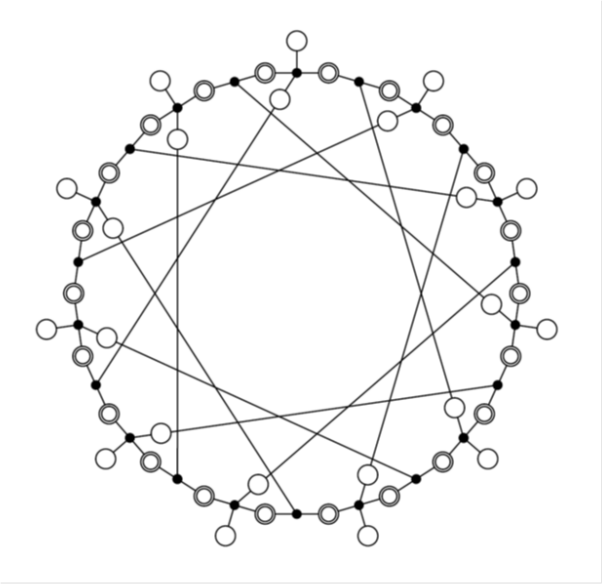
Département Sciences du Numérique

Parcours Télécommunications

Codage et détection avancés

Auteur : C. Poulliat



Version 1.0 du
21 octobre 2021

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction à la théorie de l'information

La théorie de l'information introduite par Claude Shannon en 1948 permet de définir un cadre théorique qui permet d'analyser les performances limites d'un système de communication. En utilisant un cadre probabiliste pour l'analyse des performances d'un système de communication, Shannon a montré que l'on pouvait séparer *asymptotiquement* le problème de transmission d'une source au travers d'un canal de communication physique point-à point en deux grande classes de problèmes : (a) la mise en forme de la source puis (b) la transmission sur le canal. Ces deux grandes étapes sont associées à deux grands domaines de l'ingéniérie des communications numériques : compression de source (voir cours 2A et 3A) et le codage de canal. Une représentation de la vision initiale d'une chaîne de communication dîte point-à-point est alors donnée figure **??**. On remarquera qu'à l'époque la couche réseau n'était pas considéré, ces derniers n'apparaissant que fin des années 60. Alors que la compression (encore appelée codage de source avec ou sans perte) s'intéresse à la représentation d'une source (analogique discrétisée le plus souvent) par une séquence de bits permettant de s'approcher de l'entropie minimale associée à cette source pour une distorsion donnée, la théorie du codage canal permet elle d'introduire une redondance structurée de l'information compressée afin de corriger des erreurs et de permettre d'opérer le plus proche possible de ce que l'on appellera la capacité du canal.

Notion de source

Par ma suite, on traitera essentiellement des sources d'information qui seront modélisée par un processus aléatoire. Une *source d'information* est donc représentée par une suite de symboles $(X_0, X_1, \cdots, X_n, \cdots)$. Chaque symbole X_i appartient à un alphabet discret $\mathcal{X}$, de cardinalité finie. X_i est associée à une variable aléatoire et donc $\{X_n\}_{n\in\mathbb{N}}$ définit un processus aléatoire. On suppose défini

$$p(X_0 = x_0, \cdots X_1 = x_1, \cdots, X_n = x_n),\ \forall n.$$

. Une source d'information est dîte *stationnaire* si toute distribution de probabilité de symboles est invariante par translation temporelle, ie.

$$\forall n, \forall l, \forall (x_0, x_1, \cdots, x_n) \in \mathcal{X}^n, p(X_0 = x_0, \cdots, X_n = x_n) = p(X_{0+l} = x_0, \cdots, X_{n+l} = x_n).$$

Une source stationnaire est dîte sans mémoire si les symboles X_n sont produits de manière indépendante des symboles source passés, ie. $X_i,\ \forall i = 0 \cdots n-1$. On a ainsi

$$p(X_n | X_{n-1} \cdots X_0) = p(X_n).$$

On en déduit

$$p(X_0, \cdots, X_n) = p(X_0) \cdots p(X_n).$$

On a donc les symboles X_i indépendants et par stationnarité, les X_i sont identiquement distribués.

Définition 1 : *Source sans mémoire*

Une source sans mémoire est définie par une suite de symboles $X_0,\ X_1, \cdots X_n$ indépendants et identiquement distribués (on parle de processus aléatoire à variablez aléatoires i.i.d.).

Pour une source discrète dsans mémoire, X une variable aléatoire discrète à valeurs dans l'alphabet $\mathcal{X}$ de d.d.p. discrète $p(x) = \text{Prob}(X = x), x \in \mathcal{X}$. Une source discrète sans mémoire produit une séquence de symboles i.i.d à valeurs dans $\mathcal{X}$ et suivant $p(x)$. Un exemple est une source de symboles issus d'une constelationn M-aire.

Pour une source continue sans mémoire, X une variable aléatoire pouvant prendre des valeurs réelles dans un intervalle $\mathcal{X} \subset \mathbb{R}$ de d.d.p. $f(x)$. Une source (absolument) continue sans mémoire produit une séquence de symboles i.i.d à valeurs dans $\mathcal{X}$ et suivant la d.d.p. $f(x)$. Un exemple classique est donné par un processus Gaussien.

Notion de canal

Le canal de transmission sera caractérisé par une approche probabiliste pour la modélisation. Pour un *canal discret sans mémoire*, on émet un symbol $X = x$ où $X \in \mathcal{X}$ est une valeur aléatoire à valeur dans un alphabet fini et caractérisé par une distribution des symboles $p(x)$. A la sortie du canal on observe la $Y = y$ qui peut être à valeur soit dans un alphabet discret ou appartenir à un sous ensemble $\mathcal{I} \in \mathbb{R} ou \mathcal{C}$. la définition du canal se fait autravers de ce que l'on appelle la probabilité de transition du canal donnée par la vraisemblance $p(y \mid x)$. Elle représente la probabilité de recevoir $Y = y$ sachant que l'on a émis $X = x$. Si X est une variable aléatoire (v.a.) M -aire et Y une v.a. N-aire, on appelle matrice de transition du canal la matrice de taille $N \times M$ donnée par

$$\Pi = (p(y \mid x))_{y,x}.$$

La somme de chaque colonne de Π vaut 1 et on a

$$p_Y = \Pi p_X$$

avec $p_Y = (p(y))_y$ et $p_X = (p(x))_x$ vecteurs colonnes.

1.1 Mesure de l'information : notion d'entropie

Information propre

Soit X une variable aléatoire discrète et $X = x$ un événement de probabilité $p(x)$, une mesure de l'information, notons-là $h(x)$, s'identifie à une mesure de l'inattendu, de l'improbable. Ainsi, une information apportée par la réalisation de l'événement X sera d'autant plus importante que celle-ci est peu probable, ie.

$$h(x) = f\left(\frac{1}{p(x)}\right).$$

Concernant la notion d'information, les propriétés attendues sont :

1. $f(.)$ est une fonctionnelle croissante de $p(x)$,
2. $f(p) = 0$ quand $p \to 1$ (événement certain)
3. $f(p \cdot q) = f(p) + f(q)$ (additivité de l'information pour des événements indépendants : $h(x \text{ et } y) = h(x) + h(y)$).

Compte tenu de cette axiomatique, la fonction $f(p) = -\log(p)$ est la seule fonction qui soit à la fois positive, continue sur $[0, 1)$ et qui vérifie l'additivité des informations indépendantes. La base du logarithme est elle indifférente.

Définition 2 : *Information propre*

Soit X une variable aléatoire discrète et $X = x$ un événement de probabilité $p(x)$, on appelle information propre ou quantité d'information apportée par l'événement x, la quantité

$$h(x) = \log\left(\frac{1}{p(x)}\right) = -\log(p(x))$$

Propriétés 1 : *Propriétés de l'information propre*

1. positivité : $h(x) \geq 0$
2. additivité : soient x et y deux événements indépendants, alors

$$h(x \text{ et } y) = h(x) + h(y)$$

Quand le la base du logarithme est la base naturel ($\log_e(.)$), on parle de Shannon (Sh.) ou d'unité naturelle, notée *nats* pour *natural units*. Si on utilise un logarithm en base 2 ($\log_2(.)$), on parte unité binaire, notée *bits* pour *binary units*. Ainsi, pour une source binaire à valeur dans $\{0, 1\}$ equi-distribuée de symboles indépendants, l'information propre associée à chaque symbole binaire est $h(1/2) = 1$ bit ou Sh. Pour une source M -aire à valeur dans $\{0, 1 \cdots, M-1\}$ equidistribuée de symboles indépendants, l'information propre associée à chaque symbole est $h(1/M) = \log_2(M)$ bits ou Sh.

1.1.1 Entropie associée à une variable aléatoire discrète

Définition 3 : *Entropie*

Soit $\mathbf{X}$ une variable aléatoire discrète à valeurs dans l'alphabet $\mathcal{X}$ de d.d.p. $p(x) = Prob(X = x)$, $x \in \mathcal{X}$, alors l'entropie associée est donnée par

$$\begin{aligned} \mathbf{H}(X) &= -\sum_{x \in \mathcal{X}} p(X = x) \log_2 \left(p(X = x)\right) \\ &= -\mathbb{E}(\log_2 p(X)) \end{aligned} \tag{1.1}$$

C'est la "quantité d'information moyenne" exprimée en bits/symbole.

