Projets diffusion d'images sur un lien DVB S2

Partie I : étude du système DVB S2

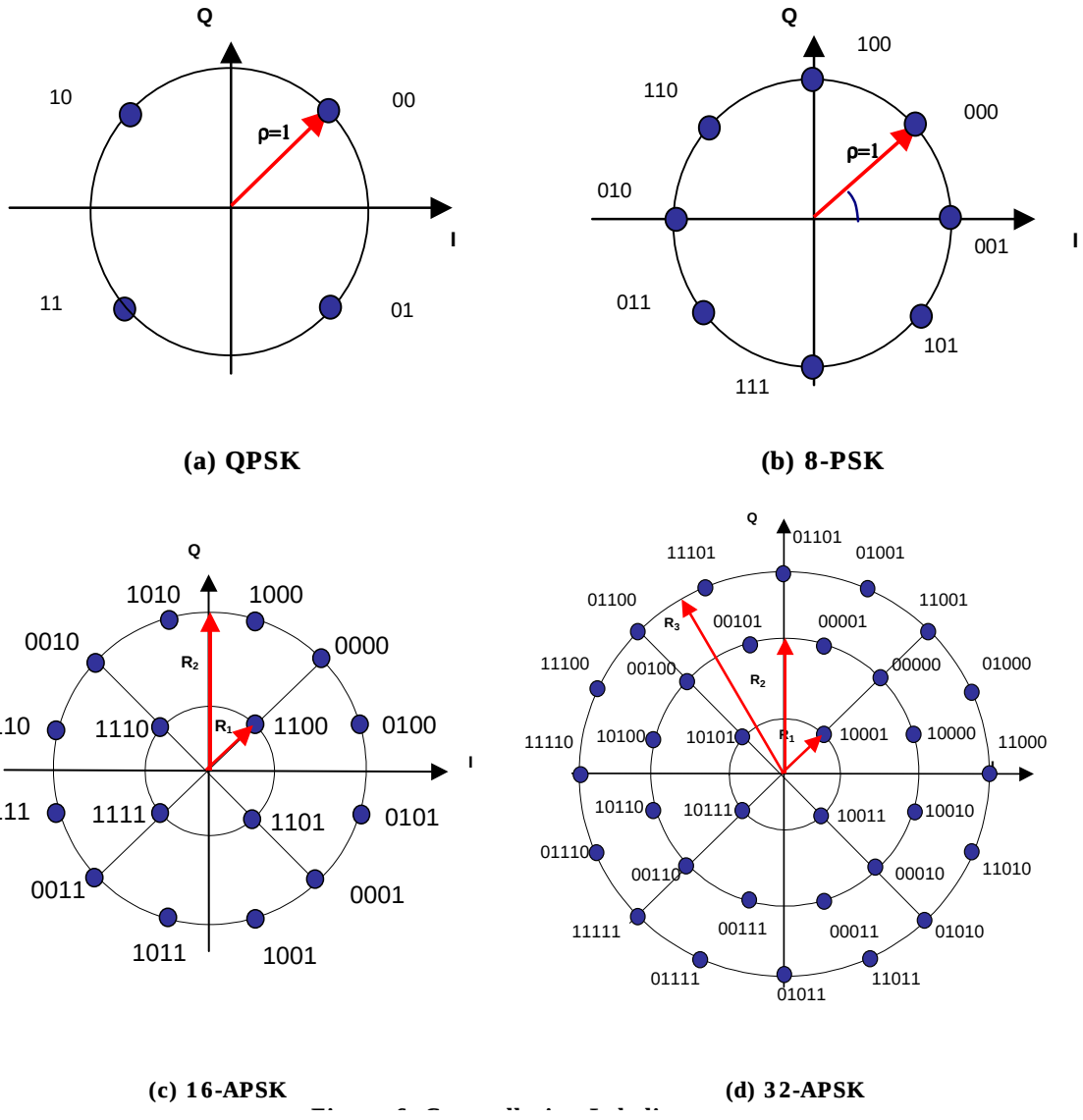
1 Introduction

Introduction L'objectif du projet de communication numérique est de simuler sous Matlab une chaîne de transmission au standard DVB-S2 (diffusion par satellite) et de réaliser la transmission d'un flux compressé sur ce lien afin de réaliser l'impacte d'erreur sur l'applicatif. Le trvail à réaliser sera fait en deux partie : la première concerne la mise en place des éléments de base de la chaîne de transmission DVB S2, la seconde la transmission et la gestion d'un flux compressé dans un canal variant dans le temps et soumis à des non linéarités.

