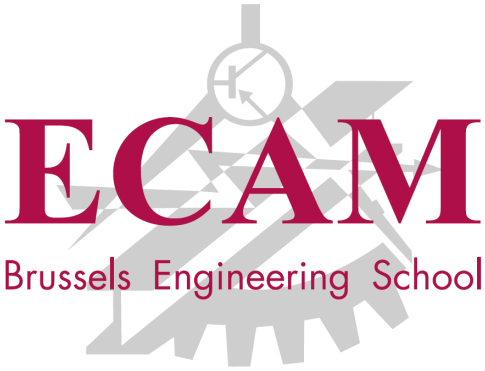

Techniques de transmission et traitement du signal

Simulation d'une chaîne de transmission numérique avec Matlab®

Alexis Nootens
16139@student.ecam.be

Armen Hagopian
14040@student.ecam.be

ECAM Brussels
Promenade de l'Alma 50
1200 Woluwe-Saint-Lambert
Belgique

5 mai 2018

Table des matières

1 Introduction

L'objectif de ce projet est simuler la couche physique d'un protocol de communication, c'est-à-dire le niveau 1 du modèle OSI. La simulation est réalisée avec le logiciel Matlab® édité par Mathworks®. Les contraintes imposées dans la simulation sont de tenir compte de plusieurs émetteurs et receveurs pouvant communiquer en même temps. Pour répondre à cette contrainte, la couche physique implémentée utilise le multiplexage fréquentiel.

Ce document reprend la conception du projet et les choix qui ont dû y être décidés, accompagnés de leur explication.

2 Implémentation

La section 2 décrit le travail apporté dans les fichiers qui composent le projet. Ces fichiers peuvent être consultés à l'annexe A. Ils consistent en :

main.m lance les scripts dans l'ordre logique.

parameters.m configure paramètres de simulation.

sender.m génère les données aléatoirement, puis sépare les canaux fréquentiellement.

channel.m simule un canal de communication en atténuant et filtrant les signaux.

receiver.m démodule les signaux reçus et tente de recomposer le signal émis.

2.1 Fichier principale

Le fichier principale, aussi nommé « main » par son nom anglais, se charge de lancer la simulation dans un ordre logique. Le fichier est consultable à l'annexe A.1. Son contenu est minime, il commence par nettoyer le plan de travail de variables et figures résiduelles. Il lance ensuite les scripts dans l'ordre parameters → sender → channel → receiver.

Une fois la simulation terminée, il affiche une figure comparant le signal dans un canal émis par sender, au signal recomposé dans ce même canal par le receveur. La figure 1 présente un exemple de cette comparaison. On peut y apercevoir que le signal recomposé est délayé par rapport au signal émis, et que son amplitude aux pics isolés est quasiment divisé par deux. Cela est normal étant donné que l'on retrouve plus de fréquence dans un pic isolé que dans une succession à la même amplitude. Ce pic souffrira donc plus fortement au filtrage fréquentiel.

2.2 Paramètres

Le fichier `parameters.m` offre un accès rapide et concentré aux différents paramètres influençant le simulation tel que le nombre de canaux fréquentiel disponibles, la taille du message à envoyer, ou encore la vitesse d'envoie. Ce fichier est consultable à l'annexe A.2. Chaque paramètre est accompagné d'un commentaire expliquant dument son utilité.

2.3 Émetteur

Le fichier `emetteur.m` consultable à l'annexe A.3 commence la simulation à proprement dire. Il débute par générer des bits aléatoire suivant une distribution normale. Il rajoute une séquence de départ pour la forme uniquement, cette dernière n'est pas exploitée dans le receveur. Les bits sont ensuite codés dans un code PAM-2 ou les 0 deviennent -1 et les 1 restent 1.

