Projet Télécommunication

Simulation d'une chaîne de transmission numérique avec Matlab

Mathieu David & William De Decker

Matricules :	14102 & 14130
Durée du projet :	Mars 12, 2018 – Mai 18, 2018
Enseignant :	T. Sartenaer

Table des matières

1 Introduction

1.1. Énoncé du projet

Dans le cadre du cours de télécommunications de 1ère Master, il nous est demandé de simuler un système de transmission numérique multi-utilisateurs destiné à fonctionner sur un réseau câblé. La simulation est réalisée à l'aide du logiciel MATLAB. Dans cet exercice, nous ne considèrerons que la couche physique d'un point de vue strictement fonctionnel : Un flux binaire se présente à l'un des émetteurs et doit être tansmis par des moyens adéquats à l'un des récepteurs situé sur le réseau. On ne prendra donc pas en compte, ni la signification des bits qu'on envoie, ni le partage d'une même ressource physique par plusieurs utilisateurs, ni l'implémentation des différents modules aux moyens de composants électroniques.

L'intérêt de la simulation est grande. Elle permet de faire varier différents paramètres du système et en observer les conséquences. Si le modèle reflète suffisament bien le système physique, il est possible d'en analyser les performances sous différentes configurations sans devoir implémenter le système physique qui pourrait se révéler être une entreprise couteuse et gourmande en temps. La simulation est une alternative pratique permettant d'approximer les résultats réels tout en évitant de devoir mettre en place le système physique.

1.2. Topologie du réseau

On considère un canal multi-utilisateurs, qui n'est autre qu'un câble partagé, auquel K modules sont raccordés en différents points du canal. Les K modules possèdent N ressources physiques, caractérisées par une bande de fréquences déterminée. Un premier niveau d'accès multiple est donc géré au niveau de la couche physique par la méthode de répartition en fréquence. Cependant, comme on suppose que $K > N$ un autre niveau d'accès multiple devra se faire au niveau du protocol. Ceci ne concerne pas la couche physique et ne sera donc pas abordé. Pour nos simulations, nous considérerons toujours que $K \leq N$.

A chaque moment, un module pourra simultanément transmettre un signal sur un des N canaux fréquentiels (ou ne rien transmettre du tout) et recevoir un signal sur tous les N canaux fréquentiels. Une représentation de la topologie du réseau est affichée à la figure 1.1.

1.3. Chaine de transmission

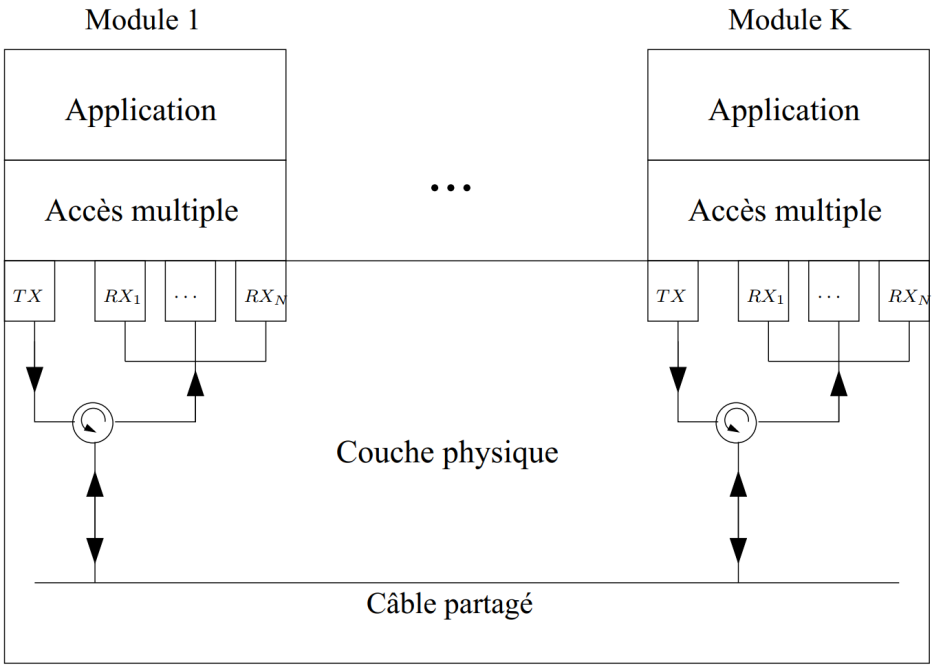


FIGURE 1.1 – Topologie du réseau partagé

2

Émetteur

2.1. Génération des messages

Les messages à transmettre sont générés en choisissant aléatoirement des 1 et des 0 jusqu'à créer des messages de M_d bits.

$$x_n(k) = 0/1 \qquad \text{avec} \quad k \in [0, M-1]$$

```
m = randi([0, 1], [K, Md]);
```

Le code ci-dessus crée une matrice de dimension $K \times M_d$ contenant K messages de M_d bits contenant aléatoirement des 1 ou des 0.

