Rapport - Codage Canal

Amaury Demesy Yan Rabefaniraka

1^{er} Juin 2025

Contents

Remarque : Fichier engendré avec:

```
pandoc -o rapport-demesy_rabefaniraka.pdf rapport-demesy_rabefaniraka.md --highlight=my_style.theme
```

Introduction

Le codage canal est l'ajout selon une loi donnée de redondance d'information au sein de notre transmission. Cela sert à minimser les erreurs de transmission en vérifiant à la réception que la loi de codage utilisée par l'émetteur a bien été respectée.

L'objectif de ce projet est d'étudier l'impact de l'ajout du codage canal sur l'efficacité spectrale et l'efficacité en puissance d'une chaîne de transmission.

Pour ce faire, seront implémentées le **codage en bloc linéaires** avec le *code de Hamming*, puis le **codage convolutif** avec *l'algorithme de Viterbi*.

Pré-requis (Transmission passe-bas équivalente BPSK)

Les implémentations seront limitées à une modulation binaire (éléments se trouvant dans $0, 1$). Nous utiliserons plus spécifiquement un mapping ***BPSK*** (ici c'est équivalent à un mapping 2-ASK avec deux valeurs opposées sur l'axe des réels), pour un débit binaire de $3000 bits/sec$, avec une fréquence d'échantillonnage de 12kHz. Nous choisirons un filtre de mise en forme et de réception rectangulaire de durée Ts.

Validation de la chaîne de transmission

Avec $n_0 = 4$, on remarque que notre chaîne de transmission sans bruit a bien un TEB nul.

```
TEB_simule_BPSK =
     0     0     0     0     0     0     0
```

Figure 1: TEB sans bruit

Là où, avec bruit, il se rapproche bien du TEB théorique pour un mapping binaire BPSK.

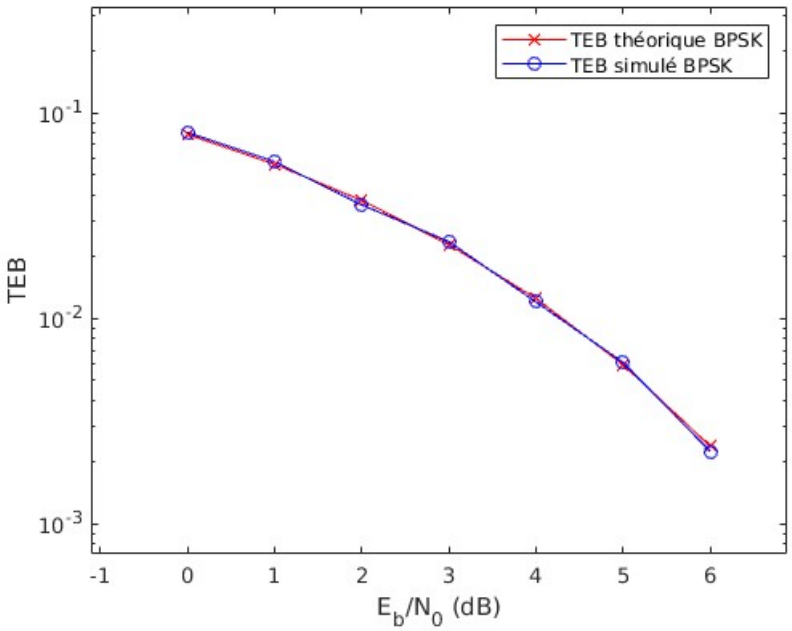

Figure 2: Tracé du TEB avec bruit

Implémentation en codes à bloc linéaires

Le code de Hamming crée de la redondance en associant à un **mot d'information** de k élemnts binaires $i = [i_0...i_{k-1}]$, un **mot de code** de n éléments binaires $c = [c_0..; c_{n-1}]$ (évidemment $k < n$). On obtient ce mot de code à l'aide d'une matrice génératrice de taille $k \times n$ appelée G. Elle est l'expression dans la base d'arrivée de l'image de l'application $i \to c = g(i)$.

Ici, il est implémenté en groupant d'abord les bits de départ en blocs de 4 bits d'information, puis en les multipliant par la matrice génératrice G pour obtenir des mots codés de 7 bits (*i.e.* $c = iG$), ajoutant ainsi la redondance nécessaire.

À la réception, à partir du code reçu c', on décide du mot de code émis le plus vraisemblable en utilisant une règle du **maximum de vraisemblance**: Pour cela, on établit une distance entre le mot de code reçu et tous les mots de code possibles dans un dictionnaire; le mot le plus plausible est donc celui qui maximise notre critère de vraisemblance, c'est-à-dire celui qui minimise sa distance avec le mot reçu.