Afin de diviser le spectre fréquentiel pour y définir des canaux, il faut s'assurer que les messages envoyées se limite à leur bande allouée. Pour ce faire nous utilise le filtre en cosinus surélevé qui à la merveilleuse propriété que pour un signal avec un durée de symbole ξ seconde, il ne consommera que $\frac{\xi}{2}$ largeur de spectre dans le domaine fréquentiel. C'est la meilleur efficacité spectrale atteignable mathématiquement. Un soucis survient avec l'utilisation de ce filtre, c'est que le signal codé maintient

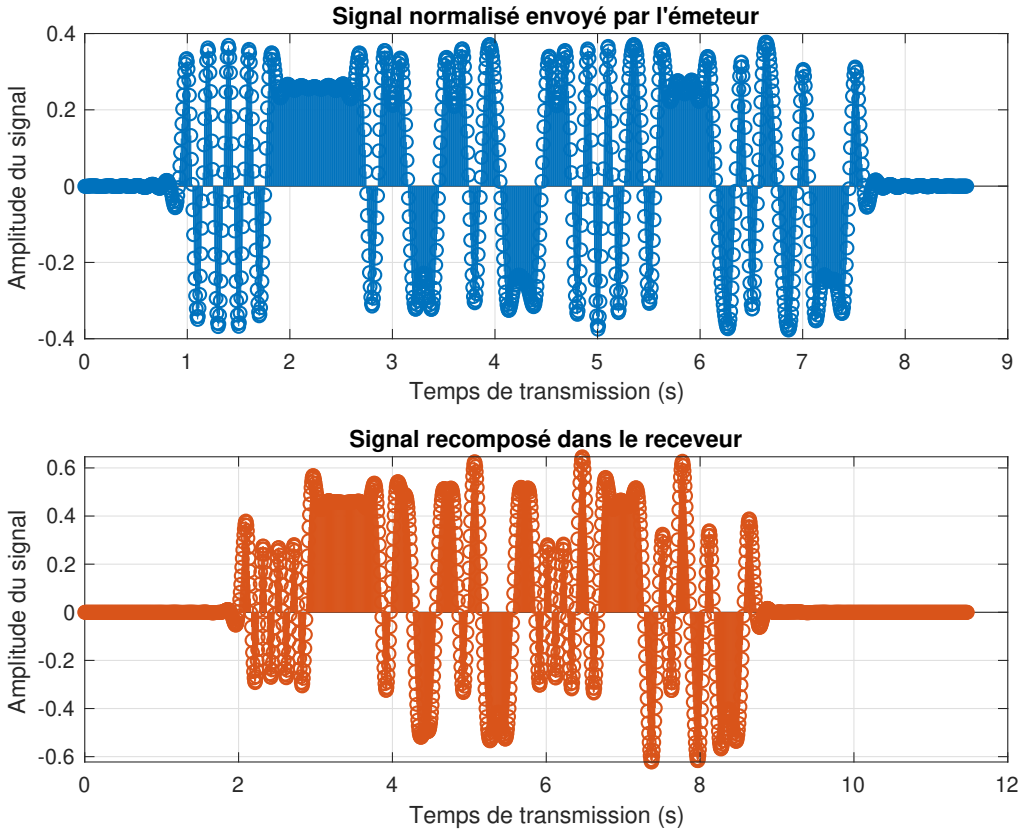

Figure 1 : Comparaison entre le signal en bande de base émis dans l'émetteur et celui recomposé dans le receveur. Le signal reçu est décalé d'approximativement 1 seconde par rapport au signal émis. Le signal reçu est également plus sévèrement atténué aux pics isolés qui contiennent des plus hautes fréquences.

son amplitude maximale durant un bref instant, puis chute rapidement. Ce type de réponse n'est pas utilisable en pratique car la fréquence de capture d'échantillons dans le receveur n'est pas monotone et stable, ni même en phase. Pour adresser ce problème, le filtre maintient son plafond durant un instant plus allongé, mais aux prix de flancs montant et descendant plus raides car la période d'expression d'un symbole n'est pas augmenté. Ces flancs plus raides entrainent une plus grand consommation de bande spectrale, nous définissons une marge au signal de $1 + \alpha$ largeur de bande avec $0 \leq \alpha \leq 1$, dénommé « facteur de roll-off ». Ce facteur contrôle le temps de « plafond » du symbole dans le domaine fréquentiel. Dans notre implémentation, nous avons choisi un facteur de 0,4 arbitrairement.