2.2. Séquence de synchronisation

Afin que le récepteur puisse se synchroniser à ce que l'émetteur envoie, on ajoute une séquence de synchronisation connue. Le récepteur sera à l'affut de cette séquence et l'utilisera pour synchroniser son moment d'échantillionage avec celle de l'émetteur. On ajoute une séquence de M_s bits, par exemple tous des 1, avant le message.

```
leading = ones(K, Ms);
m = [leading m];
```

2.3. Codage du message

Pour encoder le message, nous allons utiliser une PAM binaire. PAM (Pulse Amplitude Modulation) est une technique qui permet de coder une série de bits par une amplitude du signal. Pour une PAM-4 par exemple on peut utiliser 4 niveaux d'amplitudes différentes pour coder les données. Il est donc possible de coder $log_2(4) = 2$ bits par symbole. Comme nous utilions une PAM binaire, soit PAM-2, nous pouvons utiliser que 2 niveaux d'amplitudes et donc chaque symbole ne peut représenter qu'un bit.

$$a_n(k) = \begin{cases} 1 & \text{si } k = 1 \\ -1 & \text{si } k = 0 \end{cases}$$

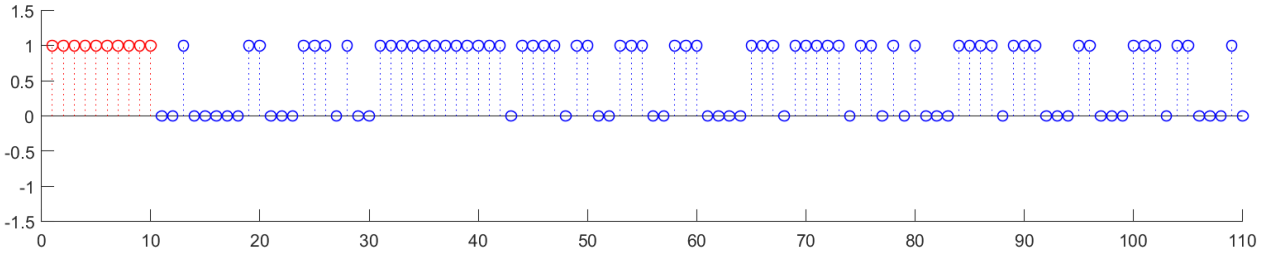


FIGURE 2.1 – Message avec séquence de synchronisation

Si on prend un message aléatoire d'une longueur $M_d = 100$ bits, avec une séquence de synchronisation de longueur $M_s = 10$ bits. Nous obtenons une séquence comme illustrée à la figure 2.1. Les bits rouges représentes la séquence de synchronisation, les bits bleus représentes les données à transmettres.

Après le codage du message en PAM binaire, nous obtenons la séquence de la figure 2.2, dont l'amplitude vaut soit 1, soit -1 en fonction de la valeur des bits du message original.

