le plus proche du mot de code reçu, au sens d'une certaine distance. Cette distance sera différente selon que l'on souhaite mettre en place un décodage dur ou souple (voir figure 2) :

- **Dans le cas d'un décodage dur**, on calculera une distance de Hamming entre les bits du mot de code reçu après décisions et les bits de tous les mots de code possibles du dictionnaire utilisé. La distance de Hamming entre deux mots composés d'éléments binaires est le nombre d'emplacements où les deux mots possèdent des éléments binaires différents. Elle pourra être obtenue en déterminant le nombre de 1 dans la somme ($\oplus$) des deux mots binaires.
- **Dans le cas d'un décodage souple**, on caculera une distance euclidienne entre les symboles correspondant au mot de code reçu et les symboles correspondant aux mots de code du dictionnaire utilisé. Notons que rechercher le minimum de distance euclidienne revient à rechercher un maximum de corrélation entre les symboles correspondant au mot de code reçu et les symboles correspondant aux mots de code du dictionnaire utilisé. Vous pourrez utiliser au choix la recherche de la distance euclidienne minimale ou bien la recherche de la corrélation maximale.

Lorsque le décodage optimal est trop complexe à implanter on pourra utiliser des algorithmes sous optimaux.

2.5 Implantation sous Matlab d'un code de Hamming

2.5.1 Codage de Hamming à implanter

Le code de Hamming est un code bloc pour lequel la matrice de contrôle de parité H (matrice duale de la matrice génératrice : $GH^T = 0$, $H = \left[P^T \ I_{n-k}\right]$ si $G = [I_k \ P]$) contient tous les représentations binaires des nombres de 1 à n.

Nous vous demanderons d'implanter un code de Hamming de paramètres $k = 4$, $n = 7$, pour lequel on peut donner la matrice génératrice suivante :

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

Le décodage se fera en utilisant le maximum de vraissemblance, avec des distances calculées différemment selon que vous mettrez en place un décodage dur ou souple (voir précédemment).

2.5.2 Travail à réaliser

Implantez la chaine passe-bas équivalente de la figure 1 (BPSK) pour transmettre un débit binaire de 3000 bits par seconde, en utilisant une fréquence d'échantillonnage de 12 kHz. Vous pouvez choisir d'utiliser un filtre de mise en forme rectangulaire de durée égale à la durée symbole ou bien un filtre en racine de cosinus surélevé. Le démodulateur devra être optimisé conjointement avec le modulateur proposé.

A partir de cette chaine implantée :

1. Vérifiez, dans un premier temps, que, sans bruit, le TEB est bien nul et, qu'avec bruit, vous obtenez bien le TEB théorique de la chaine étudiée.

2. En utilisant la matrice génératrice du code de Hamming proposée, ajoutez le codage de Hamming à la chaine précédente et :

   (a) Tracez le TEB binaire obtenu pour la transmission, en fonction du rapport signal à bruit par bit, $\frac{E_b}{N_0}$, à l'entrée du récepteur, pour un décodage dur.

   (b) Tracez le TEB binaire obtenu pour la transmission, en fonction du rapport signal à bruit par bit, $\frac{E_b}{N_0}$, à l'entrée du récepteur, pour un décodage souple.

   (c) Comparez, en les traçant sur une même figure, les TEBs obtenus pour la chaine de transmission sans codage canal, avec codage de Hamming et décodage dur, avec codage de Hamming et décodage souple.

   (d) Quel est l'impact du codage canal sur la chaine de transmission, en termes d'efficacité en puissance ? en termes d'efficacité spectrale ?

   (e) vous pouvez observez l'apport du codage sur la transmission d'une image en utilisant le code *Manipulation_ image.m* fourni, qui permet de transformer une image png en une suite binaire venant remplacer la suite de bits générée par la fonction Matlab *randi.m* et de retrouver l'image à partir d'une suite binaire.

3 Codes convolutif

3.1 Définitions

Les **codes convolutifs**, parfois également appelés codes convolutionnels, appliquent des convolutions logiques à la séquence de bits d'entrée. Ces convolutions logiques sont obtenues en utilisant des registres à décalage dont certaines sorties sont ajoutées modulo 2 (addition dans F_2 ou OU exclusif : $\oplus$). Il est possible de considérer des blocs de k bits en entrée pour obtenir un code de rendement k/n, n représentant le nombre de bits de sortie obtenus pour chaque bloc de k bits en entrée. Il faut alors utiliser plusieurs registres à décalage connectés entre eux. Nous nous limiterons ici à l'utilisation d'un seul registre à décalage, et donc à des codes de rendements $1/n$: n bits seront obtenus en sortie pour chaque bits en entrée ($k = 1$).