La description du filtre précédemment faite s'applique au domaine continue. Quand nous passons dans le domaine discret, le temps d'échantillonnage doit être pris en compte. Si nous n'utilisons qu'un seul échantillon pour convoluer le filtre en cosinus surélevé, sa réponse impulsionnelle sera semblable à une impulsion de Dirac, avec un consommation fréquentielle infinie, ce qui est l'opposé de ce qui est désiré. Si nous utilisons une infinité d'échantillon, nous obtiendrons une réponse parfaite semblable au domaine continue. Maintenant que les extrémités sont définies, il est nécessaire de savoir combien d'échantillons aux minimums sont nécessaires pour garder les spectres de différents canaux séparés. L'équation (5) répond à cette question. En partant de l'équation (1) qui fixe les contraintes.

Soit T_n le taux d'échantillonnage nécessaire pour restreindre les canaux à leur bande sans qu'ils ne s'empiètent, T_b le taux d'échantillonnage du signal avant filtrage, N le nombre de canaux actifs et α le facteur de roll-off. Puisque le premier canal non-modulé sera « single-sideband » $\frac{1}{2T_b}$, et les suivants seront « double-sideband » $\frac{1}{T_b}$:

Par le théorème de Nyquist

$$\frac{1}{2T_n} \geq \left[\frac{1}{2T_b} + (N-1)\,\frac{1}{T_b}\right](1+\alpha) \tag{1}$$

Puisque β est le rapport $T_n \div T_b$

$$\frac{\beta}{2T_b} \geq \left[\frac{1}{2T_b} + (N-1)\,\frac{1}{T_b}\right](1+\alpha) \tag{2}$$

En rajoutant $2T_b$ de chaque côté

$$\beta \geq \left[1 + (N-1)\,2\right](1+\alpha) \tag{3}$$

En considérant le pire cas de $\alpha = 1$

$$\beta \geq 2 + (N-1)\,4 \tag{4}$$

En simplifiant

$$\beta \geq 4N - 2 \tag{5}$$

Et puisque nous n'aimons pas ce qui n'est pas linéaire, nous prenons dans la simulation $\beta = 4N$.

Un dernier paramètre à décider dans la construction du filtre numérique et la longueur de la réponse impulsionnelle qui va sert à convoluer le signal. Pour que le filtrage respecte parfaitement les caractéristiques que nous lui avons défini, il lui faut une longueur infini. Ceci n'étant pas possible en pratique, la réponse doit être tronquée tout en gardant le maximum de puissance. Par essaies empiriques, nous avons déterminé que 20 fois le taux de sur-échantillonnage est une bonne valeur.

Les filtres générés et leur réponse impulsionnelle obtenues, nous les modulons par les porteuses avant d'être convoluées avec le signal. Nous obtenons les impulsions présentées à la figure 2. Cette figure présente les 3 impulsions nécessaires pour multiplexer en 3 canaux. Il est intéressant de constater que l'impulsion du canal 2 est inversée en amplitude.