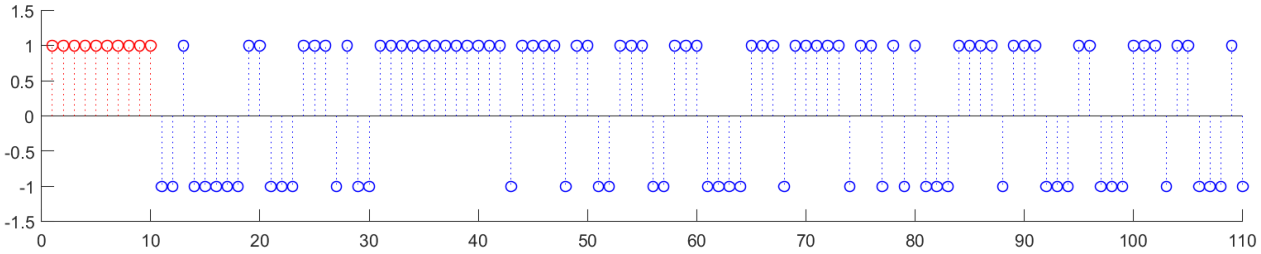


FIGURE 2.2 – Message codé en PAM binaire

2.4. Filtre de mise en forme

Un signal numérique quelconque à une grande bande passante dû au fait que le signal de référence soit carré. Un signal carré étant en effet constitué d'une somme infinie de sinus. Si la bande passante du canal était infinie, nous pourrions transmettre le signal numérique tel quel. Mais en pratique, la largeur de bande du canal est limitée, soit par le partage de celle-ci entre multiple utilisateurs, soit par des propriétés du canal qui limitent sa bande passante. Dans le cas du câble de transmission partagé, la bande passante est limitée par les effets de distorsions tel que l'effet de peau. De plus, la bande passante utilisable doit être partagée entre les K utilisateurs. Le but du filtre de mise en forme est de positionner le signal dans une certaine bande de fréquence désirée. Ce positionnement permettra la démodulation de chaque canal qui serait impossible si les messages se situaient sur la même bande de fréquence, les messages se parasiteraient dans ce cas l'un l'autre.

Pour positionner le signal dans une bande de fréquence correspondante, nous allons moduler une forme d'onde $p_n(t)$, de façon à obtenir le signal $s_n(t)$ à transmettre.

$$s_n(t) = A \sum_{k=-\infty}^{\infty} a_n(k)\, p_n(t - kT_b)$$

La forme d'onde est choisie comme

$$p_n(t) = g(t)\cos(\omega_n\, t)$$

avec $\omega_n = 2\pi\frac{2n}{T_b}$. Le filtre $g(t)$ est un filtre en *racine de Nyquist.*

2.4.1. Filtre de Nyquist

La distortion intersymbole apparait lorsque, dans une chaîne de transmission, un symboles influence la valeur du suivant.

Un filtre de Nyquist est un filtre qui a la propriété d'éliminer l'interférence entre symboles aux moments précis d'échantillonnages. C'est à dire, tout filtre qui satisfait l'équation suivante.

$$\sum_{n=-\infty}^{\infty} P(f - nR_b) = T_b$$

Pour satisfaire cette équation, la solution la plus simple est de définir le filtre idéal avec une transmittance rectangulaire.

$$P(f) = \frac{1}{2W} rect(\frac{f}{2W})$$

où $W = \frac{1}{2T_b}$

Si nous ramenons ce filtre sous sa forme temporelle, nous obtenons un sinus cardinal.

$$p(t) = sinc(2Wt)$$

Cette fonction temporelle a son maximum en $t = 0$ et passe par 0 pour tous les multiples entiers de t_b, qui est la durée d'un bit. Si nous échantillons le signal en $k \cdot T_b$, les bits ne s'influencent donc pas et la distortion intersymbole est nulle.

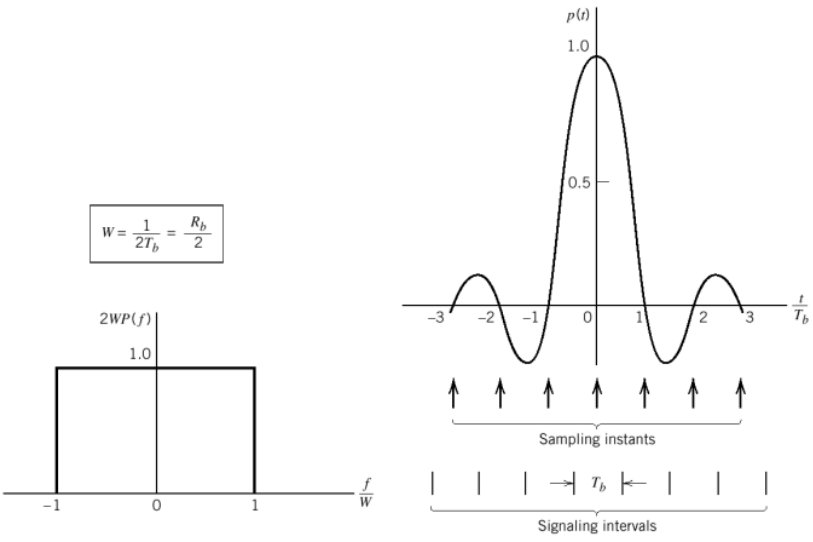


FIGURE 2.3 – Spectre fréquentiel du filter de Nyquist et fonction en temporel

Cependant, le filtre de Nyquist idéal est problématique en réalité.

1. Premièrement, sa transmittance vaut 1 entre $-W$ et W et 0 partout ailleurs. Physiquement, cette transition brutale est impossible à obtenir.
2. Deuxièmement, l'impulsion $p(t)$ diminue en $\frac{1}{2Wt}$ et donc si le timing n'est pas parfait, on peut introduire des dégradations importantes de performances.

