TP « Communications Numériques ». Codage cyclique. v1.0-py

N. Lebedev

1 Objectifs de ce TP

- **Reprendre le programme du codage-décodage source Huffman du TPn1.**
- **Appliquer un codage canal cyclique aux bits résultant de la compression de la source par Huffman, pour obtenir les mots de code.**
- **Décoder le code cyclique, vérifiez la reconstruction parfaite de la phrase initiale.**
- **Simuler la transmission des mots de code à travers un canal à erreurs, rajoutées par mot-code canal.**
- **Détecter le signal transmis.**
- **Mettre en évidence les erreurs et comparer la probabilité d'erreur sur CBS sans et avec le codage canal.**

2 Consignes et outils

1. Lecture du chapitre 1 du livre ITILA de D. MacKay.
2. Python, PyCharm/venv, libraries numpy, duhaffman, scikit-dsp-comm, scikit-commpy : `>>> pip install dahuffman scikit-dsp-comm scikit-commpy`
3. (En option : Matlab® / Comm Toolbox), sources complémentaires théorie et exemples : `https://fr.mathworks.com/help/comm/ug/error-detection-and-correction.html`

3 Etapes de réalisation

1. Prenez en main la boite à outils `scikit-dsp-comm` : `https://scikit-dsp-comm.readthedocs.io/en/latest/index.html` et en particulier, le module "Block Codes". La documentation et les exemples sont également sur github. Relevez les modules et les fonctions qui vous seront utiles pour le type de code choisi en TP. Pour cela :

Manipulation	Faites tourner le Notebook IPython fourni en exemple. Faites fonctionner et analysez les fragments de code qui s'y trouvent. Analysez les résultats.

Question	Quels sont les codes qui y sont implémentés ? Quelle est leur aptitude de correction d'erreurs ?

2. Dans la boite à outils
`https://commpy.readthedocs.io/en/latest/channelcoding.html`
analysez et testez la fonction
— `genpoly = cyclic_code_genpoly(n, k)`
Adaptez le format du polynôme-générateur `genpoly` retourné par cette fonction au format et l'ordre du degré du polynôme requis par la fonction
— `block.fec_cyclic(genpoly)`

3. L'enseignant vous attribuera le polynôme-générateur par binôme.
Générez l'objet du code. Fonction utile :
— `block.fec_cyclic(genpoly)`

4. Nous travaillons sur le codage canal en blocs, la longueur des messages à encoder à l'entrée du codeur est fixe, multiple de (k) bits, et des mots de code résultant est multiple de (n) bits. Or, il faut que le format et la taille des séquences de messages soient conformes au format géré par les fonction du codage et du décodage.
Il faut donc éventuellement ajuster la taille de la séquence en sortie de la source, pour satisfaire cette condition (longueur multiple de (k) bits. Cela se fait en insérant, par exemple, les bits de bourrage à "0") avant de les présenter à l'entrée du codeur canal... qu'il ne faut surtout pas oublier d'enlever après le décodage cyclique et avant le décodage Huffman.

5. Encodez les données binaire de la source, fonction
— `cyclic_encoder()`

6. Décodez les données avec la fonction
— `cyclic_decoder()`

7. Présentez ces données en entrée du décodage source Huffman et vérifiez que la `'phrase'` est reconstruite à l'identique.

8. Simulez le canal CBS : rajoutez 1 erreur par mot-code (bloc de (n) bits). Cela consiste à alterner, de manière aléatoire selon la loi uniforme, un des bits par mot-code de taille (n). Utilisez pour cela les fragments du code analysée à la Question 1.
La manière la plus simple est de transformer le message à encoder en matrice de taille appropriée $[N \times n]$, où N est le nombre total de mots-codes, puis faire une boucle sur les lignes (qui sont les mots-codes) et d'y rajouter une erreur.

9. Décodez,
NB : pensez à enlever le bourrage à la fin ! ! !
... et restituez avec le décodage Huffman la phrase à partir du message binaire bruité codé-décodé par le code cyclique.
Le résultat doit être égal à la phrase initiale.

10. Calculez le taux d'erreur sans et avec le codage canal.

4 Avancé (en option)

Ecrivez votre propre méthode de codage et décodage cyclique.
— Pour le codage, on peut utiliser la multiplication du message par la matrice génératrice, pour obtenir les mots-codes $c = uG$. G peut être obtenu à partir du polynôme-générateur par une technique vue en cours.
— Pour le décodage, il faut rechercher et implémenter une des méthodes : soit basée sur la matrice de parité H et le calcul du syndrome, soit sur une méthode algébrique polynômiale.