La figure suivante donne l'exemple du code convolutif de rendement $1/2$ que nous vous demanderons d'implanter.

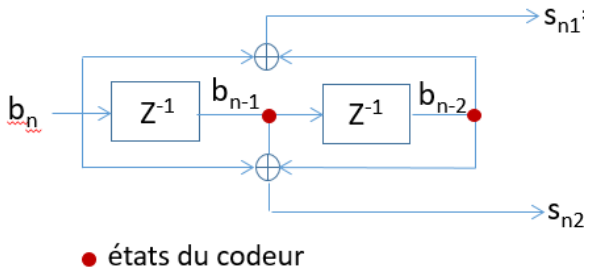

Figure 3: Exemple de codeur convolutif $(3, 1/2)$

Deux bits de sortie sont calculés pour chaque bit d'entrée de la manière suivante :

$$s_{n_1} = b_n \oplus b_{n-2}$$

$$s_{n_2} = b_n \oplus b_{n-1} \oplus b_{n-2}$$

De manière plus générale nous pouvons écrire la $i^{\text{ième}}$ sortie d'un codeur convolutif de la manière suivante

$$s_{n_i} = \sum_{j=0}^{K-1} g_i(j) b_{k-j}$$

On retrouve bien un produit de convolution, définissant un filtre de type RIF, avec une somme de F_2, c'est-à-dire modulo 2. Les codeurs convolutifs introduisent donc un effet de mémoire : les bits de sortie dépendent des $m = K - 1$ bits d'entrée précédents (en plus du bit d'entrée actuel). m représente la **mémoire du code**, tandis que $K = m + 1$ est appelé **longueur de contrainte du code**.

Les coefficients $g_i(j)$, $i = 1, ..., n$, $j = 0, ..., K - 1$, définissent le codeur. On donne les $\mathbf{g_i} = [g_i(0)...g_i(K - 1)]$, $i = 1, ..., n$:

- soit sous la forme de vecteurs de coefficients binaires, représentant les sorties du registre à décalage qui sont connectées à l'additionneur pour donner chaque bit de sortie du codeur. 1 représente une sortie connectée, tandis que 0 représente une sortie qui ne l'est pas. Dans l'exemple de la figure 3 on aurait, par exemple $g_1 = [1\ 0\ 1]$.
- soit avec un codage en octal. Dans l'exemple de la figure 3 on aurait, par exemple $g_1 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5_{octal}$.
- soit sous la forme de polynômes générateurs : $g_i = \sum_{j=0}^{K-1} g_i(j) D^j$, D représentant un délai, ou retard, de 1 bit (équivalent du z^{-1} de la transformée en z). Dans l'exemple de la figure 3 on aurait, par exemple $g_1 = 1 + D^2$.

3.2 Représentation des codes convolutifs

La sortie d'un codeur convolutif de rendement $1/n$ dépend de son entrée et du contenu d'un registre à décalage introduisant une mémoire de m. On peut considérer que c'est une machine d'états finie avec 2^m états différents. Ce qui permet de représenter le code de manière graphique sous différentes formes (arbre, treillis, diagramme d'état).

Nous représentons ci-dessous une description en treillis du codeur de la figure 3.

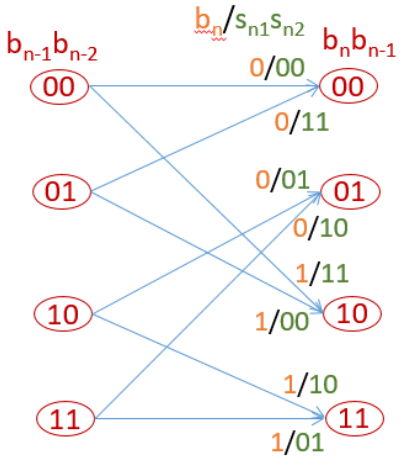

Figure 4: Représentation en treillis du codeur convolutif $(3, 1/2)$ de la figure 3

