|  |
| --- |
| ENSEEIHT |
| PROJET TELECOM |
| 1SN |

|  |
| --- |
| Jean-Baptiste PAUX  Adam MEKHICHE  Baptiste VERSTRAETEN  2017/2018 |



SOMMAIRE

[I. Introduction 1](#_Toc515695294)

[II. Etude de la compression/décompression 2](#_Toc515695295)

[III. Etude du codage canal 4](#_Toc515695296)

[A. Codage RS 4](#_Toc515695297)

[B. Entrelaceur convolutif de Forney 4](#_Toc515695298)

[C. Codage convolutif 4](#_Toc515695299)

[IV. Etude de la chaine globale 5](#_Toc515695300)

[V. Conclusion 6](#_Toc515695301)

[VI. Annexes 0](#_Toc515695302)

# Introduction

Le projet de télécommunication a pour but d’implanter la couche physique d’une transmission en respectant la norme DVB-S. Listons rapidement les différentes conditions que nous devons respecter pour la mettre en place :

* Une compression jpeg
* Un double codage canal entrelacé
* Une modulation QPSK
* Une décompression jpeg

Dans notre modélisation nous nous limiterons à l’envoie d’une image en noir et blanc. Nous étudierons respectivement la compression et la décompression, le codage et enfin la chaîne complète en termes d’efficacité spectrale et d’efficacité en puissance. Nous choisirons de ne pas évaluer la modulation QPSK car nous l’avons déjà traité durant les TP, voir figure ci-dessous.

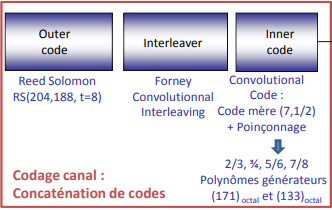


Figure 1 : Chaine de transmission

Nous tracerons les courbes TEB en fonction du SNR pour les différents codages afin d’évaluer leur efficacité et enfin, nous étudierons la chaîne complète en termes d’efficacité spectrale et de puissance.

# Etude de la compression/décompression

On commence par découper l’image en petit bloc de 56 pixels (8x8), chaque pixel étant définie par son niveau de gris (entre 0 et 255). La suite de traitement s’effectue bloc par bloc.

On applique en premier une transformation en cosinus discret pour ses propriétés de regroupement de l’énergie et permet ainsi une plus grande compression.

Par la suite, nous allons quantifier chaque pixel du bloc en fonction de l’importance de chaque pixel pour l’œil humain et moins se préoccuper des pixels moins importants. C’est l’étape responsable de la dégradation de l’image. Cette matrice permet la compression de l’image et en fonction des valeurs de la matrice ont peu plus ou moins compresser fortement. On divise alors chaque bloc par la matrice de compression terme à terme.

Après la quantification, nous passons à la lecture en zigzag, il suffit de suivre le schéma du sujet (1,1 ; 1,2 ; 2,1 ; … ;8,7 ;8,8) mais non avons fait une fonction Matlab qui produit un chemin en zigzag pour les matrices carrées peu importe leur taille. Cela permet de regrouper les pixels se ressemblant pour améliorer la compression de Huffman couplé au Run Length Encoding (RLE).

La RLE permet de compresser la suite de bits de chaque bloc en écrivant qu’un seul 0 à chaque occurrence du 0 et en marquant combien de fois ce 0 se répète à la suite. Il faut donc un deuxième vecteur pour noter cela.

Ainsi, on prend le vecteur de RLE qui a transformé le vecteur du bloc lu en zigzag.

On ajoute tous les vecteurs de chaque bloc les uns à la suite des autres pour avoir un vecteur pour toute l’image.

On le passe dans la fonction Huffman\_code qui renvoie la suite de bits compressée ainsi que le dictionnaire nécessaire à la décompression. Le codage de Huffman permet d’affecter à la valeur la plus récurrente le symbole de poids le plus fable, elle attribue un petit nombre de bits aux valeurs courante et un grand au moins courante. On entremêle les deux vecteurs bit à bit pour tout transmettre en un seul signal.

On passe ensuite à la décompression

On applique toutes les étapes précédentes dans le sens inverse pour décompresser ces données.

On utilise la fonction Huffman\_deco qui prend la suite de bits et le dictionnaire de compression pour décompresser la suite de bits. On récupère chaque vecteur de cette suite de bits puis on utilise la fonction Matlab « repelem » pour faire la fonction inverse à la RLE.

On utilise invzigzag pour remettre la suite de bits sous forme de bloc, puis on dé quantifie le bloc en multipliant terme à terme le bloc et la matrice quantification.

Ensuite, il existe la fonction IDCT qui permet de faire l’opération inverse de la DCT.

Nous sommes revenus à un bloc lisible et il faut maintenant les remettre dans le bon ordre.

On affiche ensuite l’image :

Figure 2: Avant compression Figure 3: Après compression

Lors du transfert dans le canal un bruité certaines erreurs de bits peuvent apparaitre : Si un bit change et que le dictionnaire de décompression ne reconnait plus la suie de bits alors la fonction Huffman\_deco s’arrête et on ne peut décompresser.

Pour empêcher cela il faut qu’il n’y ait pas de bruit dans le canal de transmission.

# Etude du codage canal

## Codage RS

Le codage RS s’effectue à l’aide de la fonction Matlab intitulé codage\_RS.m, elle a pour entête les paramètres :

* t : la capacité de correction du code (en nombre de symbole)
* N\_RS : le nombre de symbole du mot de code
* input\_bit : la chaine de bits à coder/décoder
* mode : le choix du codage ou du décodage

La fonction retourne ainsi la chaine de bits après codage ou décodage.

## Entrelaceur convolutif de Forney

L’entrelacement s’effectue à l’aide de la fonction Matlab intitulé entrelaceur.m, elle a pour entête les paramètres :

* nrows : le nombre de registres
* slope : la capacité de chaque registre
* input\_bit : la chaine de bits à entrelacer/désentrelacer
* mode : le choix de l’entrelacement ou du désentrelacement

La fonction retourne ainsi la chaine de bits après traitement.

## Codage convolutif

# Etude de la chaine globale

## Tracé des courbes théoriques de TEB en fonction du SNR

Traçons à présent les différentes courbes TEB, nous représenterons sur un même graphe la courbe théorique ainsi que celle simulée de la modulation QPSK sans codage. Puis celui avec codage convolutif, celui avec les codages concaténés sans entrelacement et enfin avec entrelacement.

# Conclusion

# Annexes