Propriétés 2 : *Propriétés de l'entropie*

- $\mathbf{H}(X)$ est déterministe et c'est une fonction(nelle) de $p(x)$,
- $\mathbf{H}(X) \geq 0$ (positivité),
- $\mathbf{H}(X) = 0 \Leftrightarrow X$ est déterministe,
- $\mathbf{H}(X) = \log_2(M)$ pour distribution uniforme de symboles M-aire,
- invariance par équivalence (ie. $Y = f(X)$ où $f(.)$ inversible),
- l'entropie d'une source M-aire vérifie

$$\mathbf{H}(X) \leq \log_2(M)$$

avec égalité pour une source à distribution uniforme.

Exemple 1 : *Entropie binaire*

Soit X à valeurs dans un alphabet binaire $\mathcal{X} = \{x_0, x_1\}$ tel que $P(X = x_0) = p$ and $P(X = x_1) = 1 - p$. Alors

$$\mathrm{H}(X) = \mathrm{H_b}(p)$$

où $\mathrm{H_b(p)}$ est ce que l'on nomme la fonction d'entropie binaire donnée par

$$\mathrm{H_b}(p) \triangleq p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$$

1.1.2 Entropie conjointe et conditionnelle

Définition 4

Entropie conjointe soient X et Y deux variables aléatoires discrètes

$$\begin{aligned} \mathbf{H}(X,Y) &= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(X = x, Y = y) \log_2 (p(X = x, Y = y)) \\ &= -\mathbb{E}(\log_2 p(X,Y)) \end{aligned} \tag{1.2}$$

On remarque que

$$\mathbf{H}(X,Y) = \mathbf{H}(Y,X).$$

Définition 5 : *Entropie de Y sachant $X = x$*

soient X et Y deux variables aléatoires discrètes, alors l'entropie de Y sachant $X = x$ est donnée

$$\begin{aligned} \mathbf{H}(Y|X = x) &= -\sum_{y \in \mathcal{Y}} p(Y = y|X = x) \log_2 (p(Y = y|X = x)) \\ &= -\mathbb{E}(\log_2 p(Y|X = x)) \end{aligned} \tag{1.3}$$

Définition 6 : *Entropie conditionnelle*

$$\begin{aligned}\mathbf{H}(Y|X) &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(X=x,Y=y)\log_2\left(p(Y=y|X=x)\right)\\ &= \sum_{x\in\mathcal{X}} p(X=x)\mathbf{H}(Y|X=x) = \mathbb{E}(\mathbf{H}(Y|X=x))\\ &= -\mathbb{E}(\log_2 p(Y|X)) \qquad (1.4)\end{aligned}$$

Propriétés 3 : *Propriétés de l'entropie conjointe et conditonnelle*

1. **chain rule :** $\mathbf{H}(X,Y) = \mathbf{H}(X) + \mathbf{H}(Y|X) = \mathbf{H}(Y) + \mathbf{H}(X|Y)$,
2. **borne inf. :** $\mathbf{H}(X,Y) \geq \mathbf{H}(X)$ ou $\mathbf{H}(Y)$
3. **Conditionnement :** $\mathbf{H}(X|Y) \leq \mathbf{H}(X)$
   égalité si X et Y indépendants
4. **Décroissance par conditionnement :** $\mathbf{H}(X_1|X_2,\cdots,X_n) \leq \cdots \leq \mathbf{H}(X_1|X_2,X_3) \leq \mathbf{H}(X_1|X_2) \leq \mathbf{H}(X_1)$
5. **Encadrement (sous additivité de l'entropie) :**
   $$\mathbf{H}(X,Y) \leq \mathbf{H}(X) + \mathbf{H}(Y) \leq 2\mathbf{H}(X,Y)$$
6. **Entropie conjointe et conditionnement :**
   $$\mathbf{H}(X,Y|Z) = \mathbf{H}(X|Z) + \mathbf{H}(Y|X,Z)$$
7. **positivité :** $\mathbf{H}(X|Y) \geq 0$
   égalité si $X = f(Y)$ où $f(.)$ déterministe

L'ensemble des définition précédentes se généralise assez facilement au cas de vecteurs de dimension supérieure à 2. En particulier, soit $X_1, X_2, \cdots X_n$ de loi conjointe $p(x_1, x_2, \cdots, x_n)$, on aura par définition

$$\mathbf{H}(X_1, X_2, \cdots X_n) = -\mathbb{E}(log_2(p(X_1, X_2, \cdots X_n))).$$

On peut vérifier que

$$\mathbf{H}(X_1, X_2, \cdots X_n) \leq \sum_{i=1}^{n} \mathbf{H}(X_i),$$

avec égalité si et seulement si les X_i sont indépendants. La relation chaînée associée à l'entropie est quant à elle donnée par

$$\mathbf{H}(X_1, X_2, \cdots X_n) = \sum_{i=1}^{n} \mathbf{H}(X_i|X_{i-1}, \cdots X_1).$$

1.1.3 Entropie(s) associée(s) à une variable aléatoire continue

Si la notion d'information est bien reliée à l'entropie d'une variable aléatoire discrète, ce lien est moins évident pour une variable aléatoire continue.

Définition 7 : *Entropie différentielle*

Soit X une variable aléatoire continue définie par une densité de probabilité $f(x)$, alors l'entropie différentielle est donnée

$$h(X) = -\int f(x)\log_2(f(x))dx$$

On ne peut pas interpréter $h(X)$ comme une mesure d'information ou d'incertitude dans le cas continue. Ceci peut se voir dans le cas d'un changement de variable. Soit $Y = f(X)$, par changement de variable, on a $h(X) \neq h(Y) = h(f(X))$ donc $h(X)$ n'est pas une mesure d'information stricte. Dans le cas particulier du changement d'échelle, tel que $Y = aX$, on a $h(X) \neq h(aX) = h(X) + \log(a)$ qui peut même être négatif !

Exemple 2

1. Loi uniforme sur $[a, b]$:
$$f(x) = \frac{1}{b-a}$$
$$h(x) = \log(b-a)$$
2. Loi normale de moyenne μ et variance σ^2 :
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
$$h(x) = \frac{1}{2}\log 2\pi e + \log(\sigma)$$

Comme pour le cas discret, on peut définir des entropies conjointes et conditionnelles. Soit $X_1, X_2, \cdots, X_n$ associées à la densité conjointe $f(x_1, x_2, \cdots, x_n)$, l'entropie différentielle conjointe est définie comme suit

$$h(X_1, X_2, \cdots, X_n) = -\int f(x_1, x_2, \cdots, x_n)\log_2\left(f(x_1, x_2, \cdots, x_n)\right)dx_1dx_2\cdots dx_n.$$

De même pour deux v.a. X et Y, l'entropie différentielle conditionnelle de X sachant Y est donnée par

$$h(X \mid Y) = -\int f(x,y)\log_2(f(x \mid y))dxdy.$$

Ces quantités sont sutout utiles car elles permettent de calculer l'information mutuelle entre deux variables aléatoires, quantité fondamentale en théorie de l'information et qui pour le coup à la même interprétation en discret et en continu.

1.2 Information mutuelle

Pour caractériser la similarité au sens statistique de deux variables aléatoires, on peut utiliser ce que l'on appelle l'information mutuelle qui est définie comme suit

Définition 8 : *Information mutuelle*

Soit X, Y deux v.a. discrètes, l'information muettele entre X et Y est définie par

$$\begin{aligned} \mathbf{I}(X;Y) &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(X=x,Y=y)\log_2\left(\frac{p(X=x)p(Y=y)}{p(X=x,Y=y)}\right) \\ &= -\mathbb{E}(\log_2\left(\frac{p(X)p(Y)}{p(X,Y)}\right)) \geq 0 \end{aligned} \tag{1.5}$$

L'extension au cas vectorielle est immédiate.

Propriétés 4

1. Positivité : $\mathrm{I}(X;Y) \geq 0$
2. Borne sup. : $\mathrm{I}(X;Y) \leq \min(\mathrm{H}(X), \mathrm{H}(Y))$
3. Symétrie : $\mathbf{I}(X;Y) = \mathbf{I}(Y;X)$
4. Information propre : $I(X;X) = \mathbf{H}(X)$
5. Liens avec entropie, entropie conjointe et conditionnelle

$$\begin{aligned} \mathbf{I}(X;Y) &= \mathbf{H}(X) - \mathbf{H}(X|Y) \\ &= \mathbf{H}(Y) - \mathbf{H}(Y|X) \\ &= \mathbf{H}(X) + \mathbf{H}(Y) - \mathbf{H}(X,Y) \end{aligned} \tag{1.6}$$

6. Conditionnement :

$$\mathrm{I}(X;Y \mid Z) \triangleq \mathrm{H}(X \mid Z) - \mathrm{H}(X \mid Y,Z) \geq 0$$

7. Chain rule :

$$\mathbf{I}\left(X_1, X_2, \cdots, X_n; Y\right) = \sum_{i=1}^{n} \mathbf{I}\left(X_i; Y \mid X_{i-1}, \cdots X_1\right)$$

Le lien avec les autres quantité de théorie de l"information est donné par le diagramme dit de Venn suivant

Dans le cas de variables aléatoires, on peut alors donner la défintion suivante

$$I(X;Y) = -\int f(x,y)\log_2\left(\frac{f(x)f(y)}{f(x,y)}\right)dxdy \geq 0$$

1.3 Capacité d'un canal de transmission

1.3.1 Définition

Soit $X \in \mathcal{X}$ le symbole émis et $Y \in \mathcal{Y}$ la sortie observée du canal de transmission supposé sans mémoire caractérisé par la probabilité de transition $P(Y|X)$ alors on à la définition suivante

Définition 9 : *Capacité d'un canal*

La capacité d'un canal sans mémoire est donnée par

$$\begin{aligned} \mathbf{C} &= \max_{p(X)} \mathbf{I}(X;Y) \\ &= \max_{p(X)} \mathbf{H}(X) - \mathbf{H}(X|Y) = \max_{p(X)} \mathbf{H}(Y) - \mathbf{H}(Y|X) \end{aligned} \tag{1.7}$$

Max. atteint pour distribution uniforme pour les canaux *symétriques*.