2.4.2. Filtre en cosinus surélevé

Nous pouvons résoudre les problèmes évoqués ci-dessus en élargissant légèrement la bande passante du filtre. En effet on peut développer les conditions de P(f) vers

$$P(f) + P(f-2W) + P(f+2W) = \frac{1}{2W} \qquad \text{avec } -W < f < W$$

Cette condition peut alors être respectée par la fonction cosinus surélevé dont le spectre comporte une section plate entre des sections de transitions à allure sinusoidale (rolloff sections).

$$P(f) = \begin{cases} \dfrac{1}{2W}, & 0 \le |f| < f_1 \\ \dfrac{1}{4W}\left\{1 - \sin\left[\dfrac{\pi(|f| - W)}{2W - 2f_1}\right]\right\}, & f_1 \le |f| < 2W - f_1 \\ 0, & |f| \ge 2W - f_1 \end{cases}$$

FIGURE 2.4 – Fonction de transfert $P(f)$ du filtre en cosinus surélevé

Ces zones de "rolloff" sont délimitées par le coefficent de rolloff (α) qui représente le surplus de bande passante par rapport à la solution idéale du filtre de Nyquist. Nous pouvons observer son influence à la figure suivante.

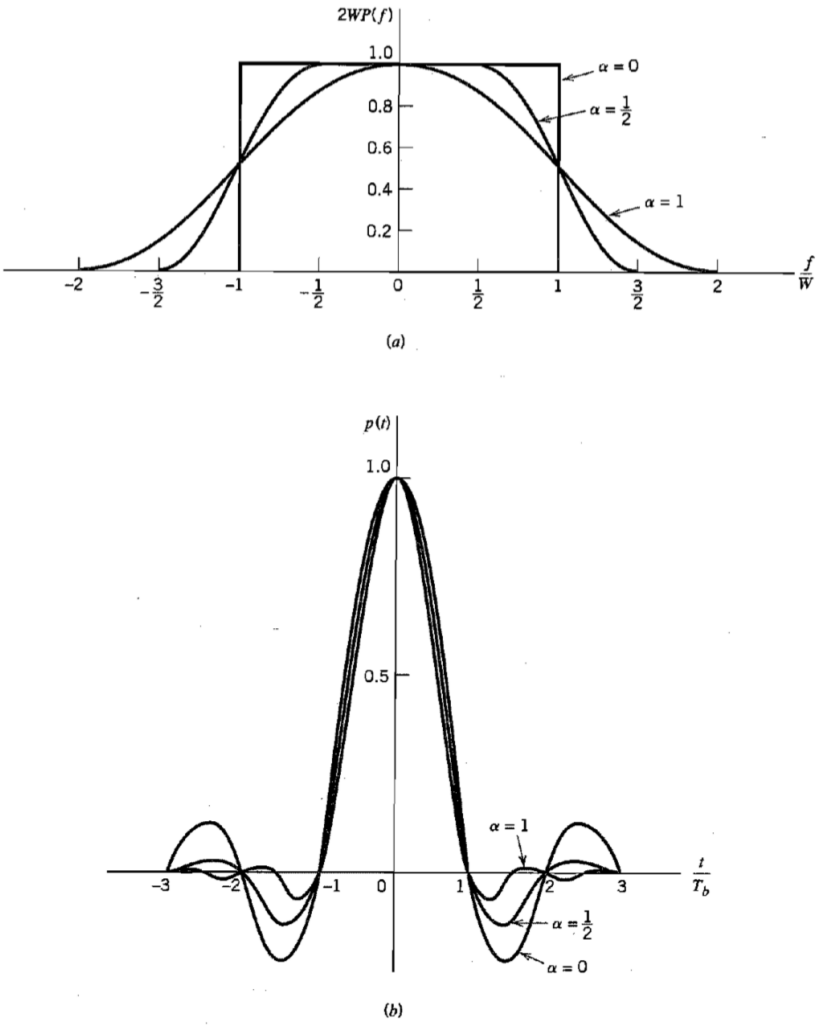


FIGURE 2.5 – Filtre en cosinus surélevé

La réponse temporelle d'un tel filtre est donnée par l'équation

$$p(t) = sinc(2Wt) \cdot \left(\frac{cos(2\pi\alpha Wt)}{1 - 16\alpha^2 W^2 t^2} \right)$$

Notons que la première partie de cette équation est similaire à celle de l'équation du filtre de Nyquist et garanti donc des passages par zéro aux instants d'échantillonnage désirés, mais que la seconde partie rend la fonction décroissante selon $1/T^2$ pour de grandes valeurs de t. C'est ce second terme qui permet l'atténuation des oscillations secondaires dans le graphe temporel et permet une plus grandes tolérances sur les instants d'échantillonnage. Remarquons d'ailleurs que plus α tend vers 1, plus les oscillations secondaires tendent à disparaitre mais plus on consomme de bande passante. La distortions intersymboles diminue également lorsque α tend vers 1. De façon générale, nous pouvons dire que plus α s'approche de l'unité, plus le filtre en cosinus surélevé se comporte comme un filtre de Nyquist idéal.