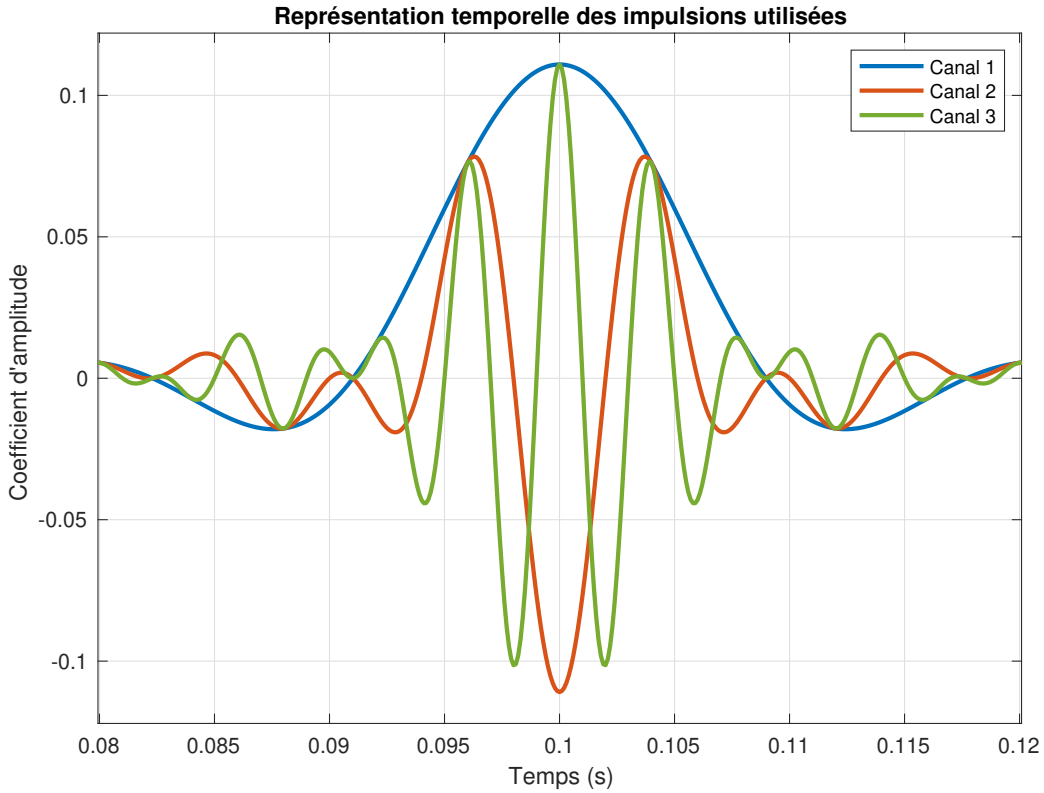

Figure 2 : Représentation temporelle des impulsions utilisées pour multiplexer en 3 canaux. Chaque des impulsions est sa réponse en bande de base modulée par sa porteuse. Il est intéressant de constater que l'impulsion du canal 2 est inversée en amplitude.

Les signaux modulés, leur puissance est normalisée à une quantité de milliwatt par seconde en divisant/multipliant les amplitudes par le facteur adéquat. La puissance de chaque canal est évaluée en intégrant la norme au carré sur l'impédance caractéristique du milieu de propagation.

$$P[s] = \frac{\frac{1}{N}\sum_{n=0}^{N} s[n]^2}{Z_0} \tag{6}$$

Le script `sender.m` affiche un récapitulatif de la transmission en temporel et fréquentiel avant de sommer tous les canaux pour les envoyer sur le seul lien physique disponible, le cable ou le rayonnement électro-magnétique. Un exemple de ce récapitulatif est présenté à la figure 3.