Cette quantité représente le débit maximum atteignable avec un probabilité arbitrairement faible par un canal de transmission donné par $P(Y|X)$. Il est mesuré en bit par symbole, souvent noté bpcu (bit per channel use).

1.3.2 Exemples de canaux à entrées et sorties discrètes sans mémoire

Canal à effacement

Le canal à effacement (binary erausre channel, BEC) est un exemple simple de canaux sans mémoire à entrée binaire et sortie ternaire. c'est une modélisation simple d'un canal de type "internet" (ecartement des paquet à l'aide de CRC quand intégrité du paquet n'est pas assurée). Ainsi chaque bit X émis $'0'$ ou $'1'$ est soit reçu sans erreur avec une probabilité $1-\epsilon$ ou considéré comme effacé, noté E, avec une probabilité ϵ. On peut alors montrer que la capacité de ce canal se résume à

$$C_{\text{BEC}} = 1 - \epsilon.$$

Canal binaire symétrique

Le canal binaire symétrique à erreurs (binary symetric channel, BSC) est un canal qui modélise les erreurs dans le domaine décidé, et donc sous décision dure au récepteur (hard decision). Dans ce cas, le bit X est reçu coorectement avec un probabilité $1-p$ ou erroné avec une probabilité p, où p est a probabilité d'erreur bit ou "taux de flip". Dnas ce cas la capacité est donnée par

$$C_{\text{BSC}} = 1 - \text{H}_\text{b}(p)$$

1.3.3 Exemples de Canaux à entrées continues et sorties continues

Pour des entrées et sorties à densités continues, l'expression générale de l'information mutuelle est donnée par

$$I(X;Y) = \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy.$$

La capacité est alors obtenue pour un canal donné par maximisation sur la densité de la variable en entrée de canal. Parmi les canaux d'intérêt, on a le canal Gaussien à temps continu qui est un des cas de canaux continus adressé par Shannon à l'origine. Dans ce modèle, le signal observé $y(t)$ est une version bruité du signal envoyé $x(t)$ est un signal de puissance P et à bande limitée $W = 1/T$, où T est le temps symbole. Le modèle continue est alors le suivant

$$y(t) = x(t) + b(t).$$

où $b(t)$ est un processus Gaussien complexe de denité spectrale bilatérale de bruit N_0. On définit alors le rapport signal-sur-bruit par

$$\mathrm{SNR} \triangleq \frac{P}{\mathrm{N_0}W}.$$

On peut alors donner la capacité associée à ce canal par la proposition suiante (preuve admise).

Proposition 1 : *Canal Gaussien à temps continu*

La capacité d'un canal Gaussien à temps continu et bande limitée W est donnée par

$$\mathrm{C_{AWGN-C}} = W \log_2(1 + \mathrm{SNR})[\mathrm{bits/s}]$$

où

$$\mathrm{SNR} \triangleq \frac{P}{\mathrm{N_0}W}.$$

P est la puissance d'émission de X et N_0 est la densité spectrale bilatérale du bruit. La capacité est atteinte pour une distribution Gaussienne de l'entrée X.

Un modèle plus simple à abordé et cependant équivalent sous certaines conditions est le modèle du canal Gaussien à temps discret. Celui-ci correspond à une version discrétisée du canal précédent. Ainsi, on obtient le modèle discret suivant, ie. pour tout instant symbole n, on a

$$y[n] = x[n] + b[n],$$

où $b[n]$ suit une loi normal complexe circulaire, $\mathcal{CN}(0, N_0)$. La loi de transition du canal s'écrit alors

$$p(y|x) \propto e^{-\frac{|y-x|^2}{N_0}}.$$

On peut alors démontrer la proposition suivante.

Proposition 2 : *Canal Gaussien à temps discret*

La capacité d'un canal Gaussien à temps discret real avec une énergie moyenne par symbole $\mathrm{E_s}$ et une variance par dimension $\sigma^2 = \mathrm{No}/2$ est donnée par

$$C_{\mathrm{AWGN}} = \frac{1}{2}\log\left(1 + \frac{E_s}{\sigma^2}\right) \text{ [bits/channel use] or [bits/symbol]},$$

Cette capacité est atteinte pour $X \sim \mathcal{N}(0, \mathrm{E_s})$.
Pareillement, La capacité d'un canal Gaussien complexe à temps discret avec une énergie moyenne par symbole $\mathrm{E_s}$ et une variance par dimension $\sigma^2 = \mathrm{No}/2$ est donnée par

$$C_{AWGN} = \log\left(1 + \frac{E_s}{2\sigma^2}\right) = \log\left(1 + \frac{E_s}{N_0}\right) \text{ [bits/channel use] or [bits/symbol]},$$

La partie "difficile" de la preuve (non détaillée ici) reste de montrer que le maximum est atteint pour une distribution gaussienne. Une fois ceci établi, la preuve fait appel aux expressions simple de l'entropie de variables aléatoire gaussiennes.

En remarquant que $W = 1/T$, on obtient $P = \mathrm{E_s}W$. Le lien entre les deux régime est alors donné par la relation suivante :

$$\mathrm{C_{AWGN-C}} = W * C_{AWGN}.$$

On remarque alors que $\mathrm{C_{AWGN-C}}$ est homogène à un débit alors que $\mathrm{C_{AWGN}}$ est homogène à une efficaité spectrale en $bit/s/Hz$.

Pour les E_s/N_0 faibles (régime limité en puissance), on a

$$\begin{aligned}\mathrm{C_{AWGN}} &= \log_2(1 + E_s/N_0) \\ &\approx \frac{1}{\ln 2} \cdot E_s/N_0.\end{aligned}$$

On a donc un régime linéaire. A contrario, pour les forts E_s/N_0, on a un régime logarithmique, ie.

$$\mathrm{C_{AWGN}} \approx \log_2(E_s/N_0)$$

1.3.4 Canaux à entrées discrètes et sorties continues

Capacité à entrées contraintes

On considère maintenant des canaux à entrées discrètes. Ceci représente le cas d'usage le plus fréquent en communications numériques où les syboles émis appartiennent à des constellations à des alphabets finis comme les modulations de type PAM, QAM ou encore PSK/APSK. Dans ce cas, l'information mutuelle s'écrit comme suit

$$\mathrm{I}(X;Y) = \sum_{x\in\mathcal{X}} \int p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} dy,$$

avec $p(x,y) = p(y|x)p(x)$ et $p(y) = \sum_x p(y|x)p(x)$ La capacité associée à ce type d'entrées en souvent appelée *capacité à entrées contraintes (constrained input capacity)*. Le maximum de l'informatio mutuelle pour une capacité à entrées contraintes est obtenu pour une distribution uniforme des symboles d'entrée. Si $|\mathcal{X}| = M$, on aura $p(x) = 1/M, \forall x \in \mathcal{X}$, et on a donc l'expression suivante

$$\mathbb{C} = \mathrm{I}(X;Y) = \frac{1}{M} \sum_{x\in\mathcal{X}} \int p(y|x) \log_2 \frac{p(y|x)}{\frac{1}{M}\sum_x p(y|x)} dy. \tag{1.8}$$

Calcul de la capacité

Contrairement, au cas précédent, même pour un canal de type AWGN, il n'existe pas d'expression analytique plus simple que cette formulation intégrale. Il faut donc pour un canal particulier calculer l'expression intégrale par intégration numérique ou de Monte-Carlo. Pour calculer efficacement cette capacité, item Calculer la capacité revient à évaluer $\mathbf{C} = \mathbf{I}(X;Y) = \mathbf{H}(X) - \mathbf{H}(X|Y)$ au travers des Termes $\mathbf{H}(X)$ ou $\mathbf{H}(X|Y)$. Voilà alors les différentes étaoes pour calculer la capacité

- Calcul de $\mathbf{H}(X)$: En considérant que le maximum est atteint pour une distribution uniforme des symboles d'entrée, on obtient facilement que

$$\mathbf{H}(X) = \log_2(M).$$

- Calcul de $\mathbf{H}(X|Y)$:

$$\mathbf{H}(X|Y) = -\mathbb{E}(\log_2 p(X|Y)) = \mathbb{E}(h(X|Y))$$

On doit donc calculer le terme $h(X|Y) = -\log_2(p(X|Y))$. Ceci est aisément obtenu en considérant la vraisemblance du canal de la manière suivante

$$p(X|Y) = \frac{p(Y|X)}{\sum_{x\in\mathcal{X}} p(Y|X=x)}.$$

Le calcul de $\mathbf{H}(X|Y)$ peut alors se faire par Monte-Carlo en utilisant un estimateur asymptotiquement sans biais de cette quantité. En effet, par ergodicité, on a

$$\mathbf{H}(X|Y) = \mathbb{E}(h(X|Y)) = \lim_{N\to+\infty} \frac{1}{N}\sum_n h(x(n)|y(n))$$

où $h(x(n)|y(n)) = \log_2(p(x(n)|y(n))))$.
La procédure globale pour l'estimation par monte-Carlo peut alors se résumé comme suit

1. Tirer aléatoirement et uniformément des symboles issue de constellation,
2. Calculer pour chaque couple $(x(n), y(n))$, $h(x(n)|y(n))$ et moyenner.