2.4.3. Filtre en racine de Nyquist

Le principe du filtre en racine de Nyquist est d'avoir deux filtres complémentaires, un dans l'émetteur et un dans le récepteur (filtres adaptés) de façon à ce que la convolution du signal par les deux donne l'équivalent d'un filtre de Nyquist. C'est à dire, un filtre $g(t)$ de façon à ce que $u(t)$ soit un filtre de Nyquist pour :

$$g(t) * g(-t) = u(t)$$

Dans MATLAB, la construction d'un filtre en racine de Nyquist de type cosinus surélevé peut se faire de la façon suivante :

```
rolloff = 0.4;      % Arbitrary value between 0 and 1
L = 10;             % Span (in symbols) of the Nyquist filter
oversampling = 64; % Number of samples per symbol

% Create a raised cosine (finite impulse response) filter
rcosfilter = rcosdesign(rolloff, L, oversampling, 'sqrt');
```

2.4.4. Suréchantillonnage

Le filtre de Nyquist est difficile à implémenter de façon analogique, un solution consiste alors à générer une version échantilonnée du signal $s_n(t)$ à l'aide d'un filtre numérique à réponse impulsionelle finie (FIR) et dont les coefficients sont programmables. Ensuite nous allons interpoler le signal ainsi constuit à l'aide d'un convertisseur numérique-analogique (DAC).

On construit un signal β fois plus rapide que la cadance des symboles.

Le facteur d'échantillionage β doit être choisi de façon à respecter le critère de Shannon. Pour rappel, le critère de Shannon stipule que la fréquence d'échantillonnage doit être au moins deux fois supérieure à la fréquence maximale présente dans le signal.

Si on considère le canal fréquentiel le plus haut et un roll-off $\alpha = 1$, on trouve que

$$\beta \geq 4N - 2$$

2.4.5. Implémentation dans MATLAB

On commence par définir le filtre en cosinus surélevé, comme montré précédemment. Ensuite MATLAB nous offre une fonction qui permet de convoluer le message avec le filtre en faisant le suréchantillonage.

Ensuite nous construisons les porteuses et modulons leurs amplitudes par les symboles filtrés. Les fréquences des porteuses sont calculées à l'aide de l'équation suivante

$$\omega_n = 2\pi \cdot \frac{2n}{T_b} \qquad \Rightarrow \omega_1 = 4000\pi\,, \quad \omega_2 = 8000\pi\,, \quad \omega_3 = 12000\pi\,, \ldots$$

$$f_n = \frac{\omega_n}{2\pi} = \frac{2n}{T_b} \qquad \Rightarrow f_1 = 2\ kHz\,, \quad f_2 = 4\ kHz\,, \quad f_3 = 6\ kHz\,, \ldots$$

```
% Create a raised cosine (finite impulse response) filter
rcosfilter = rcosdesign(rolloff, L, oversampling, 'sqrt');

% Convolution of the messages with the filter
m_conv = upfirdn(m', rcosfilter, oversampling)';
t_window = linspace(0, 2*pi * length(m_conv) * Tn, length(m_conv));


% Carriers
n = 0:(N-1);

% Frequencies
wn = 2*pi * 2*n / Tb;
fn = wn / (2*pi);

% Modulate carriers
carriers = cos(wn(1:K)'*t_window);
modulated_carriers = carriers .* m_conv;
```

2.5. Normalisation

Les signaux que nous avons générés jusqu'à présent possèdent une puissance arbitraire. Pour calculer la puissance moyenne du signal discret, passant par un câble d'impédance Z_c nous pouvons utiliser la formule suivante :

$$P_{\text{moy}} = \frac{1}{N} \sum_{k=0}^{N} \left| \frac{x^2[k]}{Z_c} \right|$$

Nous voudrions pouvoir définir le niveau de puissance auquel nos signaux sont envoyés depuis l'émetteur. Pour cela, nous allons d'abord normaliser le signal pour ensuite pouvoir l'amener à la puissance désirée. Pour ce faire, nous allons **diviser** chaque échantillon par la racine carrée de la puissance moyenne du signal.

