TP3 de communications numériques

D Dallet et R. Tajan

1 Objectifs et évaluation

Le but de ce TP est de s'intéresser aux performances des communications numériques en bande de base et sur fréquence porteuse émettant des symboles M-PSK dans le cas de présence de bruit.

Dans cette séance, vous écrirez un script interactif de Matlab (notebook) permettant

1. de mettre en œuvre la chaîne de communications numériques sans présence de bruit
2. d'étudier l'impact de la présence de bruit sur le taux d'erreur binaire

Vos codes et votre script interactif sont à rendre en fin de séance sur l'interface Thor `https://thor.enseirb-matmeca.fr/ruby/`.
Pour ce troisième TP, les fonctions du premier et du deuxième TP vous sont fournies en .p . Vous pouvez faire appel à ces fonctions si jamais vous n'avez pas réussi à développer toutes les fonctions du premier TP.

Dans toute la suite, on considérera les paramètres suivants :
— $T_s = 1\mu s$ le temps symbole
— $F_e = \frac{1}{T_e} = 10MHz$ la fréquence d'échantillonnage
— $F_{se} = \frac{T_s}{T_e}$ le facteur de sur-échantillonnage
— $g(t)$, le filtre de mise en forme, est une porte de largeur T_s
— $N_s = 5000$ le nombre de symboles par paquet envoyé

2 Simulation d'une M-PSK sans bruit

L'objectif de cette partie est de tester en premier lieu votre chaîne de communications numériques en l'absence de bruit.

Reprendre votre fonction `recepteur_optimal` du TP2. On rappelle qu'un récepteur est dit optimal s'il vérifie les conditions suivantes :

1. le filtre de réception est adapté au filtre d'émission (et filtre du canal s'il y en a un)
2. la cascade filtre de mise en forme - filtre adapté vérifie le critère de Nyquist.

La fonction `recepteur_optimal` aura le prototype suivant :

Listing 1 – Fonction **recepteur_optimal**.

```
1 function r_n = recepteur_optimal(y_l, Fse, g, Ns)
2 % Arguments :
3 % y_l : (vecteur complexe) signal reçu (sur-échantillonné)
4 % Fse : (int) facteur de sur-échantillonnage
```

```
% g : (vecteur réel) Réponse impulsionnelle du filtre de mise en forme
% Ns : (int) Nombre d'échantillons à renvoyer
% Retour :
% r_n : (vecteur complexe) signal après filtrage adapté et échantillonnage à T_s
```

Cette fonction réalisera les actions suivantes sur le signal en sortie de canal y_l (ici $y_l = s_l$) :

— filtrer le signal y_l par le filtre adapté (ce qui donne r_l),

— sous-échantillonner le signal r_l d'un facteur F pour récupérer N_s échantillons (chacun servant à décider un symbole transmis),

Dans votre notebook : Tester votre chaîne de communications numériques en absence de bruit. pour cela, gérnérer N_s symboles 8-PSK et illustrer les propriétés suivantes :

1. les bits reçus sont bien identiques aux bits transmis,
2. il n'y a pas d'interférences entre symboles : sans bruit on doit avoir

$$r_n = R_g(0)s_n \tag{1}$$

où s_n est le symbole envoyé et $R_g(\tau) = \int_{\mathbb{R}} g(t)g^*(t-\tau)dt$.

3 Simulation d'une M-PSK en présence de bruit

On s'intéresse à présent à l'étude des performances d'une chaîne de communications numériques en présence de bruit. Nous allons réaliser une simulation pour différentes variances de bruit.

Nous allons donc créer une fonction qui calcule le TEB ainsi que la probabilité d'erreur P_b en fonction du rapport $\frac{E_b}{N_0}$ en dB. Cette fonction sera la suivante.