Dans la première partie, nous concidérerons donc le système DVB S2 seul. Le canal de propagation considéré en premier lieu est le canal à bruit additif Gaussien (AWGN). Nous regarderons ensuite l'effet des non linéarité sur le signal reçu et la perte induite au récepteur. La simulation se fera par étapes. Dans un premier temps on mettra en place le couple modulateur/démodulateur. On évaluera alors les performances asymtptotiques du DVB S2 en calculant la capacité du système. On ajoutera ensuite le codage canal. Enfin, on considèrera d l'influence de non linéarité sur le système.

2 Mise en oeuvre du modulateur démodulateur

Le modulateur DVB-S2 utilise différentes modulations : QPSK,8-PSK, 16-APSK et 32-APSK (représentées Figure 1 et Table 1). On se limitera ici aux trois premières uniquement. Le filtrage de mise en forme est réalisé un filtre en racine de cosinus surélevé (SRRCF : Square root raised cosine filter). La densité spectrale bilatéralé de puissance associée au bruit $n(t)$ introduit par le canal s'écrit $S_n(f) = N0/2$. Afin de réduire le temps de simulation on travaillera sur la chaine équivalente bande de base associée à la chaine de transmission sur fréquence porteuse (voir Figs. 2 et 3).



(a) QPSK **(b) 8-PSK**

(c) 16-APSK **(d) 32-APSK**

FIGURE 1 – Modulations possibles sur DVB-S2

Code rate	Modulation/coding spectral efficiency	γ
2/3	2,66	3,15
3/4	2,99	2,85
4/5	3,19	2,75
5/6	3,32	2,70
8/9	3,55	2,60
9/10	3,59	2,57

Code rate	Modulation/coding spectral efficiency	γ_1	γ_2
3/4	3,74	2,84	5,27
4/5	3,99	2,72	4,87
5/6	4,15	2,64	4,64
8/9	4,43	2,54	4,33
9/10	4,49	2,53	4,30