1.4 Théorème du codage canal

Théorème 1 : *Codage canal*

Pour un canal à temps discret, il est possible de transmettre de l'information avec une probabilité d'erreur arbitrairement faible si le débit de communication R est inférieur à la capacité du canal, i.e. $R < C$. Plus précisément, pour tout $R < C$, il existe une séquence de schémas de codage de longueur n de probabilité d'erreur moyenne $P_e^(n)$ tendant vers zéro quand $N \longmapsto +\infty$
Inversement, toute séquence de schémas de codage avec une probabilité d'erreur tendant veers zéro doit vérifier $R < C$. Si $R > C$, la probabilité d'erreur est forcément non nulle.

La capacité représente donc la quantité d'informations qui peut être transmise de manière *fiable* à travers un canal par utilisation de canal. Pour $R > C$, aucune communications fiable, sans erreur n'est possible.

Dans le cas d'un canal AWGN, on aura quelques remarques intéressantes à faire sur cette relation. Par le théorème précédent, on aura

$$\begin{aligned} R &< C_{AWGN} && (1.9)\\ &= \log_2(1+\tfrac{E_s}{2\sigma^2}) && (1.10)\\ &= \log_2(1+\tfrac{E_s}{N_0}) && (1.11)\\ &= \log_2(1+R\tfrac{E_b}{N_0}) && (1.12)\end{aligned}$$

où E_b est l'énergie par bit utile/d'information. On a alors

$$\frac{E_b}{N_0} > \frac{2^R-1}{R} \underset{R\to 0}{\longrightarrow} \ln(2) = -1.59dB$$

Chapitre 2

Codage de canal - critère de décodage

Nous rappelons ici quelques notions de bases relatives au codage de canal.

2.1 Codes en blocs linéaires

2.1.1 Définition

On considèrera des codes de canal définis sur $\mathbb{F}_2 = GF(2)$.

Définition 10 : *Codes linéaires en blocs*

Un code en bloc binaire $\mathcal{C}(N, K)$ de *longueur* N et de *dimension* K est une application $g(.)$ de l'ensemble $\mathbb{F}_2^K = \{0,1\}^K$ vers l'ensemble $\mathbb{F}_2^N = \{0,1\}^N$ qui associe à tout bloc de données d'information $\mathbf{u}$ un mot de code $\mathbf{c}$.

$$\begin{array}{rccc} g : & \mathbb{F}_2^K & \rightarrow & \mathbb{F}_2^N \\ & \mathbf{u} & \mapsto & \mathbf{c} = g(\mathbf{u}) \end{array} \tag{2.1}$$

$\mathcal{C}(N, K)$ est dît linéaire si $g(.)$ est une application linéaire.

Pour un code linéaire, le nombre de mots de code est 2^K et forme un sous-espace vectoriel de $\mathbb{F}_2^N$. De plus, toute combinaison linéaire de mots de code est un mot de code de $\mathcal{C}(N, K)$. On définit le rendement de codage comme le rapport entre le nombre K de symboles d'information divisé par le nombre N de symboles codés :

$$R = \frac{K}{N}$$

On adoptera les notations vectorielles suivantes $\mathbf{c} = [c_0, \ldots, c_{N-1}]$ et $\mathbf{u} = [u_0, \ldots, u_{K-1}]$.

2.1.2 Matrice génératrice

Définition 11 : *Matrice génératrice*

La matrice génératrice $\mathbf{G}$ de dimensions $K \times N$ est la matrice associée à l'application linaire g définie comme

$$\mathbf{c} = \mathbf{uG}.$$

L'espace image du code est alors défini par $\mathrm{Im}(\mathcal{C}) = \{\mathbf{c} \in \mathbb{F}_2^N | \mathbf{c} = \mathbf{uG}, \ \ \forall \mathbf{u} \in \mathbb{F}_2^K\}$

On a alors les propriétés suivantes

Propriétés 5 : *Propriétés de la matrice génératrice*

1. $\text{rang}(G) = K$,
2. les lignes de $\mathbf{G}$ sont K mots de codes indépendants,
3. $\mathbf{G}$ n'est pas définie de manière unique (pourquoi ?),
4. $\mathbf{G}$ est dite systématique si $\forall k \in [0, K-1], \exists n \in [0, N-1]$ tel que $c[n] = u[k]$. $\mathbf{G}$ peut alors se mettre sous la forme
$$\mathbf{G} = [P|I_K]$$
5. Pour chaque coordonnée d'un mot de code de $\mathcal{C}(N, K)$, la probabilité d'occurrence d'un $'1'$ ou d'un $'0'$ est identique si la génération de l'information est i.i.d..

2.1.3 Matrice de parité

Définition 12 : *Matrice de parité*

Le code $\mathcal{C}^{\perp}(N-K, K)$, dît code dual, vérifie que tout mot du code dual est orthogonal à tout mot du code $\mathcal{C}(N, K)$. On note une matrice génératrice pour ce code $\mathbf{H}$. On a alors
$$\{\mathbf{c} \in \mathcal{C}(N, K) | \mathbf{c}\mathbf{H}^{\top} = \mathbf{0}\}$$

Ainsi, on a tous les bits d'un mot de code $\mathbf{c}$ vérifie un jeu de contraintes linéaires appelées *équations de parité*.

Propriétés 6

— Relation avec $\mathbf{G}$: $\mathbf{G}\mathbf{H}^{\top} = \mathbf{0}$
— Pour un code systématique défini par $\mathbf{G} = [P|I_K]$, on a
$$\mathbf{H} = [I_{N-K}|P^{\top}].$$

On peut également utiliser la matrice de parité pour la détection d'erreur à l'aide du **syndrome**. Soit $\mathbf{r} = \mathbf{c} + \mathbf{e}$ où $\mathbf{r}$ est le mot reçu et $\mathbf{e}$ le vecteur de bruit. Alors

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\top} = \mathbf{e}\mathbf{H}^{\top}$$

Si $\mathbf{e}$ n'est pas un mot de code alors, une erreur est détecter car le syndrome est non nul dans ce cas. Si $\mathbf{e}$ est un mot de code, alors on parle d'erreurs *non détectable* (syndrome nul).

2.1.4 Distance minimum et spectre de distance

Définition 13 : *Poids de Hamming d'un vecteur*

On appelle poids de Hamming d'un mot de code $\mathbf{c} \in \mathbb{F}_2^n$ le nombre de coordonnées non nulles de ce vecteur. On notera

$$w(\mathbf{c}) = \sum_{k=0}^{n-1} \delta(c_k), \text{ où} \delta(c_k) = \begin{cases} 1 & \text{si } c_k = 1 \\ 0 & \text{sinon} \end{cases}$$

Définition 14 : *Distance de Hamming entre deux vecteurs*

La distance de Hamming entre deux mots de code est définie par

$$d_H(c_i, c_j) = w(c_i \oplus c_j)$$

Définition 15 : *Distance minimale*

La distance minimale du code $\mathcal{C}$ est donnée par

$$\begin{aligned} d_{\min} &= \min \{d_H(c_i, c_j) | c_i, c_j \in \mathcal{C}(N, K); c_i \neq c_j\} \\ &= \min \{\boldsymbol{w}(c) | c \in \mathcal{C}(N, K), c \neq 0\} \end{aligned} \tag{2.2}$$

Définition 16 : *Enumérateur/spectre de poids*

On définit alors le *spectre de poids* d'un code par les coefficients

$$\forall i = 1 \dots N, w_i = \#\mathbf{c} \in \mathcal{C}(N, K), \ \boldsymbol{w}(c) = i$$

Le set $\{w_0, \cdots, w_N\}$ est alors appelé la distribution des poids du code que l'on représente par le polynôme énumérateur de poids

$$W(x) = \sum_{i=0}^{N} w_i x^i$$

On peut alors remarquer que $d_{\min}$ est égale au plus petit nombre de colonnes dont la somme est le vecteur nul.

2.1.5 Exemples

Un code de répétition consiste en la répétition de N fois un bit d'information. On obtient un code $\mathcal{C}(N, 1)$ de matrice génératrice

$$G_1 = [\underbrace{1 \dots 1 \dots 1}_{N}]$$

Un code de vérification de parité est construit tel que $c_{N-1} = u_0 \oplus u_1 \oplus \ldots \oplus u_{N-2}$ définissant un code $\mathcal{C}(N, N-1)$. Sa matrice génératrice est alors donnée par

$$G_2 = \begin{pmatrix} & & 1 \\ & & \vdots \\ & I_{N-1} & 1 \\ & & \vdots \\ & & 1 \end{pmatrix}$$

On peut alors remarquer que les deux codes précédents sont duaux, ie.

$$G_1 G_2^\top = \mathbf{0}$$

2.2 Critères de décodage séquence

2.2.1 Décodage MAP/ML

Définition 17 : *Décodage par Maximum a Posteriori (MAP) (Séquence)*

$$\begin{aligned} \hat{\mathbf{c}} &= \arg\max_{\mathbf{c}'} p(\mathbf{c}'|\mathbf{y}) \\ &= \arg\max_{\mathbf{c}'} \frac{p(\mathbf{y}|\mathbf{c}')p(\mathbf{c}')}{p(\mathbf{y})} \\ &= \arg\max_{\mathbf{c}'} p(\mathbf{y}|\mathbf{c}')p(\mathbf{c}') \end{aligned} \tag{2.3}$$

Définition 18 : *Décodage par Maximum de Vraisemblance (ML) (séquence)*

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}'} p(\mathbf{y}|\mathbf{c}') \tag{2.4}$$

Dans le cas où l'on considère l'ensemble des mots de codes équiprobable ($p(\mathbf{c}) = 1/2^K$), les critères MAP et ML séquence sont équivalents. En pratique, le critère minimisé est équivalent à la minimisant d'un taux d'erreur trame (Frame Error Rate, FER) et non à la minimisaton d'un taux d'erreur binaire (Bit Error Rate, BER).