$$s_{\text{normalisé}}(k) = \frac{s(k)}{\sqrt{P_{\text{moy}}}}$$

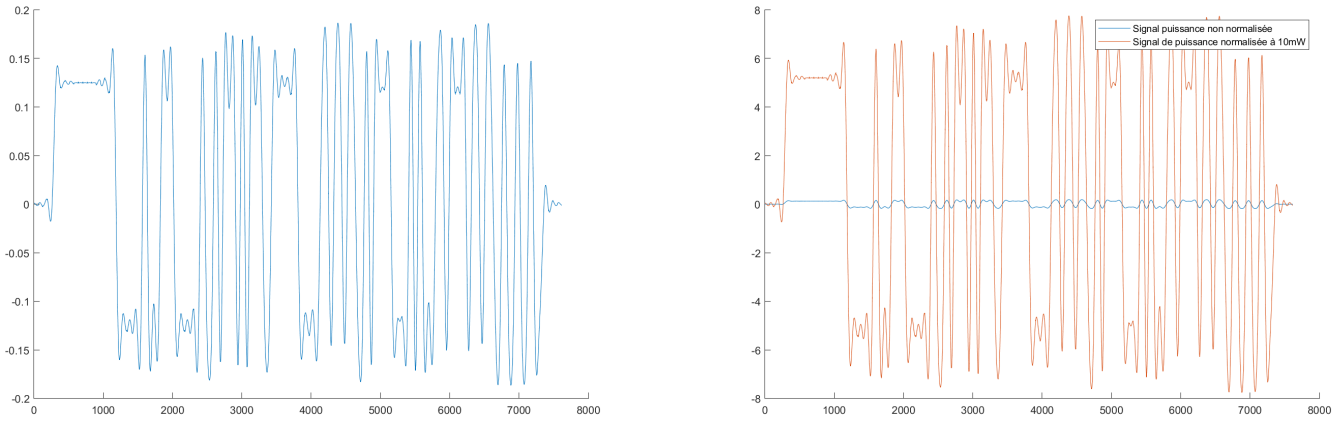
Ensuite nous amener le signal à la puissance désirée, nous pouvons simplement **multiplier** les échantillons par la racine carrée de la puissance désirée.

$$s_{P_t}(k) = s_{normalisé}(k) \cdot \sqrt{P_t}$$

Dans le code MATLAB, cela donne :

```
P_avg = sum(((signal / Zc) .^ 2), 2) / size(signal, 2);
signal_normalized = signal ./ sqrt(P_avg);

m_normalized = signal_normalized .* sqrt(Pt);
```



(a) Signal avant normalisation

(b) Signal après normalisation

FIGURE 2.6 – Normalisation du signal

3

Canal

Le canal physique, qui est dans notre cas un câble de cuivre, transporte le message depuis l'émméteur jusqu'au récepteur. Il peut introduire des distorsions linéaires du signal du à son effet dispersif. Ces distorsions sont causées entre autres par l'effet de peau qui fait varier la résistance du canal en fonction de la fréquence du signal. On peut également citer les réflexions multiples des signaux sur les terminaisons du câble.

Hypothèse : On fera l'hypothèse que ces distorsions peuvent être considérées négligeables tant qu'on travaille dans une bande de fréquence étroite de l'ordre du kHz.

3.1. Modèle du canal

Pour notre simulation, le canal sera simplement modélisé par

- Un délai τ_n qui est beaucoup plus petit que T_b
- Une atténuation $\alpha_n < 1$
- Un bruit additif blanc gaussien $n(t)$ de densité spectrale bilatérale $\frac{N_0}{2}$ qui sera ajouté à tout signal traversant le canal

Si en entrée du canal on a le signal $s_n(t)$, en sortie du canal nous obtiendrions le signal $r(t)$ définit comme :

$$r(t) = \sum_{n=0}^{N-1} \alpha_n \, s_n(t - \tau_n) + n(t)$$

3.2. Implémentation

Maintenant que nous avons une idée de ce que nous devons obtenir, voyons comment nous devons implémenter le cannal dans le code MATLAB.

3.2.1. Délais

Chaque message va se voir retardé d'un délais τ_n différent, qui se situe entre 0 et T_b secondes. Pour cela, nous créons un vecteur de longueur K contenant des nombres aléatoires entre 0 et T_b. Ensuite nous passons sur chacune de ces valeurs dans une boucle for afin d'ajouter x zéros devant le signal et un certain nombre de zéros à la fin de la rangée afin que les dimensions de la matrice soient respectées.

```
delays = randi([0 Tb/Tn],1,K);

% Pre-allocate the matrix for performance reasons
m_normalized_delayed = zeros(K, size(m_normalized, 2) + Tb/Tn);

for i = 1:K
    m_normalized_delayed(i,:) = [zeros(1, delays(i))...
                                 m_normalized(i,:)...
                                 zeros(1, Tb/Tn - delays(i))];
end
```

On se retrouve alors avec une matrice *m_normalized_delayed* pour laquelle chaque canal est retardé de façon indépendante.