TABLE 1 – Tableau sur les rayons optimum sur DVB-S2

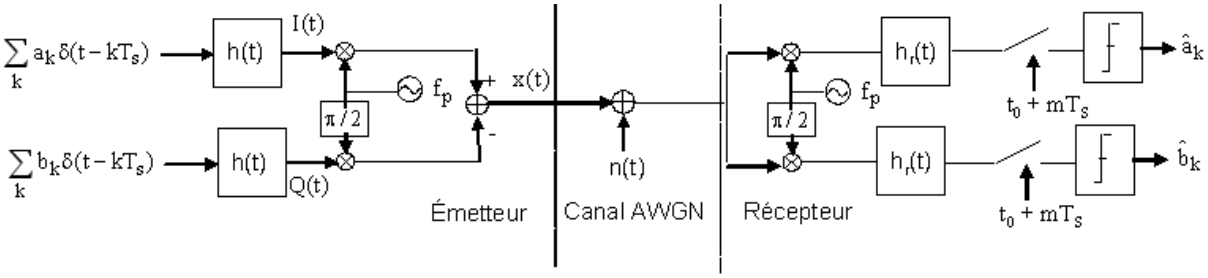


FIGURE 2 – Chaîne de transmission sur porteuse

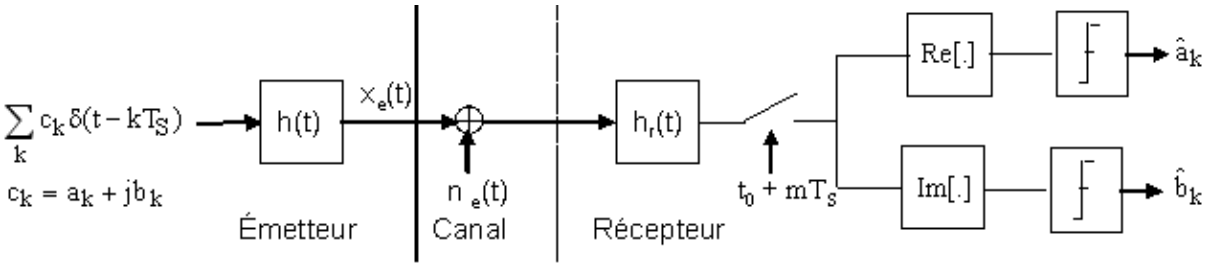


FIGURE 3 – Chaîne de transmission passe-bas équivalente

2.1 Génération de l'enveloppe complexe associée au signal à transmettre

Pour chaque modulation, on générera une suite de symboles équiprobables et indépendants, s_k, conformément aux constellation du standard DVB-S2 (voir Fig. 1 En utilisant la fonction *kron* de matlab, on génerera la suite d'impulsions de Dirac correspondante.

Pour la génération des symboles, on utilisera de préférence les fonctions Matlab. La génération de symboles PSK est illustrèe par le code suivant :

```
% Construct a modulator object for QPSK modulation.
h = modem.pskmod(4)

% Construct a modulator object for 8-PSK modulation with
% constellation shifted by pi/8 radians.
h = modem.pskmod(8, pi/8)

% Construct an object to modulate binary data using 16-PSK .
% modulation. The constellation has Gray mapping and is
% shifted by -pi/16 radians.
h = modem.pskmod('M', 16, 'PhaseOffset', -pi/16, ...
                 'SymbolOrder', 'Gray', 'InputType', 'Bit')
```

Pour générer des symboles à l'aide des précédents objets, on utilisera les fonctions suivantes

```
%Matlab modulate function
modulatedsig = modulate(h, message);
```

Le constructeur pour la modulation 16-APSK et la modulation du signal sera réalisé à l'aide des fichiers fournis :

```
% 16 APSK constellation and bit Mapping
gamma = gamma_dvbs2(R);
[constellation, bitMapping] = DVBS2Constellation('16APSK',gamma);

% Modulate the signal for 16-APSK
modulatedsig = mod_ 16apsk(codword,gamma);
```