2.2.2 Exemples de canaux.

On peut alors spécifier ces critères en fonction du canal considéré. Deux types de canaux rencontrés régulièrement sont le canal à erreur binaire symétrique (BSC) et le canal à bruit blanc additif Gaussien (AWGN).

Canal binaire symétrique (BSC) :

Pour ce canal représenté figure 2.1, le mot reçu est modélisé par le signal émis et une erreur binaire :

$$\forall n = 0, \cdots N-1, y(n) = c(n) \oplus e(n)$$

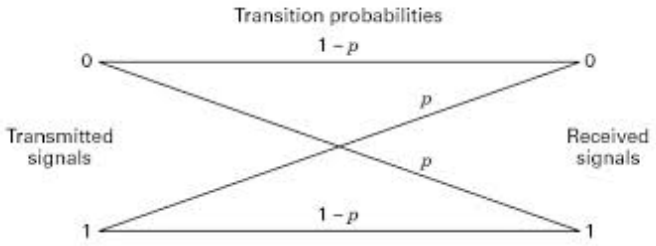


FIGURE 2.1 – Représentation d'un canal de type BSC

où $e(n) \in \{0,1\}$ sont des v.a. i.i.d. binaires telles que $p(e(n)=1)=p$, indépendantes de $c(n)$. $\oplus$ est l'addition modulo-2 (équivalent à l'opération OU-Exclusif/XOR).

Le canal étant sans mémoire, on a donc

$$p(\mathbf{y}|\mathbf{c}) = p(\mathbf{e}) = \prod_{n}^{N-1} p(e(n)) = \prod_{n}^{N-1} p(y(n)|c(n))$$

ce qui équivaux à

$$p(\mathbf{y}|\mathbf{c})) = (1-p)^{N-d_H(\mathbf{y},\mathbf{c})} p^{d_H(\mathbf{y},\mathbf{c})}$$

soit

$$\begin{aligned} \log\left(p(\mathbf{y}|\mathbf{c})\right)) &= N\log(1-p) - d_H(\mathbf{y},\mathbf{c})\log(1-p) + d_H(\mathbf{y},\mathbf{c})\log(p) \\ &= N\log(1-p) + d_H(\mathbf{y},\mathbf{c})\log\left(\tfrac{p}{(1-p)}\right) \end{aligned} \tag{2.5}$$

comme $p \leq 0.5$, on a $\log\left(p/(1-p)\right) < 0$, d'où l résultat suivant :

$$\begin{aligned} \mathbf{c} &= \arg\max_{\mathbf{c}'} p(\mathbf{y}|\mathbf{c}') \\ &= \arg\max_{\mathbf{c}'} \log\left(p(\mathbf{y}|\mathbf{c}')\right) \\ &= \arg\min_{\mathbf{c}'} d_H(\mathbf{y},\mathbf{c}') \end{aligned} \tag{2.6}$$

Sur canal binaire symétrique (détection à décisions "dures" (hard decoding/decision)), le mot de code le plus probable est donc celui qui minimise la distance au sens de Hamming avec le mot reçu.

Canal additif blanc Gaussien à entrées binaires/antipodales :

Sur canal Gaussien, le modèle échantillonné est donné par

$$\forall n = 0 \cdots N-1, y(n) = x(n) + b(n)$$

où $x(n) \in \{+1,-1\}$ est la version modulée en BPSK des bits codés $c(n) \in \{0,1\}$ tels que $x(n) = \mathcal{M}(c(n)) = 1 - 2c(n) = (-1)^{c(n)}$, soit le mapping $\{'0' \leftrightarrow' +1','1' \leftrightarrow' -1'\}$. $b(n)_{n=0\cdots N-1}$ est un processus aléatoire Gaussien où $b(n)$ sont des v.a. Gaussiennes réelles i.i.d de densité de probabilité $\mathcal{N}(0,\sigma_b^2)$.

Le canal étant sans mémoire, on a donc

$$p(\mathbf{y}|\mathbf{c}) = p(\mathbf{y}|\mathbf{x}) = \prod_{n}^{N-1} p(y(n)|x(n))$$

ce qui équivaut à

$$p(\mathbf{y}|\mathbf{x}) \propto \prod_n e^{-\frac{(y(n)-x(n))^2}{2\sigma_b^2}}$$

En prenant le log, on obtient alors facilement,

$$\begin{aligned} \mathbf{c} &= \arg\max_{\mathbf{c}'} p(\mathbf{y}|\mathbf{c}') \\ &= \arg\max_{\mathbf{c}'} \log\left(p(\mathbf{y}|\mathbf{c}')\right) \\ &= \arg\min_{\mathbf{c}'=\mathcal{M}^{-1}(\mathbf{x}'(\mathbf{n}))} \sum_n (y_n - x'_n)^2 \end{aligned} \tag{2.7}$$

où $d_E(\mathbf{y}, \mathbf{c}') = \sum_n (y_n - c'_n)^2$ représente la distance Euclidienne entre les deux séquences de bits codés émises. Donc sur canal AWGN, le mot de code le plus probable est celui qui est le plus proche de la séquence reçue au sens de la distance euclidienne.

On peut aller plus loin sur la simplification de l'expression (2.16). En effet, en développant on obtient

$$\sum_n (y_n - x_n)^2 = \sum_n y_n^2 - 2\sum_n y_n x_n + \sum_n x_n^2$$

Comme le premier et troisième ($x_n^2 = 1$) termes du membre de droite sont les mêmes pour tout mot de code, en prenant le log, on obtient alors facilement,

$$\mathbf{c} = \arg\max_{\mathbf{c}'=\mathcal{M}^{-1}(\mathbf{x}'(\mathbf{n}))} \sum_n y_n x'_n \tag{2.8}$$

On interprète ici comme la recherche de l'intercorrélation maximale entre le signal reçu et le signal émis. On peut encore le réécrire en fonction de $c(n)$ comme suit

$$\begin{aligned} \mathbf{c} &= \arg\min_{\mathbf{c}'} \sum_n y_n c'_n \\ &= \arg\min_{\mathbf{c}'} \sum_{n|c'_n=1} y_n \end{aligned} \tag{2.9}$$

On verra par la suite que l'on peut avoir également une interprétation dans le cadre d'un décodage "souple".

2.3 Notions d'information souple

2.3.1 Définition et propriétés

On définit la quantité suivante le log-rapport de probabilité ou log-likelihood (LLR) la quantité donnée par la définition suivante :

Définition 19 : *Détection souple d'un bit*

$$L(c_n) = \log\left[\frac{p(c_n = 0|y_n)}{p(c_n = 1|y_n)}\right] = \log\left[\frac{p(c_n = 0; y_n)}{p(c_n = 1; y_n)}\right] \tag{2.10}$$

Cette quantité est homogène à une information a posteriori. A partir de la relation précédente, on peut donner la relation suivante par simple application de la règle de Bayes

$$L(c_n) = \log\left[\frac{p(y_n|c_n=0)}{p(y_n|c_n=1)}\right] + \log\left[\frac{p(c_n=0)}{p(c_n=1)}\right] = L_c(y_n) + L_a(c_n)$$

$L_c(y_n)$ est ce que l'on le log-rapport de vraisemblance qui est associé à la règle de décision ML dans le cas d'absence d'information a priori sur le bit c_n où dans le cas d'une source binaire i.i.d.. Le test est alors donné par

$$L_c(y_n) \underset{\mathcal{H}_1 : c_n = 1}{\overset{\mathcal{H}_0 : c_n = 0}{\gtrless}} 0$$

$|L_c(y_n)|$ représente la fiabilité de la décision associée au signe du LLR, qui donne lui la décision dure. $L_a(c_n)$ est le LLR a priori associé au bit c_n. Le test

$$L(c_n) \underset{\mathcal{H}_1 c_n = 1}{\overset{\mathcal{H}_0 : c_n = 0}{\gtrless}} 0$$

correspond donc au test MAP. La détection souple associée est donc donnée par :

Définition 20 : *Détection souple d'un bit (MAP bit)*

$$\begin{aligned} \hat{c}_n &= \arg\max_{c_n} p(c_n|y_n) \\ &= \frac{1 - \text{signe}(L(c_n))}{2} \end{aligned} \tag{2.11}$$

On a les mêmes interprétations que pour $L_c(y_n)$ quant au module et au signe.