Un état du codeur est défini par les bits du registre à décalage qui sont conservés pour un nouveau bit entrant (identifiés en rouge sur les figures figures 3 et 4). Dans notre description en treillis du codeur nous allons représenter toutes les évolutions possibles au cours du temps des états du codeur. Par rapport à une représentation en diagramme d'état, qui donne également la manière dont les états du codeur communiquent entre eux, notre représentation fait intervenir le temps en positionnant à gauche les états du codeur à un instant n ($b_{n-1}\ b_{n-2}$) et à droite les états du codeur à l'instant $n + 1$ (($b_n\ b_{n-1}$). Sur les flèches se trouvent le bit entrant permettant de passer de l'état à l'instant n à l'état l'instant $n + 1$ à et les deux bits de sortie correspondant : $b_n/s_{n_1} s_{n_2}$.

3.3 Décodage des codes convolutifs

Comme pour les codes en blos, le décodage d'un code convolutif va utiliser le maximum de vraisemblance en recherchant, pour une séquence reçue de longueur N, $Y = [y_0...y_{N-1}]$, la séquence binaire émise la plus vraisemblable parmi toutes les séquences possibles de longueur N :

$$\widehat{X} = [x_0...x_{N-1}] = \underset{X}{argmax}\ P[Y/X]$$

. Une fois la séquence émise en sortie du codeur, X, retrouvée on en déduit la séquence de bits d'information non codée.

Dans un contexte de canal à bruit additif Gaussien, la séquence la plus vraisemblable correspond également à la séquence la plus proche de la séquence reçue, au sens d'une certaine distance, qui pourra être, comme nous l'avons vu pour les codes en blocs, une distance de Hamming (décodage dur ou hard decoding) ou une distance euclidienne (décodage souple ou soft decoding).

La séquence la plus vraisemblable correspondant à un chemin particulier dans le treillis associé au code. Pour une séquence reçue de N bits il faudra donc effectuer la comparaison avec $2^m \times 2^N$ séquences possibles (2^m est le nombre d'états du codeur). Le coût calculatoire est rapidement prohibitif (exemple pour une séquence de 20 bits avec le codeur donné en exemple qui comporte 4 états ($m = 2$) : $2^m \times 2^N = 4\ 194\ 304$). Heureusement des algorithmes permettent de contourner cette difficulté. C'est le cas de l'algorithme de Viterbi qui est utilisé pour décoder des codes de longueur de contrainte peu élevées ($K \leq 7$) et que nous vous demanderons d'implanter sous Matlab.

3.4 Algorithme de Viterbi

Le principe de l'algorithme va consister à ne pas tester de manière exhaustive toutes les séquences pour trouver la plus vraisemblable mais à ne conserver, à chaque étape n du treillis, pour chaque état, que les chemins qui donnent la distance cumulée (depuis l'instant 0) la plus faible pour y arriver. On appelle ces chemins les chemins survivants.
La figure 5 présente un exemple de décodage souple de Viterbi pour une séquence de symboles BPSK bruités reçue égale à $[0.4\ \ -0,2\ \ -1,4\ \ -0,5\ \ 0,3\ \ -0,9\ \ 1,2\ \ 0,6]$.

