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

2.1 Émetteur

2.2 Canal

2.3 Receveur

3 Performances

4 Conclusion

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(:,2));
title('Signal normalisé envoyé par l''émeteur')
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(:,2), 'Color', [0.85 0.33 0.1]);
title('Signal recomposé 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 = 1e3;        % 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*L;   % 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 control bits
%x = controlbit(x, 7);
% 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("Représentation temporelle des impulsions utilisées")
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('Représentation temporelle du signal envoyé')
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('Représentation fréquentielle du signal envoyé')
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;
% 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('Représentation temporelle du signal reçu')
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('Représentation fréquentielle du signal reçu')
ylabel('Puissance (dBm)'), xlabel('Frequency (Hz)')
legend(strcat("Canal ", num2str((1:N)')), 'Location', 'North')
grid
```