Exemple 3 : *Cas du canal AWGN*

Dans le cas du canal AWGN, on a

$$y_n = x_n + b_n$$

où $x_n = 1 - 2c_n$ et b_n un processus Gaussien centré de variance σ_b^2. Un calcul immédiat permet de d'avoir pour le cas d'une source binaire uniforme

$$L(c_n) = L_c(y_n) = \frac{2}{\sigma^2} y_n$$

L'application de la règle de décision ML/MAP dans ce cas donne

$$y_n \underset{\mathcal{H}_1 c_n = 1}{\overset{\mathcal{H}_0 : c_n = 0}{\gtrless}} 0 \ ,$$

ce qui correspond à la règle de décision dure classique sur une modulation de type BPSK. On remarque que le LLR est une version pondérée de l'observation. Le facteur de pondération est proportionnel au rapport signal à bruit (exprimé en échelle linéaire). L'interprétation est la suivante : plus ce dernier est élevé, plus on donne de l'importance à ce que l'on observe, mais quand ce dernier tend vers 0, on donne peu de crédit à cette dernière.
En développant l'expression du LLR qui est une fonction de l'observation, on en déduit que, conditionnellement au bit émis, ce dernier est une variable aléatoire telle que

$$L(c_n) \sim \mathcal{N}(m, 2.|m|)$$

où

$$m = \frac{2}{\sigma_b^2} x_n$$

Exemple 4 : *Cas du canal de Rayleigh*

Dans le cas du canal de Rayleigh parfaitement entrelacé et connaissance parfaite du canal, on a le modèle suivant

$$y_n = \alpha_n x_n + b_n$$

où $x_n = 1 - 2c_n$, b_n un processus Gaussien centré de variance σ_b^2 et $\alpha \sim \mathcal{R}(1/\sqrt{(2)})$ (loi de Rayleigh normalisé, ie. $\mathbb{E}(\alpha^2) = 1$). Un calcul immédiat permet de d'avoir pour le cas d'une source binaire uniforme

$$L(c_n) = \log\left[\frac{p(y_n|c_n = 0; \alpha_n)}{p(y_n|c_n = 1; \alpha_n)}\right] = \frac{2}{\sigma^2} \alpha_n y_n$$

On peut également noter que la connaissance du LLR permet de remonter facilement aux probabilité en remarquant que

$$p(u_n = 0|y_n) + p(u_n = 1|y_n) = 1 \tag{2.12}$$

$$L(u_n) = \log\left(\frac{p(u_n = 0|y_n)}{p(u_n = 1|y_n)}\right) \tag{2.13}$$

On obtient alors les relations suivantes

$$p(u_n = 0|y_n) = \frac{e^{L(u_n)}}{1 + e^{L(u_n)}} \tag{2.14}$$

$$p(u_n = 1|y_n) = \frac{1}{1 + e^{L(u_n)}} \tag{2.15}$$

Ces expressions nous donnent alors le terme générique de cette probabilité a postériori donnée en fonction de u_n par

$$p(u_n|y_n) = \frac{e^{(1-u_n)L(u_n)}}{1 + e^{L(u_n)}}$$

L'expression en fonction de $x_n = 1 - 2.u_n$ s'obtient par multiplication au numérateur et dénominateur par $e^{-\frac{L(u_n)}{2}}$

$$p(u_n|y_n) = \frac{e^{x_n \frac{L(u_n)}{2}}}{e^{-\frac{L(u_n)}{2}} + e^{+\frac{L(u_n)}{2}}}$$

2.3.2 Interprétation du point de vue de la théorie de l'estimation

La quantité $L(u_n)$ est souvent appelée *information souple*. En utilisant les expressions précédentes, elle peut être reliée à la quantité $\hat{x}_n = \mathbb{E}(X_n|Y_n = y_n)$, quelquefois dénommée *soft bits*, de la façon suivante :

$$\hat{x_n} = \mathbb{E}(X_n|Y_n = y_n) = \tanh\left(\frac{L(u_n)}{2}\right)$$

On peut montrer que $\hat{x}_n$ est la meilleure estimée non linaire au sens du MMSE, ie.

$$\hat{x}(n) = \arg\min_{x \in \mathbb{R}} \mathbb{E}(|x(n) - \hat{x}(n)|^2)$$

2.3.3 Interprétation dans le domaine de Fourier

La transformée de Fourier d'une fonction f définie sur $\mathbb{Z}_2$ munie des opérations usuelles (addition, multiplication) définies sur le corps binaire est donnée par

$$\hat{\mathbf{f}} = \mathbf{F}\mathbf{f}$$

$$\begin{pmatrix} \hat{f}[0] \\ \hat{f}[1] \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} f[0] \\ f[1] \end{pmatrix}$$

L'application de cette transformation de Fourier à la fonction de Z_2 dans $\mathbb{R}$ nous permet d'écrire

$$\begin{pmatrix} \hat{f}[0] \\ \hat{f}[1] \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} p(u_n = 0|y_n) \\ p(u_n = 1|y_n) \end{pmatrix},$$

ce qui donne

$$\begin{pmatrix} \hat{f}[0] \\ \hat{f}[1] \end{pmatrix} = \begin{pmatrix} 1 \\ \tanh\left(\frac{L(u_n)}{2}\right) \end{pmatrix}.$$

On en déduit que l'estimée $\hat{x_n}$ se base sur un critère fréquentiel.

2.3.4 Estimation efficace de la capacité des canaux binaires

Si on considère le modèle discret équivalent suivant

$$y[n] = x[n] + b[n],$$

où $x[n] \in \{-1, +1\}$ et $b[n]$ suit une loi Gaussienne centrée de variance σ_b^2; la formule de la capacité gaussienne pour une entrée contrainte nous donne

$$I(X;Y) = \frac{1}{2} \sum_{x=\pm 1} \int_{\mathbb{R}} f(y \mid x) \log_2 \left(\frac{2f(y \mid x)}{f(y \mid x=+1) + f(y \mid x=-1)} \right) dy.$$

En utilisant le fait que

$$I(X;Y) = H(X) - H(X \mid Y),$$

on peut écrire

$$I(X;Y) = 1 + \mathbb{E}_{X,Y} \left(\log_2(p(X \mid Y)) \right)$$

En introduisant l'expression du LLR (dont la dépendance explicite à l'observation est appuyée par la notation choisie dans cette section)

$$L(y) = \log \left(\frac{p(x=+1 \mid Y=y)}{p(x=-1 \mid Y=y)} \right),$$

en utilisant le fait que $p(x=+1 \mid y) + p(x=-1 \mid y) = 1, \ \forall y$, il vient

$$\begin{aligned} p(x=+1 \mid y) &= \frac{e^{L(y)}}{1+e^{L(y)}} \\ &= \frac{1}{1+e^{-L(y)}} \\ p(x=-1 \mid y) &= \frac{1}{1+e^{L(y)}} \end{aligned}$$

Ce qui permet d'écrire, l'expression générique en x

$$p(x \mid y) = \frac{1}{1+e^{-xL(y)}}.$$

Par intégration de Monte-Carlo, on obtient alors un estimateur empirique non biaisé simple donné par

$$\hat{I}(X;Y) = 1 + \frac{1}{N} \sum_{n=0}^{N-1} \log_2(p(x[n] \mid y[n])) = 1 - \frac{1}{N} \sum_{n=0}^{N-1} \log_2 \left(1 + e^{-x[n]L(y|n])} \right).$$

2.3.5 Estimation ML séquence basée sur les LLR.

On également redériver le critère ML en fonction du LLR En supposant, le canal sans mémoire et en repartant de l'expression

$$p(\mathbf{y}|\mathbf{c}) = p(\mathbf{y}|\mathbf{x}) = \prod_{n}^{N-1} p(y(n)|x(n))$$

et en utilisant le fait que

$$L(c_n) = \log \left[\frac{p(y_n|c_n=0)}{p(y_n|c_n=1)} \right],$$

on peut alors par changement de variable redonner l'expression du critère ML de la façon suivante :

$$\begin{aligned}
\mathbf{c} &= \arg\max_{\mathbf{c}'} p(\mathbf{y}|\mathbf{c}') \\
&= \arg\max_{\mathbf{c}'=\mathcal{M}^{-1}(\mathbf{x})} \prod_{n}^{N-1} \frac{e^{x_n \frac{L(c'_n)}{2}}}{e^{-\frac{L(c'_n)}{2}} + e^{+\frac{L(c'_n)}{2}}} \\
&= \arg\max_{\mathbf{c}'=\mathcal{M}^{-1}(\mathbf{x})} \sum_{n} x_n L(c'_n) \\
&= \arg\max_{\mathbf{c}'=\mathcal{M}^{-1}(\mathbf{x})} \sum_{n} (1-2c_n) L(c'_n) \qquad (2.16)
\end{aligned}$$

On remarque alors que pour chaque mot de code on réalise une *corrélation* entre le mot reçu (ou de manière équivalent la séquence de LLR associée) et un mot de code valide. Ainsi, la dernière expression permet de donner la structure brute d'un décodeur ML : on peut l'assimiler à un banc de corrélateurs dont chacune des 2^K branche est la corrélation avec un mot de code possible. Pour la décision il suffit alors de sélectionner la branche dont la sortie est la plus grande. D'un point de vue complexité, l'approche brute force est de $2^K N = 2^{R\,N} N$ multiplications et $2^{R\,N}(N-1)$ additions. On a donc une complexité exponentielle en la taille du mot de code. Pour chaque famille de code, on cherchera à utiiser la structure propre du code pour "casser" cette complexité.

On discute maintenant de quelques cas simples.