La période symbole T_s sera composée de N échantillons : $T_s = NT_e$, T_e étant la période d'échantillonnage et N le nombre d'échantillons par symbole. On mettra en forme en utilisant un filtre en racine de cosinus surélevé de roll-off α qui pourra prendre différentes valeurs comme spécifié par le standard $\alpha = \{0.2, 0.25, 0.35\}$ et de réponse impulsionnelle notée $h(t)$. La synthèse de $h(t)$ pourra être effectuée en utilisant par exemple la fonction *rcosfir()* de Matlab.

```
function K = kron(A,B)
%KRON   Kronecker tensor product.
%   KRON(X,Y) is the Kronecker tensor product of X and Y.
%   The result is a large matrix formed by taking all possible
%   products between the elements of X and those of Y.   For
%   example, if X is 2 by 3, then KRON(X,Y) is
%
%      [ X(1,1)*Y  X(1,2)*Y  X(1,3)*Y
%        X(2,1)*Y  X(2,2)*Y  X(2,3)*Y ]
%
```

```
%   If either X or Y is sparse, only nonzero elements are multiplied
%   in the computation, and the result is sparse.
```

```
function [bb, tim] = rcosfir(r, N_T, rate, T, fil_type, col)
%RCOSFIR Design a raised cosine FIR filter.
%   B = RCOSFIR(R, N_T, RATE, T) designs and returns a raised cosine FIR filter.
%   A raised cosine filter is typically used to shape and oversample a symbol
%   stream before modulation/transmission as well as after reception and
%   demodulation.  It is used to reduce the bandwidth of the oversampled symbol
%   stream without introducing intersymbol interference.
%
%   The time response of the raised cosine filter is,
%
%     h(t) = SINC(t/T) COS(pi R t/T)/(1 - 4 R^2 t^2 /T^2)
%
%   The frequency domain has the spectrum
%
%            /  T                                      when 0 < |f| < (1-r)/2/T
%            |           pi T          1-R    T          1-R          1+R
%     H(f) = < (1 + cos(----) (|f| - ----) ---    when ---- < |f| < ----
%            |             r           2T    2          2 T          2 T
%            \  0                                      when |f| > (1+r)/2/T
%
%
%   T is the input signal sampling period, in seconds.  RATE is the
%   oversampling rate for the filter (or the number of output samples per input
%   sample). The rolloff factor, R, determines the width of the transition
%   band.  R has no units.  The transition band is (1-R)/(2*T) < |f| <
%   (1+R)/(2*T).
%
%   N_T is a scalar or a vector of length 2.  If N_T is specified as a
%   scalar, then the filter length is 2 * N_T + 1 input samples.  If N_T is
%   a vector, it specifies the extent of the filter.  In this case, the filter
%   length is N_T(2) - N_T(1) + 1 input samples (or
%   (N_T(2) - N_T(1))* RATE + 1 output samples).
%
%   The default value for N_T is 3.  The default value of RATE is 5.
%   The default value of T is 1.
%
%   B = RCOSFIR(R, N_T, RATE, T, FILTER_TYPE) designs and returns a
%   square root raised cosine filter if FILTER_TYPE == 'sqrt'. The default
%   value of FILTER_TYPE, 'normal', returns a normal raised cosine filter.
%
%   RCOSFIR(R, N_T, RATE, T, FILTER_TYPE, COL) produces the time response
%   and frequency response with the curve color as specified in the string
%   variable COL. The string in COL can be any type as defined in
%   PLOT.  If COL is not present, the default color will be used in the plot
%
%   [B, Sample_Time] = RCOSFIR(...) returns the FIR filter and the output sample
%   time for the filter. Note that the filter sample time is T / RATE.
%
%   See also RCOSIIR, RCOSFLT, RCOSINE, FIRRCOS, RCOSDEMO, GAUSSFIR.
```

2.2 Mise en place du récepteur en l'absence de canal

On placera en réception un filtre permettant de respecter le critère de Nyquist. On mettra ensuite en place l'échantillonneur. On choisira l'instant optimal d'échantillonnage en faisant at-

tention de prendre en compte les retards introduits par les filtres de la chaine de transmission. On vérifiera que le taux d'erreur symbole est bien nul dans ce cas.

2.3 Canal de transmission et démodulation souple

On ajoutera ensuite un canal AWGN (canal satellite fixe, voie aller) à la chaîne de transmission précédente. On pourra utiliser plusieurs puissances de bruit que l'on calculera en fonction de E_s/N_0 (rapport signal sur bruit par symbole codé). On rappelle que la variance du bruit à appliquer sur les voies en phase et quadrature du bruit complexe ne(t) s'écrit en fonction du E_s/N_0 souhaité de la manière suivante :

$$\sigma_{n_I}^2 = \sigma_{n_Q}^2 = \frac{\sum_n |h(n)|^2 \sigma_s^2}{2E_s/N_0}, \tag{1}$$

où σ_s^2 représente la variance des symboles s_n, $h(n)$ la réponse impulsionnelle numérique du filtre de mise en forme et E_s/N_0 le rapport signal à bruit par symbole à l'entrée du récepteur.

Pour la démodulation souple, on implantera ensuite l'organe de décision à l'aide des fonctions natives de Matlab qui permettent de générer des *log-likelihood ratios (LLR)*. On pourra vérifier que le taux d'erreur binaire est bien nul à fort rapport signal sur bruit.

Les fonctions de démodulations suivent l'exemple suivant :

```
% Construct a demodulator object from an existing
% modulator object for PSK modulation in order to
% compute approximate log-likelihood ratio for
% a baseband signal whose estimated noise variance is 0.81.
modObj = modem.pskmod('M', 8, 'InputType', 'Bit')
demodObj = modem.pskdemod(modObj, 'DecisionType', ...
           'LLR', 'NoiseVariance', 0.81)

% Compute log-likelihood ratios (AWGN channel) from noisy received signal
llr = demodulate(demodObj, noisedsig);
```

Pour le cas de la modulation 16-APSK, on réalisera une fonction de démodulation souple spécifique.