3.2.2. Atténuation

L'atténuation par un gain $\alpha_n < 1$ se fait tout simplement par la multiplication de la matrice par un scalaire représentant le gain.

```
m_normalized_delayed = m_normalized_delayed * attenuation;
```

3.2.3. Bruit blanc gaussien

A chaque canal, nous allons ajouter un bruit additif blanc gaussien. Nous créons donc une matrice de taille $K \times (M \cdot \beta)$ que nous remplissons avec des nombres aléatoires selon une distribution normale, de moyenne μ nulle et variance σ^2 unitaire.

```
noise = randn(K, size(m_normalized_delayed, 2));
```

Ensuite nous devons adapté la puissance du bruit de façon à obtenir le rapport $\frac{E_b}{N_0}$ souhaité. Ce rapport est le **rapport entre l'énergie par bit et la densité spectrale de puissance du bruit**.

E_b est l'énergie par bit, soit la puissance du signal divisé par le débit binaire :

$$E_b = \frac{P_t}{R}$$

N_0 est la densité spectrale de puissance du bruit, soit la puissance du bruit pour une bande de 1 Hz. Pour obtenir la puissance du bruit, nous devons multiplier par la bande passante du signal.

$$N = N_0 \cdot \mathrm{BW}_n$$

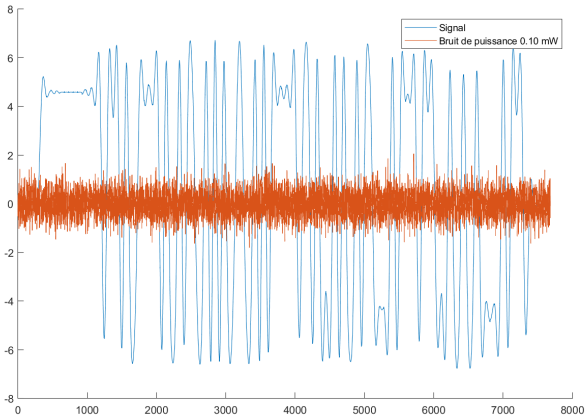
Pour trouver la puissance du bruit qu'on désire obtenir en partant du ratio $\frac{E_b}{N_0}$ on fait :

```
EbN0 = 0.1;
Eb = Pt / R;
N0 = 1/EbN0 * Eb;
```

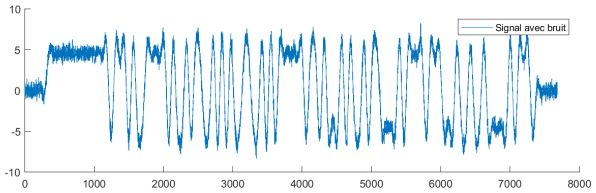
Une fois que nous avons trouvé la densité spectrale de puissance du bruit que nous désirons, imposée par le ratio $\frac{E_b}{N_0}$, nous pouvons normaliser la puissance du bruit comme fait précédemment avec le signal.

```
% Normalize noise power
Pnoise = sum(((noise / Zc) .^ 2), 2) / size(noise, 2);
noise_normalized = noise ./ sqrt(Pnoise) .* sqrt(N0);

m_channel = m_normalized_delayed + noise_normalized;
```



(a) Signal et bruit



(b) Signal avec bruit

FIGURE 3.1 – Signal avec le bruit

3.2.4. Addition des canaux

Finalement, après avoir ajouté les délais, l'atténuation et le bruit, nous devons encore additionner les différents canaux entre eux pour n'obtenir plus qu'un signal qui peut être transmit sur le câble. Comme nos différents canaux sont représentés par des rangées dans la matrice, il nous suffit de faire une addition des éléments par colonne.

```
signal = sum(m_channel, 1);
```

Sur la figure 3.2 on peut voir le signal obtenu par l'addition de quatres canaux pour un message de 100 bits.

