TP4 - Codes convolutionnels

Axel REMACK - Stacy VARLOTEAUX

17 octobre 2021

1 Code convolutionnel C (2, 1, 4)

1.1 Algorithme de décodage du code convolutionnel (Fonction *GSM_decode*)

Pour cette fonction, nous avons appliqué l'algorithme de Viterbi : il permet d'identifier, à partir du message reçu, la séquence de code ayant la plus forte probabilité de correspondre au message transmis. Il s'agit donc de cumuler dans une métrique les probabilités de tous les états possible à partir du symbole d'entrée.

La première étape est de créer des vecteurs de structure de données modélisant des chemins potentiels (Classe *Path*). Pour chaque mot de N bits du message transmis, on va ensuite créer deux nouveaux chemins (les plus probables d'apparition) à partir des chemins mémorisés précédemment : ces derniers sont identifiés par un calcul de la distance de Hamming (Fonction *hammingDistance*) entre le codage convolutionnel de leurs état (Fonction *code*) et le mot de N bits transmis. Ces chemins seront ajoutés à une liste globale de chemins, qui représentera un treillis de codage.

A partir de ce graphe, il est ensuite possible de sélectionner la séquence de bits la moins éloignée du message d'origine en isolant le chemin ayant la plus petite distance de Hamming cumulée entre toutes ces liaisons (Fonction *distanceComparison*).

1.2 Simulation d'un canal de transmission (Fonction *GSM_transmission*)

Afin de vérifier si notre codage convolutionnel permet effectivement la correction d'erreur, il est nécessaire de simuler les bruits d'un canal de transmission. Nous avons donc ajouté aléatoirement des erreurs de transmission sur le message pour que le résultat à la réception soit différent du message originellement envoyé.

Pour cela, nous avons sélectionné un mot de N bits aléatoirement dans le message transmis, puis un indice aléatoire compris entre 0 et N - 1 pour isoler un seul bit de la séquence. Nous avons ensuite inversé ce bit puis répété l'opération X fois, X étant le nombre théorique d'erreurs maximum pouvant être corrigées par le codage convolutionnel.

En testant, il est clair que l'algorithme de Viterbi implémenté corrige bien les erreurs car le message décodé à la réception est bien identique au message envoyé. Si on ajoute un nombre d'erreurs trop élevé par contre, des erreurs commencent à apparaître dans le message décodé.