3 Codage de canal et entrelacement

Le standard DVB-S2 spécifie un codage canal constitué de deux codes concaténés : un code BCH suivi d'un code LDPC. Dans un premier temps, on ne mettra en oeuvre que les codes LDPC pour la taille du mode standard $N = 64800$ bits codés. Pour générer le code, coder et décoder, on utilisera les fonctions natives de Matlab données comme données si après. Un exemple de déclaration est donné ci-dessous.

```
% LDPC construction
R=2/3;                % Rate of the LDPC code
H = dvbs2ldpc(R);     % Construct DVB-S2 LDPC matrix with rate R
enc = fec.ldpcenc(H); % Construct an LDPC encoder object linked to H

% Construct a companion LDPC decoder object
dec = fec.ldpcdec(H);
```

Modulation	Rows (for n_{ldpc} = 64 800)	Rows (for n_{ldpc} = 16 200)	Columns
8PSK	21 600	5 400	3
16APSK	16 200	4 050	4
32APSK	12 960	3 240	5

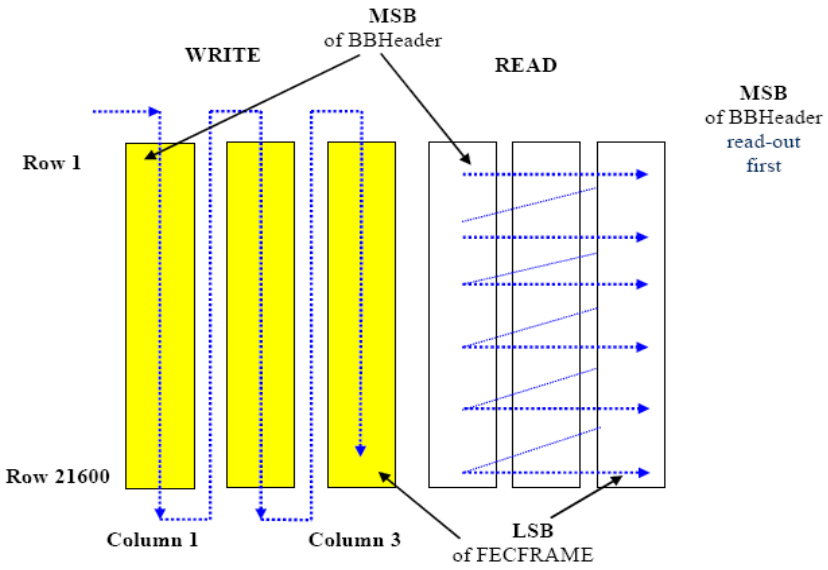


FIGURE 4 – Paramètres d'entrelacement

```
dec.DecisionType = 'Hard decision';
dec.OutputFormat = 'Information part';
dec.NumIterations = 50;

% Stop if all parity-checks are satisfied
dec.DoParityChecks = 'Yes';
```

L'encodage peut sécrire de la manière suivante :

```
% Generate and encode a random binary message
msg = randint(1,enc.NumInfoBits,2);
codeword = encode(enc,msg);
```

Le décodage est réalisé avec la function suivante à partir des LLRs reçus après démodulation :

```
% Decode received signal using LLR and 'dec' LDPC decoder object
decodedmsg = decode(dec, receivedllr);
```

L'entrelacement est un entrelacement ligne-colonne dont les caractéristiques sont données Fig. 4 avec un exemple d'utilisation pour la 8-PSK pour des rendements différent de $R = 3/5$.

4 Travail à réaliser

1. Mettre en place les différents éléments de la chaîne,
2. Calculer les capacités des systèmes mis en oeuvre pour différentes modulations,
3. Calculer les taux d'erreurs binaires pour les différentes modulations et différents rendement pour 50 itérations de décodage. En déduire les plages d'utilisation des différents couples (rendement, modulation),
4. Comparer l'efficacité du système à l'efficacité spectrale théorique pour les différents modes pour un taux d'erreur binaire cible de 10^{-5},
5. si vous le temps, introduire le code BCH.