On a deux distances différentes selon le décodage:

Cas du décodage dur et son impact

Le **décodage dur** calcule une *distance de Hamming* entre les bits reçus et ceux possibles. Cette distance est définie comme étant le nombre de bits de même rang mais différents. Ici, le décodage est essentiellement implémenté comme suit:

```
distance_hamming = sum(code_mots ~= mot_courant, 2);
[~, index_min] = min(distance_hamming); %min retourne l'indice en 2eme pos
bits_decodes_BPSK_hamming_dur(i, :) = dico_mots(index_min, :);
```

On obtient le tracé du TEB suivant:

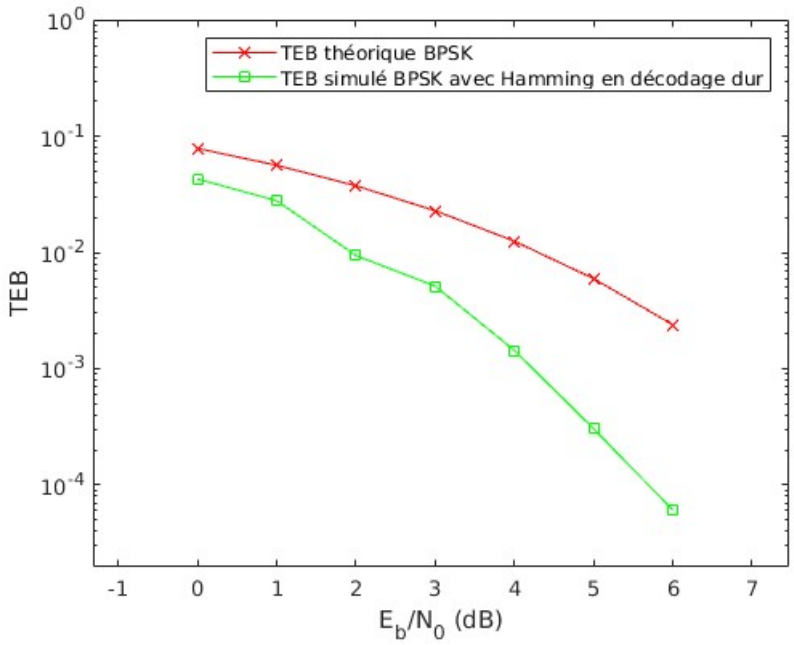

Figure 3: Tracé du TEB avec codage de Hamming dur

Cas du décodage souple et son impact

Le **décodage souple** calcule une *distance euclidienne* entre les **symboles** reçus et ceux possibles. Ici, le décodage est essentiellement implémenté comme suit:

```
mot_courant = Signal_echantillonne_BPSK_hamming_reshaped(i, :);
distance_euclidienne = pdist2(mot_courant, code_mots, 'euclidean');
[~, index_min] = max(distance_euclidienne);
bits_decodes_BPSK_hamming_souple(i, :) = dico_mots(index_min, :);
```

On obtient le tracé du TEB suivant:

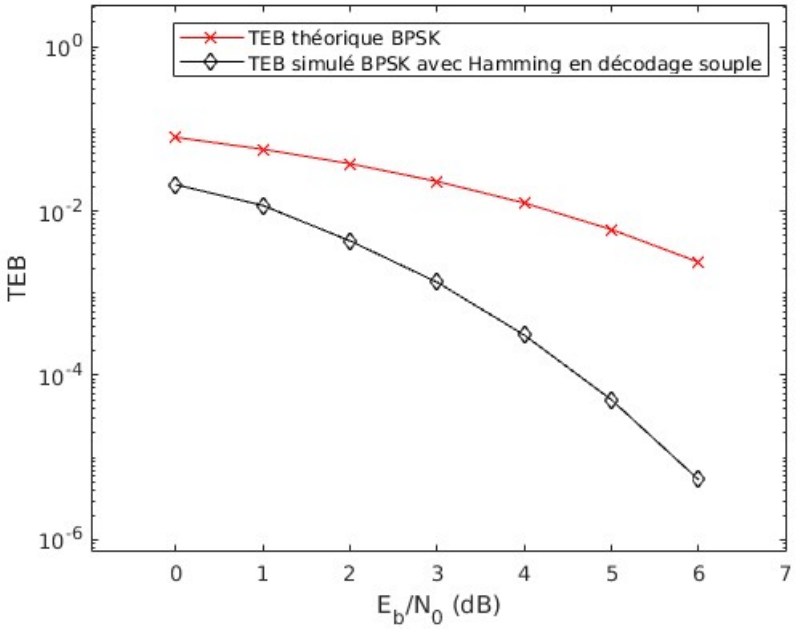

Figure 4: Tracé du TEB avec codage de Hamming souple