Exemple 5 : *Code de répétition*

Pour un code de répétition de taille N, de rendement $R = 1/N$, on a deux mots de codes le mot de codes $\mathbf{c_0} = [0 \cdots 0]$ et $\mathbf{c_1} = [1 \cdots 1]$, tous les deux dans $\mathbb{F}_2^N$. Sélectionner le mot de code le plus probable revient à réaliser le test suivant

$$\sum_{n=0}^{N-1} L(c_n) \underset{\mathcal{H}_1 : \mathbf{c_1}}{\overset{\mathcal{H}_0 : \mathbf{c_0}}{\gtrless}} 0$$

Ce test permet également de déterminer le bit d'information émis :

$$\sum_{n=0}^{N-1} L(c_n) \underset{\mathcal{H}_1 : u = 1}{\overset{\mathcal{H}_0 : u = 0}{\gtrless}} 0$$

Exemple 6 : *Code de parité*

Pour un code de répétition de taille N, de rendement $R = (N-1)/N$, on a 2^{N-1} mots de codes, tous dans $\mathbb{F}_2^N$. Si on prend une approche "brute-force" non structurée, ie. purement énumérative, sélectionner le mot de code le plus probable revient à réaliser une "banc de filtre" ou "de corrélations" du vecteur des LLRs reçus avec l'ensemble des mots de codes possibles. Cela revient à un test à 2^{N-1} hypothèses . On voit bien que sans prendre en compte la structure de ce code, l'énumération devient prohibitive quand N croit.

Exemple 7 : *Code de Hamming*

Un code de Hamming peut être simplement défini par une matrice de parité dont les colonnes sont tous les M-uplets de $\mathbb{F}_2^M$. On obtient des codes de type

2.3.6 Représentation en treillis des codes en bloc

Pour pouvoir décoder efficacement un code en bloc au sens du maximum de vraisemblance sans passer par une énumération à complexité prohibitive, on peut dériver une représentation graphique appelée *treillis* qui permet de représenter efficacement la structure sous-jacente des mots de codes. Plusieurs types de treillis peuvent être définis pour un code en bloc, parmi les plus utilisé, on peut citer le treillis associé à la matrice de parité. En partant de l'équation $\mathbf{c}\mathbf{H}^{\mathrm{T}} = \mathbf{0}$, où $\mathbf{H}$ est la matrice de parité de taille $M \times N$. En notant $\mathbf{h}_j$ la j-ième colonne de $\mathbf{H}$, on obtient l'équation vectorielle

$$c_1\mathbf{h}_1 + c_2\mathbf{h}_2 + \cdots + c_n\mathbf{h}_n = \mathbf{0}.$$

On peut alors définir une représentation dans l'"espace des états" où l'état S_ℓ correspond à la somme partielle des colonnes $\mathbf{h}_j,\ j = 1 \cdots \ell$ défini comme suit

$$S_\ell = c_1\mathbf{h}_1 + c_2\mathbf{h}_2 + \cdots + c_\ell\mathbf{h}_\ell$$

Par convention, on a $S_0 = S_N = \mathbf{0}$, ie. on commence par un vecteur 'somme partielle' nul et par définition de l'équation de parité, la somme totale des colonnes est nulle. De plus, comme $\mathbf{h}_j$ sont des vecteurs appartenant à $\mathbb{Z}_2^M$, le nombre d'états maximum est de 2^m pour chaque somme partielle $\ell = 1, 2, \ldots, N-1$. Cette description du code dans l'espace des états S_ℓ donne une représentation graphique de type treillis dont la structure permet d'énumérer efficacement les mots de code, chaque chemin représentant un mot de code possible. Contrairement au cas des codes convolutifs, comme nous le reverrons dans le chapitre suivant, ce treillis varient dans le temps. Son application dépend également du rendement du code. En effet la complexité pour le décodage ML sera de l'ordre de $\mathcal{O}(2^M)$. Ainsi cette méthode sera intéressante pour des rendements élevés (M "petit"). Quand le rendement diminue, on préferra une représentation par un treillis donné par la matrice génératrice ou en utilisant le code dual.

2.4 Décodage MAP bit/symbole pour un code en bloc

On va maintenant s'intéresser un critère MAP symbole, le symbole étant ici un bit. Ce critère diffère du MAP séquence ou ML. L'idée ici est de calculer la probabilité a posteriori d'un bit d'information (ou codé comme on le verra après) sachant l'ensemble de observations. Ceci est donné par la définition suivante si on considère une séquence codée $c_n \in \{0, 1\}$ de taille N

Définition 21 : *Critère MAP bit*

$$\begin{aligned} \hat{c}_n &= \arg\max_{c_n} p(c_n|\mathbf{y}) \\ &= \frac{1-\text{signe}(L(c_n))}{2} \end{aligned} \tag{2.17}$$

avec

$$L(c_n) = \log\left[\frac{p(c_n=0|\mathbf{y})}{p(c_n=1|\mathbf{y})}\right]$$

La quantité $L(c_n)$ est ce que l'on appelle le LLR (*Log-Likelihood Ratio*) a posteriori associé au bit c_n.

Le critère est une conséquence du fait que l'on considère une variable binaire. En effet, si on considère le bit c_n, alors le test de décision statistique binaire associé donne

$$\hat{c}_n = \arg\max_{c_n} p(c_n|\mathbf{y}) = 0$$

si

$$p(c_n=0|\mathbf{y}) > p(c_n=1|\mathbf{y}).$$

Par passage au log et définition du LLR associé à u_n on obtient alors $L(c_n) > 0$. De la même manière, on aura $\hat{c}_n = 1$ si $L(c_n) < 0$. On a donc la relation $\text{signe}(L(c_n)) = 1 - 2u_n$ qui nous définit un mapping bijectif entre le signe du LLR et la valeur binaire. Ce mapping bijectif correspond d'ailleurs à un mapping de type BPSK : $\{`0' \leftrightarrow +1, `1' \leftrightarrow -1\}$. Compte tenu de cette relation directe, il est souvent plus commode de directement considérer la version "mappée/modulé" du bit (cas des canaux souples), de sorte que l'on considérera de manière indifférenciée $c_n = 0/1$ et $c_n = \pm 1$ dès lors le contexte en suffisamment clair. Ainsi, on pourra adopter la règle MAP suivante

$$\begin{aligned} \hat{c}_n &= \arg\max_{c_n} p(c_n|\mathbf{y}) \\ &= \text{signe}(L(c_n)) \end{aligned} \tag{2.18}$$

Le LLR associé à c_n peut être associé à une mesure de fiabilité de la décision. En effet, $\text{signe}(L(u_n))$ est identifiable à une décision "dure" du bit à laquelle on peut associer une mesure de fiabilité de cette décision grâce à $|L(u_n)|$. Un algorithme qui fournit en sortie des informations homogènes à des LLRs (ou symboles bruités) est appelé *algorithme à sorties souples (Soft Outputs, SO)* par opposition à des algorithmes qui fournissent une version décidée des symboles dénommés *algorithme à sorties dures (Hard Outputs, HO)*. Par analogie, si les entrées sont des symboles décidés, on parle d'algorithme à entrées dures (Hard Input, HI) par opposition à des algorithmes à entrées souples (Soft Inputs, SI) prenant en compte des symboles bruités ou des entrées représentant des statistiques suffisantes. Comme pour le cas du décodage ML, la mise en oeuvre d'un algorithme de décodage efficace dépend de la structure du code sous-jacent. Nous explorerons différents types de détecteur symbole MAP : les algorithme de type BCJR qui acte sur des treillis et et qui seront appliqués à des codes convolutifs ou codes en blocs. Les algorithmes itératifs de type Belief Propagation qui s'appliquent sur des structure graphiques des codes, comme pour les codes LDPC.

2.5 Démodulation MAP bit et systèmes BICM.

2.5.1 Démodulation souple : détection MAP bit.

L'avènement des systèmes de codage à décodage souple et itératifs s'est accompagné du développement de la détection "souple" des modulations. Comme nous le verrons par la suite, l'utilisation de codes dîts "modernes" tels que les turbo-codes (déjà vieux de 30 ans) et les codes LDPC (presque 60 ans...) n'est possibles que si à l'entrée du décodeur on fournit une entrée "souple", souvent homogène à un LLR. Nous allons donc voir le principe de la démodulation souple, ie. la détection MAP d'un bit appartenant à un symbole non binaire M-aire qui est associé à une constellation M-aire.