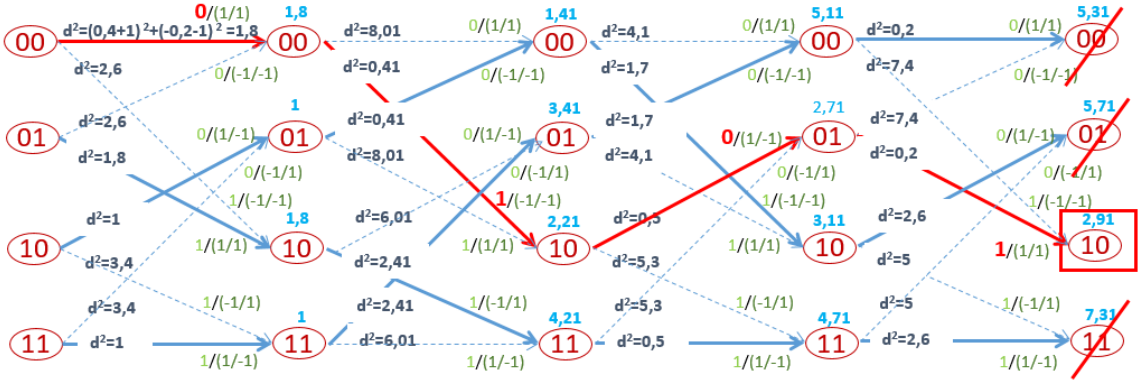

Figure 5: Exemple de décodage de Viterbi en soft decision

A chaque étape du treillis, pour chaque état possible, une distance euclidienne est calculée entre les deux symboles bruités reçus et les deux symboles BPSK possiblement émis sur chaque chemin memant à l'état considéré (voir figure 4). Les deux distances calculées (2 chemins mènent à chaque état) sont ajoutées aux distances cumulées associées aux deux états précédents (ceux d'où on vient). Seul le chemin donnant la distance cumulée la plus faible est conservé (chemin survivant, représenté en train plein), tandis que l'autre chemin est oublié (représenté en trait pointillé). En fin d'algorithme (dernier groupe de 2 symboles reçus) on regarde quelle est la distance cumulée la plus faible et l'état correspondant devient le point de départ à la remontée dans le treillis pour retrouver les bits émis (étape de traceback), donnant ici [0 1 0 1] .

Le temps de calcul en utilisant l'algorithme de Viterbi pour une séquence de longueur N est de $N \times 2 \times 2^m$, à comparer à $2^m \times 2^N$ par calcul direct.

En pratique, il est possible de réaliser le décodage de Viterbi sur des trames de longueurs finies si la transmission code des trames indépendamment les unes des autres. Cependant, dans les systèmes qui transmettent les données de façon continue, le décodeur ne peut pas attendre d'avoir reçue toute la séquence pour prendre une décision, le retard introduit serait trop grand et le décodeur nécessiterait trop de mémoire. Dans ce cas le décodeur prend une décision avec un retard maximal d'environ 5 fois la longueur de contrainte du code. En effet, on montre qu'alors la dégradation, en termes de probabilité d'erreur, introduite par la limitation à cette longueur finie de la séquence, est généralement négligeable (les chemins survivants convergent presque toujours vers un même chemin).
Dans le projet, par souci de simplicité, nous vous demanderons de réaliser le décodage sur l'ensemble de la séquence reçue, en supposant que l'état initial du codeur est 00 **et en considérant une longueur de séquence égale à, au moins,** 5 **fois la longueur de contrainte du code.**

3.5 Travail à réaliser

Reprenez la chaine passe-bas équivalente réalisée précédemment sans codage canal (figure 1). Assurez-vous qu'elle fonctionne correctement en retrouvant le TEB théorique. Ajoutez-y le code convolutif $(3, 1/2)$ défini par la figure 3.

1. Quelle est la mémoire de ce code ?
2. Donnez les expressions de g_1 et g_2 définissant ce code dans les 3 représentations possibles (vecteur d'éléments binaires, octal, polynômes générateurs).
3. Soit la séquence de bits émise suivante : [0 1 1 1 0 0 1 0]
   (a) Déterminez la séquence de bits en sortie du codeur.
   (b) En supposant qu'il n'y a pas de canal de transmission, c'est-à-dire que la séquence de bits reçue à l'entrée du décodeur est égale à celle émise en sortie du codeur, donnez, comme dans l'exemple de la figure 5, le treillis associé à un décodage de Viterbi mais de type décisions dures.
4. Implantez le décodage de Viterbi de type dur (hard decoding) et tracez le TEB binaire obtenu pour la transmission, en fonction du rapport signal à bruit par bit, $\frac{E_b}{N_0}$.
   Remarque : l'étape précédente peut vous permettre de tester, dans un premier temps, l'implantation de votre décodeur.
5. Comparez, en les traçant sur une même figure, les TEBs obtenus pour la chaine de transmission sans codage canal, avec codage convolutif $(3, 1/2)$ et décodage de Viterbi de type dur, avec codage convolutif $(3, 1/2)$ et décodage de Viterbi de type souple.
6. Quel est l'impact du codage canal sur la chaine de transmission, en termes d'efficacité en puissance ? en termes d'efficacité spectrale ?
7. Comparez l'impact des codages de Hamming et convolutifs implantés sur la chaine de transmission en termes d'efficacité en puissance et d'efficacité spectrale.

4 Références

[1] "Introduction aux communications numériques", M. Joindot, A. Glavieux, Dunod
[2] "Eléments de communications numériques", J.C. Bic, D. Duponteil, J.C.Imbeaux, Dunod