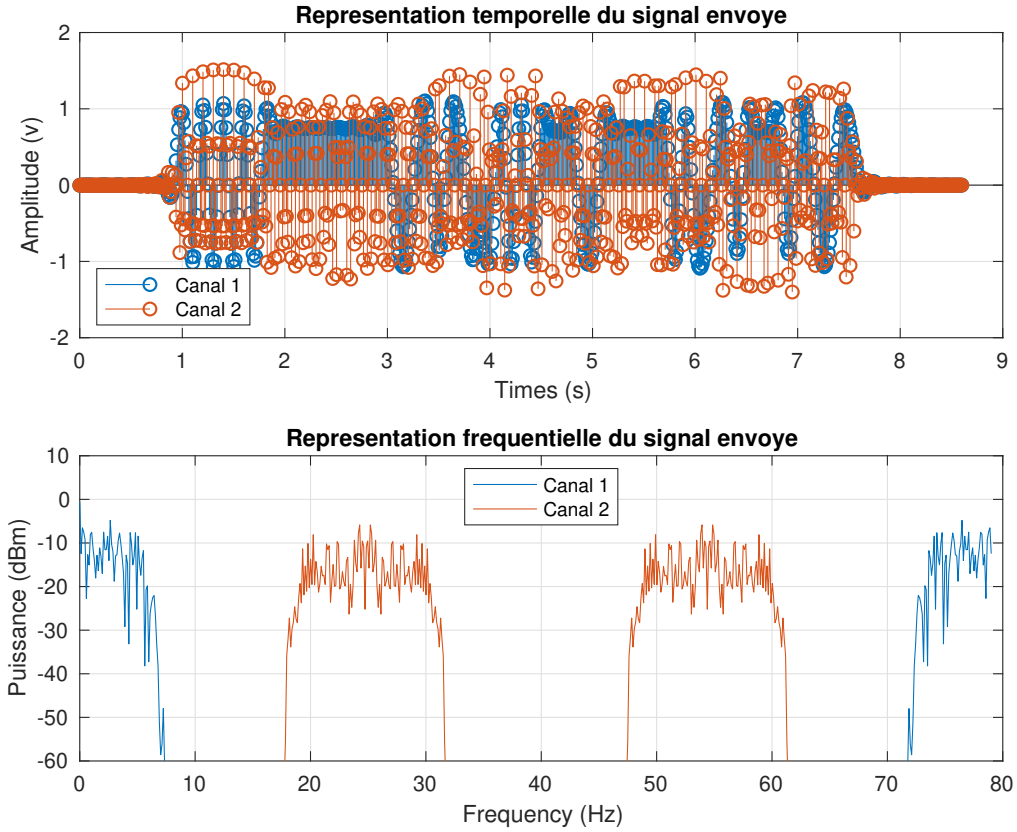

Figure 3 : Signaux de 50 bits chacun envoyés sur 2 canaux fréquentiels au rythme de 10 bit/s avec une fréquence d'échantillonage de 80 Hz.

