TP 1 de communications numériques

M. Ellouze et R. Tajan

1 Objectifs et évaluation

Le but de ce TP est de développer, à l'aide du logiciel Matlab, un modulateur et un démodulateur numérique pour des modulations M-PSK. Dans cette séance, vous écrirez un script interactif de Matlab (notebook) ainsi que des fonctions Matlab permettant

1. de mettre en œuvre un modulateur M-PSK pour lequel l'utilisateur pourra choisir
   — M l'ordre de la modulation,
   — le mapping (naturel ou Gray),
2. d'afficher une constellation ainsi que son mapping
3. de mettre en œuvre un démodulateur M-PSK

Vos codes et votre script interactif sont à rendre en fin de séance sur l'interface Thor `https://thor.enseirb-matmeca.fr/ruby/`

2 Développement du modulateur

L'objectif de cette partie est de développer une fonction Matlab qui aura (à la fin) le prototype suivant :

Listing 1 – Fonction **mod_psk.m**.

```
function constellation = mod_psk(M, is_gray)
% M       : (int) ordre de la modulation
% is_gray : (bool) type du mapping
%           true => "Gray", false => "Naturel"
```

Cette fonction construit une "constellation" comme une table de correspondance et prends en entrée trois paramètres :

— `M` : l'ordre de la modulation,
— `is_gray` : un booléen qui indique si l'étiquetage est de Gray ou non.

Le vecteur retourné est nommé `constellation`, il possède une taille M et il contient tous les symboles de la modulation numérique et est construit de telle sorte que sa i^e case contienne le symbole associé à la représentation binaire de $i-1$ ($i-1$ et non i à cause des indices de Matlab).

Astuce : cette représentation de la constellation permets une association bit/symbole symplifiée. En effet, si on souhaite moduler avec une 16-PSK les bits $b = [0, 1, 1, 1]$ il nous suffira de lire la case 8 du tableau constellation dans Matlab. La représentation binaire du message pourra alors être simplifiée en la remplaçant par sa représentation décimale. Dans l'exemple précédent, on aura alors $b = [7]$.

2.1 Mapping Naturel

Compléter la fonction `mod_psk` afin qu'elle renvoie la constellation d'une M-PSK avec un mapping naturel. On rappelle que les symboles M-PSK sont à considérer dans l'ensemble $\mathcal{M} = \left\{ e^{j2\pi \frac{k}{M}} | k \in [0, M-1] \right\}$

2.2 Affichage de la constellation

Écrire la fonction `plot_cstl` qui aura le prototype suivant :

Listing 2 – Fonction **plot_cstl.m**.

```
function h = plot_cstl(constellation)
% constellation : (vector) vecteur de taille M contenant la
    modulation

% ... votre code
% h = plot(...)
% return h
```

Cette fonction permettra d'afficher la constellation ainsi que les étiquettes associées aux symboles sur un même graphique.

Dans votre notebook : faites appel à vos fonctions pour illustrer leur bon fonctionnement pour $M = 4$, $M = 8$ et $M = 16$. Vous expliquerez, à l'aide du graphique pour $M = 16$ l'inconvénient de l'étiquetage naturel.

2.3 Mapping Gray

Compléter la fonction `mod_psk` afin qu'elle renvoie la constellation d'une M-PSK avec un mapping de Gray. Le mapping de Gray devra être construit en utilisant la construction miroir présentée dans la Figure 1.

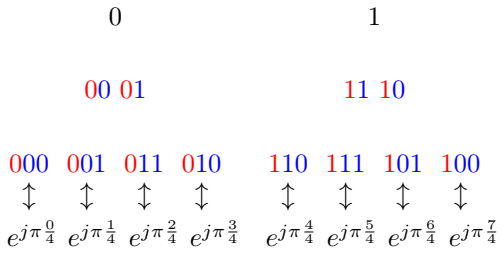

FIGURE 1 – Construction miroir du mapping de Gray pour une 8-PSK

Dans la Figure 1 vous remarquerez que

1. les étiquettes de gauche en bleu sont identiques aux étiquettes de l'itération précédente,
2. les étiquettes de gauche commencent par un 0 (rouge),
3. les étiquettes de droite en bleu sont "symétriques" par rapport aux étiquettes de l'itération précédente,
4. les étiquettes de droite commencent par un 1.

Une fois la liste de M étiquette obtenue, créer la liste de constellation de sorte que la $k^{\text{éme}}$ étiquette de la liste corresponde au symbole $e^{j2\pi\frac{(k-1)}{M}}$ (cf. dernière ligne de la Figure 1).

Astuce : Il n'est pas utile d'implémenter le mapping miroir avec des vecteurs binaires, il est plus simple et plus efficace de l'implémenter en travaillant directement sur les entiers correspondants.

Dans votre notebook : faites appel à vos fonctions pour illustrer le bon fonctionnement de votre mapping de Gray pour $M = 4$, $M = 8$ et $M = 16$. Vous expliquerez, à l'aire du graphique pour $M = 16$ l'avantage de ce mapping.

3 Développement du démodulateur

L'objectif de cette partie est de développer une fonction Matlab qui aura le prototype suivant :

Listing 3 – Fonction **demod_psk.m**.

```
function uh = demod_psk(y, constellation)
% y             : (vecteur) vecteur du signal reçu
% constellation : (vecteur) vecteur de la constellation
```

Cette fonction démodule le signal `y` pour une `constellation` donnée en argument. Cette fonction s'appuiera sur la méthode des plus proches voisins (le symbole décidé sera le symbole de la constellation le plus proche de y_i). Cette fonction renvoie le vecteur binaire (ou décimal) correspondant aux symboles décidés.

Dans votre notebook : faites appel à vos fonctions pour illustrer le bon fonctionnement de votre démodulateur. Vous pourrez, par exemple, calculer le taux d'erreur binaire pour un vecteur `y` bien choisi.

3.1 Bonus : affichage des régions de décisions

Pour valider les régions de décisions, afficher la constellation de 1000 symboles bruités choisis au hasard dans la constellation. Chaque région de décision doit avoir sa propre couleur.

4 Bonus : Développement d'un modulateur QAM

Sur le même principe que celui utilisé pour le modem PSK, implémentez et présentez un modem QAM. Le mapping naturel sera "naturel" sur la voie I et sur la voie Q.

5 Contacts

— Malek Ellouze - malek.ellouze@ims-bordeaux.fr
— Romain Tajan - romain.tajan@ims-bordeaux.fr