Listing 2 – Fonction **compute_TEB**.

```
function [TEB, Pb] = compute_TEB(eb_n0_dB, constellation, g, Fse, Ns, max_bit_err)
% Arguments :
% eb_n0_dB : (vecteur réel) ensemble des valeurs de Eb/N0 à tester en dB
% constellation : (vecteur complexe) constellation des symboles
% g : (vecteur réel) Réponse impulsionnelle du filtre de mise en forme
% Fse : (int) facteur de sur-échantillonnage
% Ns : (int) Nombre de symboles à envoyer dans un paquet
% max_bit_err : (int) Nombre d'erreurs binaires à attendre avant de passer au point de Eb/N0 suivant

% Retour :
% TEB : (vecteur) Une valeur de Taux d'Erreur Binaire (TEB) par point de Eb/N0
% Pb : (vecteur) Une valeur de probabilité d'erreur par point de Eb/N0

```

```
%% Initialisation des paramètres
M = length(constellation)
n_b = log2(M); % Nombre de bits par symboles
Eg    = % Energie du filtre de mise en forme
sigA2 = % Variance théorique des symboles
eb_n0    = 10.^(eb_n0_dB/10); % Liste des Eb/N0
sigma2   = sigA2 * Eg ./ (n_b * eb_n0); % Variance du bruit
    complexe en bande de base

% Probabilité d'erreur pour un mapping de Gray (théorie)
if M == 2
    Pb = qfunc(sqrt(2*eb_n0));
else
    Pb = (2/n_b)*qfunc(sqrt(2*n_b*(sin(pi/M).^2)*eb_n0));
end

TEB = zeros(size(eb_n0)); % Tableau des TEB (résultats)
for i = 1:length(eb_n0)
    error_cnt = 0;
    bit_cnt = 0;
    while error_cnt < 100
        %% Émetteur
        % Mettre ici votre émetteur

        %% Canal
        nl = sqrt(sigma2(i)/2) * (randn(size(sl)) + 1j*
           randn(size(sl))); % Génération du bruit blanc
           gaussien complexe
        % Bruiter votre signal transmis

        %% Récepteur
        % Mettre ici votre récepteur

        error_cnt = error_cnt + ... % incrémenter le
           compteur d'erreurs
        bit_cnt   = bit_cnt   + ... % incrémenter le
           compteur de bits envoyés
    end
    TEB(i) = error_cnt/bit_cnt;
end
```

Dans votre notebook tracer et interpréter les résultats suivants :

1. Pour une modulation QPSK, tracer sur un même graphique les courbes suivantes :
   — P_b en fonction de $\frac{E_b}{N_0}$ (mapping de Gray),
   — TEB en fonction de $\frac{E_b}{N_0}$, en considérant le mapping naturel,
   — TEB en fonction de $\frac{E_b}{N_0}$, en considérant le mapping de Gray.
   Pour cela, en plus de ceux décrits précédemment, vous utiliserez le jeu de paramètres suivants :
   — `max_bit_err = 100`

— `eb_n0_dB = 0:0.5:10`

2. Toujours avec `max_bit_err = 100` tracer sur un même graphique les courbes suivantes
   — P_b en fonction de $\frac{E_b}{N_0}$ (QPSK),
   — TEB en fonction de $\frac{E_b}{N_0}$, (QPSK Gray),
   — P_b en fonction de $\frac{E_b}{N_0}$ (8-PSK),
   — TEB en fonction de $\frac{E_b}{N_0}$, (8-PSK Gray),
   — P_b en fonction de $\frac{E_b}{N_0}$ (16-PSK),
   — TEB en fonction de $\frac{E_b}{N_0}$, (16-PSK Gray).

À l'aide du graphique 2, donner les plages d'utilisations des différentes modulations permettant de maximiser le débit binaire si l'application nécessite un TEB inférieur à 10^{-4}.

4 Simulation sur fréquence porteuse

En pratique, un signal complexe est d'abord mis sur fréquence porteuse avant d'être transmis. Cette opération est réalisée comme suit

$$s(t) = \sqrt{2} Re\left(s_l(t) e^{j2\pi F_p t}\right) \tag{2}$$

où $s_l(t)$ est le signal en bande de base et F_p la fréquence porteuse. Écrire un script `compute_TEB_FP` permettant, dans votre notebook, de réaliser et de commenter les simulations suivantes :

1. DSP de $s(t)$ illustrant la mise sur fréquence porteuse,
2. Comparaison de P_b en fonction de $\frac{E_b}{N_0}$ (8-PSK), et du TEB en fonction de $\frac{E_b}{N_0}$, (8-PSK Gray, $F_p = 2.5Hz$)

5 Contacts

— Dominique Dallet - dominique.dallet@ims-bordeaux.fr
— Romain Tajan - romain.tajan@ims-bordeaux.fr