2.4 Canal

2.5 Receveur

3 Performances

4 Conclusion

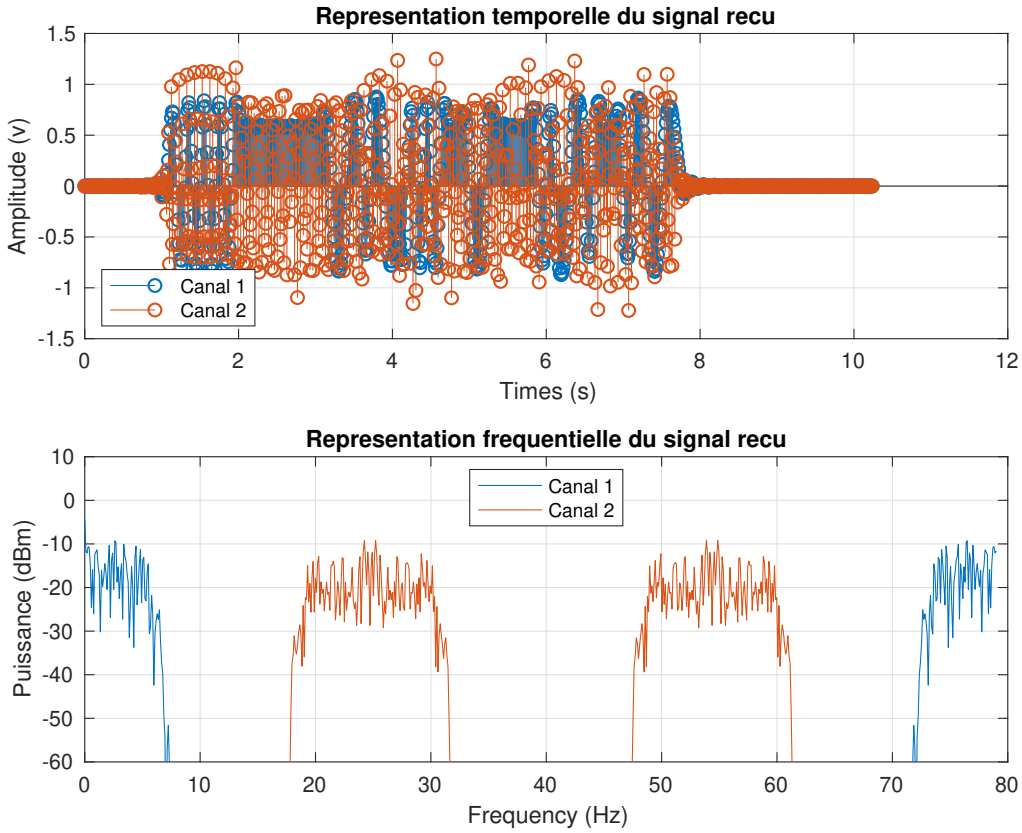

Figure 4 : Signaux de 50 bits chacun envoyés sur 2 canaux fréquentiels au rythme de 10 bit/s avec une fréquence d'échantillonage de 80 Hz.