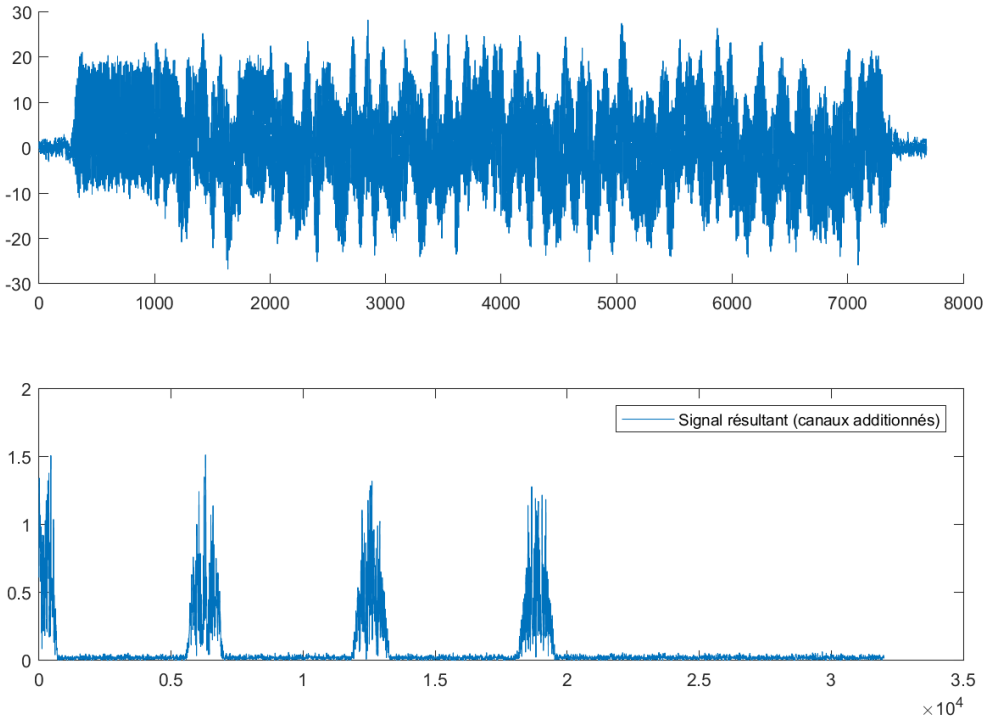


FIGURE 3.2 – Signal obtenu en additionnant les canaux

4

Récepteur

Contrairement aux émetteurs, qui n'envoient chacun que sur un canal fréquentiel, le récepteur doit être en mesure de recevoir un signal pour chaque canal fréquentiel.

4.1. Filtres analogiques

A l'entrée du récepteur, nous recevons un signal analogique, bruité contenant la somme de tous les canaux fréquentiels, comme expliqués précédemment. La première chose que nous devons donc faire est de séparer les N composantes fréquentielles qui correspondent aux N bandes sur lesquels les émetteurs pourraient transmettre.

La séparation se fait à l'aide de N **filtres analogiques**. Le premier filtre, sera de type *passe-bas* et les $N-1$ suivants seront de type *passe-bande*.

Filtre idéal
Un filtre idéal aurait un gain unitaire pour toute la bande passante et un gain nul en dehors. Malheureusement, ce genre de filtre n'est pas réalisable physiquement. On ne peut s'en approcher uniquement en augmentant l'ordre du filtre. L'ordre du filtre correspond plus ou moins au nombres d'éléments réactifs, des composants dont l'impédance dans le circuit varie avec la fréquence.

Filtre réel
En pratique, on se limite à des filtres d'un certain ordre (relativement faible). Les filtres analogiques physiquement réalisables les plus connus sont répartis en 4 grandes fammiles :

— Butterworth
— Chebyshev
— Bessel
— Filtres elliptiques

Quand on dimensionne un filtre analogique, il faudra prendre en compte différents paramètres, tel que : l'ordre du filtre, les fréquences de coupures à 3 dB f_n^- et f_n^+, le niveau d'oscillation dans la bande passante (ripple), ou encore l'affaiblissemement hors-bande.

Pour commencer, nous allons utiliser des filtres de Butterworth.

Si $r(t)$ est le signal reçu à l'entrée du récepteur alors le signal filtré par le n-ième filtre $r_n(t)$ est donné par

$$r_n(t) = r(t) * f_n(t)$$

où $f_n(t)$ est la réponse impulsionelle du filtre analogique correspondant au n-ième canal fréquentiel.

4.1.1. Filtre de Butterworth

Pour implémenter un filtre de Butterworth en MATLAB, nous utilisons la fonction *butter.m* qui permet d'obtenir les coefficients a_i et b_i des polynômes de la fonction de transfert $F_n(f)$ du filtre.

Ensuite nous allons calculer la réponse en fréquence du filtre à l'aide de *freqs*. Ce qui nous permet de prendre la transformée de Fourier inverse afin de trouver la réponse impulsionelle du filtre.

Bibliographie