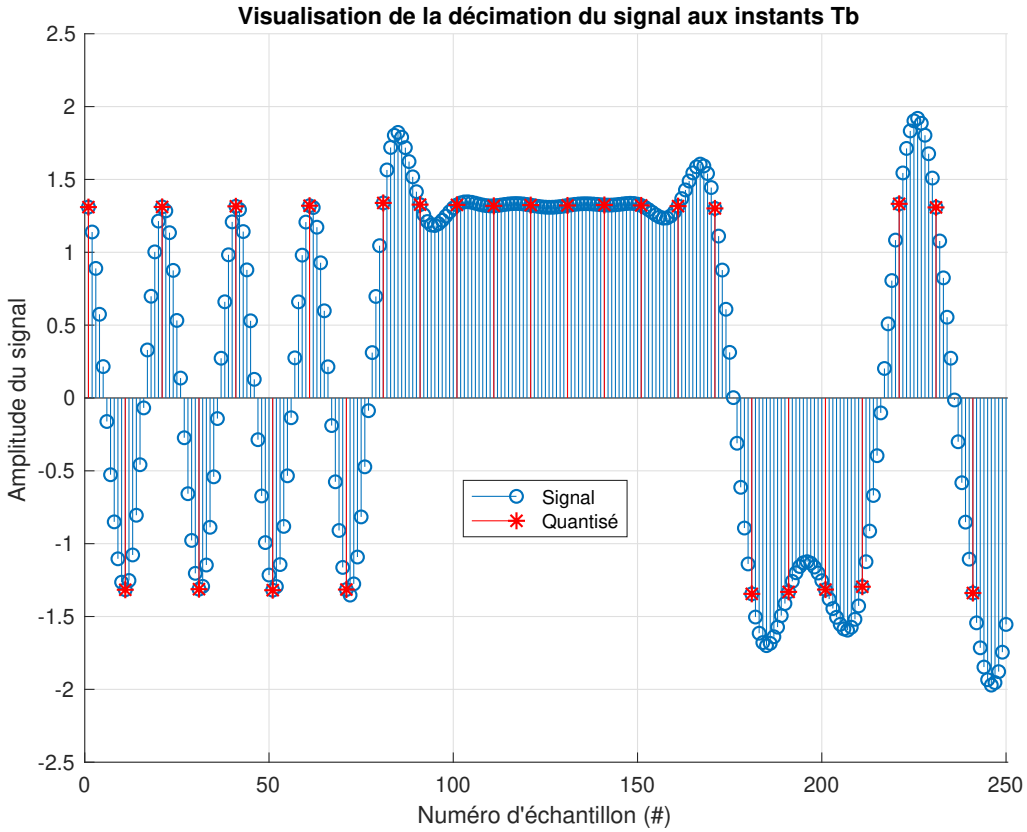

Figure 5

A Fichiers sources

A.1 main.m

```
% This work is licensed under the Creative Commons Attribution 4.0
% International License. To view a copy of this license, visit
% http://creativecommons.org/licenses/by/4.0/ or send a letter to
% Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
clear, close all

parameters
% generate data and send it
sender
% add noise and delay
channel
% filter data and read it
receiver

% compare the sent signal with the received one
figure
subplot(2,1,1)
stem(linspace(0, len1*Tn, len1), s1(:,1));
title('Signal normalise envoye par l''emetteur')
xlabel('Temps de transmission (s)')
ylabel('Amplitude du signal')
grid

subplot(2,1,2)
len3 = size(s2,1);
stem(linspace(0, len3*Tn, len3), s2(:,1), 'Color', [0.85 0.33 0.1]);
title('Signal recompose dans le receveur')
xlabel('Temps de transmission (s)')
```

```
ylabel('Amplitude du signal')
grid

% report QS
disp("Taux d'erreurs :")
disp(sum(xor(x, decoded))/size(x,1))
```

A.2 parameters.m

```
% This work is licensed under the Creative Commons Attribution 4.0
% International License. To view a copy of this license, visit
% http://creativecommons.org/licenses/by/4.0/ or send a letter to
% Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

codesymbol = @(x)x.*2-1;

% System
N = 2;        % available channels
M = 50;       % message size (bits)

% Sender
R = 10;       % bit rate
Tb = 1/R;     % bit duration
roll = 0.40;  % rolloff factor
L = 1.25;     % bandwidth xTb
beta = 4*N;   % upsampling factor
Tn = Tb/beta; % upsample sampling rate
span = 20;    % rcos span for thinner bandwidth consumption
pwr = 1;      % channel power in mW

% Channel
Z0 = 50;      % characteristic impedance
shift = 4;    % samples delay

% Receiver
impulseL = 128;
startSeq = [1 0 1 0 1 0 1 0 ... % test the channel response
            1 1 1 1 1 1 1 1];   % set an unique sequence
```

A.3 sender.m

```
% This work is licensed under the Creative Commons Attribution 4.0
% International License. To view a copy of this license, visit
% http://creativecommons.org/licenses/by/4.0/ or send a letter to
% Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

x = randi([0 1], M, N);
% append the start sequence
x = [startSeq'*ones(1, N); x];
a = codesymbol(x);
% shape to impulse
rcos = rcosdesign(roll, span, beta);
a = upsample(a, beta);
s1 = conv2(rcos, 1, a);
len1 = size(s1, 1);
% carrier frequencies
carfreq = (0:N-1)'*L*2/Tb;

%% plot impulsions
iX = linspace(0, span/1e2, 1e2*span+1);
iY = rcosdesign(roll, span, 1e2);
```

```
plot(iX, iY' * ones(1, N) .* ...
     cos(carfreq*linspace(0, 2*pi, span*1e2+1))')
ylim([-max(iY)*1.1 +max(iY)*1.1])
title("Representation temporelle des impulsions utilisees")
ylabel("Coefficient d'amplitude"), xlabel("Temps (s)")
legend(strcat("Canal ", num2str((1:N)')))
grid
clear iX iY

%% modulate by carriers
t = (0:Tn:(len1-1)*Tn)'*ones(1,N);
s1High = s1.*cos(2*pi*carfreq'.*t);

% normalise power to 'pwr' mW
pwrTimesSec = pwr*len1*Tn; % mW per second * transmission time
avgPower = bandpower(s1High)/Z0*1000/(pwrTimesSec);
s1High = s1High./sqrt(avgPower);

% sum all channels before transmission
data = sum(s1High, 2);

%% plot visual representation of the transmission
figure
subplot(2,1,1)
stem(linspace(0, len1*Tn, len1), s1High)
title('Representation temporelle du signal envoye')
ylabel('Amplitude (v)'), xlabel('Times (s)')
legend(strcat("Canal ", num2str((1:N)')), 'Location', 'SouthWest')
grid

subplot(2,1,2)
plot(linspace(0, 1/Tn-1, len1), pow2db(abs(fft(s1High/len1)).^2/Z0)+30)
ylim([-60 10])
title('Representation frequentielle du signal envoye')
ylabel('Puissance (dBm)'), xlabel('Frequency (Hz)')
legend(strcat("Canal ", num2str((1:N)')), 'Location', 'North')
grid
```

A.4 channel.m

```
% This work is licensed under the Creative Commons Attribution 4.0
% International License. To view a copy of this license, visit
% http://creativecommons.org/licenses/by/4.0/ or send a letter to
% Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

% gaussian noise
noise_1 = randn([numel(data) 1]);
[bf,af] = butter(1, 0.5);
noise_f = ifft(freqz(bf, af, impulseL, 'whole', 1/Tn));
noise_2 = conv(noise_f, noise_1);
noise_2 = noise_2(impulseL/2:end-impulseL/2);

% damping factor; between 0.60<=x<=0.90
alpha = (0.90-0.60)*rand([1 1])+0.60;

% increase noise with variance
variance = 0;
std_dev = sqrt(variance);
noise_3 = noise_2*std_dev;

data = alpha*data+noise_3;
data = [zeros(shift,1); data];
```

A.5 receiver.m

```
% This work is licensed under the Creative Commons Attribution 4.0
% International License. To view a copy of this license, visit
% http://creativecommons.org/licenses/by/4.0/ or send a letter to
% Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

% calculate the bandwidth limits for each channel
cutoff = [carfreq-1/Tb carfreq+1/Tb]*2*Tn;
if N == 1, cutoff = [0 0.9999]; end
% pre-allocate filters matrix
H = zeros(impulseL, N);

% first channel lowpass
[bf,af] = butter(10, cutoff(1,2));
H(:,1) = ifft(freqz(bf, af, impulseL, 'whole', 1/Tn));

% others channels bandpass
for n = 2:N
    [bf,af] = butter(10, [cutoff(n,1) cutoff(n,2)]);
    H(:,n) = ifft(freqz(bf, af, impulseL, 'whole', 1/Tn));
end

% separate channels
s2High = conv2(data, 1, H);

% demodulate
len2 = size(s2High,1);
t = (0:Tn:(len2-1)*Tn)'*ones(1,N);
s2 = s2High.*cos(2*pi*carfreq'.*t);
s2(:,1) = s2High(:,1);
for n = 2:N
    [bf,af] = butter(5, carfreq(n)*2*Tn);
    impulse = ifft(freqz(bf, af, impulseL, 'whole', 1/Tn));
    s2(:,n) = conv(s2(:,n), impulse(1:+1:end), 'same'); % forward
    s2(:,n) = conv(s2(:,n), impulse(end:-1:1), 'same'); % backward
end

% filter the canal noise with the adequate filter
s2 = conv2(rcos, 1, s2);
% find filters delay
[~,i] = max(H);
% compensate the start trame
s2t = s2(span*beta+i+shift-3:end, :);
% generate the index vector
s2i = 1:beta:beta*size(x,1);
% extract the values at index
decoded = s2t(s2i,:);
% quantize the extracted values
decoded = decoded>0;

% hit markers *PEW* *PEW*
figure, hold on
stem(s2t(:,2))
stem(s2i, s2t(s2i,2), 'r*', 'MarkerSize', 8.0)
grid, hold off

%% plot visual representation of the transmission
figure
subplot(2,1,1)
stem(linspace(0, len2*Tn, len2), s2High)
title('Representation temporelle du signal recu')
ylabel('Amplitude (v)'), xlabel('Times (s)')
legend(strcat("Canal ", num2str((1:N)')), 'Location', 'SouthWest')
grid

subplot(2,1,2)
plot(linspace(0, 1/Tn-1, len2), pow2db(abs(fft(s2High/len2)).^2/Z0)+30)
ylim([-60 10])
title('Representation frequentielle du signal recu')
ylabel('Puissance (dBm)'), xlabel('Frequency (Hz)')
legend(strcat("Canal ", num2str((1:N)')), 'Location', 'North')
grid
```