Soit le vecteur m-binaire de bits $\mathbf{x}[n] = [x_1[n] \cdots x_m[n]]$ qui représente l'étiquette binaire qui sera utilisée pour pour identifier un symbole d'une constellation$M$-aire, $m = \log_2(M)$. Les symboles sont supposés indépendants et équi-distribués. Chaque vecteur $\mathbf{x}[n]$ est alors "mappé" sur un symbole dune constellation M-aire $\mathcal{S}$ de cardinalité $|\mathcal{S}| = M$ en suitvant la règle de mapping donnée par la fonction bijective suivante

$$\begin{array}{rcl} \mathcal{M} : \mathbb{F}_2^m & \longrightarrow & \mathcal{S} \subset \mathbb{C} \\ x & \mapsto & s = \mathcal{M}(x) \end{array} \tag{2.19}$$

Un critère classique de détection symbole est donné par la règle du maximum de vraisemblance donné par

$$\hat{s}_n = \arg\max_{s_n} p(y[n]|s[n]),$$

où $p(y[n]|s[n])$ est la vraisemblance associée à l'observation $y[n]$. Cette derbière est supposée connue au récepteur car on sait caractériser la loi de transition du canal. De manière similaire au cas bloc, on peut également appliqué un critère MAP symbole qui est donné par

$$\hat{s}_n = \arg\max_{s_n} p(y[n]|s[n])p(s[n]),$$

si on possède un a priori sur le symbole $s[n]$. Maximum de vraisemblance et MAP sont équivalents si on a un apriori uniforme (ce qui est équivalent à ne pas en avoir...), dans ce cas $p(s[n]) = 1/M$. Ces deux dernières règles sont utiles pour un décodage à sorties "dures". Après "demapping", ie. appliquaton de la règle de mapping inverse donnée par $\mathcal{M}^{-1}(.)$, on pourra fournir des bits décidés à un décodeur de canal utilisant des entrées dures/décidées. Cependant, si on considère que le code de canal utilisé pour la transmission est un code binaire et que son décodage peut faire usage d'entrées "souples", ie. sous forme de probabilités bit ou de manière équivalente sous forme de LLRs, il convient alors de dériver une règle de démodulation à décisions "souples" au niveau bit. On va donc dériver un critère de détection MAP bit dont l'expression dans le domaine lograrithmique est donnée par

Définition 22 : *Démodulation MAP bit*

Soit $\mathbf{x}[n] = [x_1[n] \cdots x_m[n]]$ le vecteur binaire de m bits correspondant à l'étiquetage du symbole $s[n]$ d'une constellation M-aire $\mathcal{S}$, le log-rapport de probabilité est alors donné par

$$\begin{aligned} L(x_i[n]) &= \log\left(\frac{p(x_i[n]=0 \mid y[n])}{p(x_i[n]=1 \mid y[n])}\right) \\ &= \log\left(\frac{\sum_{s[n]\in\mathcal{S}_0^i} p(y[n]|s[n])p(s[n])}{\sum_{s[n]\in\mathcal{S}_1^i} p(y[n]|s[n])p(s[n])}\right) \end{aligned} \tag{2.20}$$

où $\mathcal{S}_0^i \subset \mathcal{S}$ (resp. $\mathcal{S}_1^i$) est le sous ensemble des symboles $s \in \mathcal{S}$ tels que le i-ième bit $x_i[n]$ du vecteur $x[n]$, tel que $s[n] =$ est le bit $'0'$ (resp. le i-ième bit $x_i[n]$ du vecteur $x[n]$ est le bit $'1'$). $p(s[n])$ est la probabilité a priori de $s[n]$ qui sera considéré comme uniforme pour la plupart des cas d'intérêt.

Comme une a une relation bijective entre $s[n]$ et $x[n]$, on peut donner également donner l'expression

$$\begin{aligned} L(x_i[n]) &= \log\left(\frac{p(x_i[n]=0 \mid y[n])}{p(x_i[n]=1 \mid y[n])}\right) \\ &= \log\left(\frac{\sum_{x[n]\in\mathcal{X}_0^i} p(y[n]|x[n])p(x[n])}{\sum_{x[n]\in\mathcal{X}_1^i} p(y[n]|x[n])p(x[n])}\right) \end{aligned} \tag{2.21}$$

où $\mathcal{X}_0^i \subset \mathbb{F}_2^m$ est l'ensemble des vecteurs de $\mathbb{F}_2^m$ de composante i nulle (définition similaire pour $\mathcal{X}_1^i$). On aura également $p(y[n]|x[n]) = p(y[n]|s[n] = \mathcal{M}(x([n])$. Quand une a rpiori sera disponible, ce sera souvent un apriori indépendant pour chaque bit du vecteur $x[n]$, ce qui s'écrit comme

$$p(x[n]) = \prod_{i=1}^{m} p(x_i[n]).$$

Cette dernière relation pourra également s'écrire à partir des LLRs a priori associés.

2.5.2 Capacité des systèmes BICM

Les capacités à entrées contrantes sont déterminées pour des systèmes qui opérent (codent) dans l'espace de la modulation. Pour des raisons pratiques (facilité de design, complexité), la plupart des schémas codés considèrent une concaténation série entre code binaire et une modulation $M = 2^m$-aire éventuellement séparé par un entrelaceur. Dans la plupart des système BICM, le décodeur de canal utilise des entrées "souples", il faut donc utiliser un système à démodulation souple. L'ajout d'un entrelaceur dépend du système de codage utilisé. Généralement si le code externe est un code défini par un treillis ou un code en bloc court, il faut ajouter un entrelaceur. Si on a un code de type LDPC, nous verrons que ce n'est pas utile. Une fois le système spécifié, on peut maintenant se poser la question de performances asymptotiques des systèmes BICM. Comme on démodule pour se ramener à l'estimation d'une information binaire, si on inclut symboliquement l'opération de modulation/démodulation dans le canal, alors un système BICM peut être vu comme un canal constitué de $\log_2(M)$ canaux parallèles *indépendants*, l'indépendance étant garantie théoriquement par l'ajout de l'entrelaceur entre le code et la modulation. La capacité atteignable dépend du mapping utilisé et est donné par

$$C_{\text{bicm}} = m - \frac{1}{2}\sum_{k=0}^{m-1}\sum_{c=0}^{1}\mathbb{E}\left(\log_2\left(\frac{\sum_{s_i\in\mathcal{S}} p(y|s_i)}{\sum_{s_j\in\mathcal{S}_c^k} p(y|s_j)}\right)\right)$$

On peut montrer la formule suivante, plus commode à utiliser et interpréter :

$$C_{\text{bicm}} = \sum_{k=1}^{m}\mathbf{I}(X_k;Y) \leq C_{AWGN},$$

où $\mathbf{I}(X_k;Y) = 1 - \mathbb{E}_{X_k,Y}(\log_2(1+e^{-(1-2X_k)L(X_k)}))$. $\mathbf{I}(X_k;Y)$ est la capacité binaire associée au canal k. Notons également que $L(X_k)$ est une fonction implicite de Y. A partir de ces expressions, on peut alors facilement calculer une estimée

$$\hat{I}(X_k,Y) = 1 - \frac{1}{N_k}\sum_{n}(\log_2(1+e^{-(1-2x_k[n])L(x_k[n])})).$$

En général pour les modulations linéaires, le mapping de Gray permet d'être quasi-optimal pour les efficacités moyennes à grandes.

Chapitre 3

Codes convolutifs : structure et décodage

3.1 Un exemple : le code $(5,7)_8$

3.1.1 Représentation et notations

La représentation sous forme de registre à décalage du code convolutif $(5,7)_8$ est donnée par la figure 4.4. C'est un code qui a un bit d'information $u[n]$ en entrée fait correspondre deux sorties $c_1[n]$ et $c_2[n]$ comme données sur la figure 3.1. Les positions du registre qui

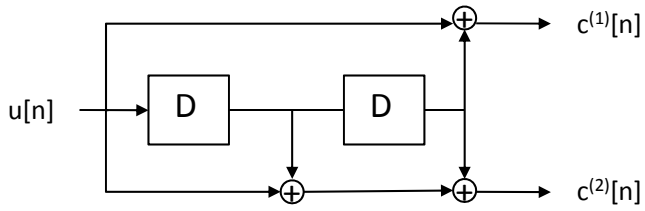


FIGURE 3.1 – Registre à décalage pour codage du code $(5,7)_8$

permettent de calculer les sorties codées par OU-exclusif (addition sur GF(2)/addition mod-2) peuvent être représentées par des vecteurs $g^{(1)} = [101] = [5]_8$ et $g^{(2)} = [111] = [7]_8$. Si on note D l'opérateur du retard, on peut également rencontrer la représentation polynômiale suivante $g^{(1)}(D) = 1 + D^2$ et $g^{(2)}(D) = 1 + D + D^2$. Le rendement de codage est défini par $R = k/n = 1/2$ (k entrées, n sorties). La mémoire ν du code est définie comme le nombre de bits dans le registre et la longueur de contrainte est donnée par $l_c = \nu + 1$.

3.1.2 Relations entrées-sorties

Soit $u[n]$ la séquence d'entrée, on a alors comme sorties

$$c^{(i)}[n] = u \circledast g^{(i)}[n],\ \forall i \in \{1,2\}$$

où $\circledast$ représente le produit de convolution sur $GF(2)$ (d'où le nom de code convolutif).

On introduit généralement la transformée en D telle

$$\mathcal{D}\{x[n]\} \triangleq \sum_{n\in\mathbb{Z}} x[n]D^n = X(D).$$

Cette transformée est l'équivalent de la transformée en $\mathcal{Z}$ où l'opérateur D est préféré à l'opérateur z^{-1} pour la définition de l'opérateur du retard. Ainsi toute convolution se traduit dans le domaine associé par le produit des polynômes, ie.

$$\mathcal{D}\{x \circledast y[n]\} = X(D)Y(D).$$

On peut alors adopter la notation polynomiale suivante $c^{(i)}(D) = u(D)g^{(i)}(D),\ \forall i \in \{1,2\}$. On obtient au final la *Représentation vectorielle polynomiale*

$$\underline{c}(D) = [c^{(1)}(D), c^{(2)}(D)] = u(D)[g^{(1)}(D), g^{(2)}(D)] = u(D)G(D)$$

où $G(D)$ matrice génératrice polynômiale de taille $k \times n$.

3.1.3 Terminaison de codage

Pour un code convolutif, différents types de terminaison pour le codage sont possibles.

1. **Troncature directe :**

   Dans cette méthode, le codage s'arrête quand on n'a plus de bits d'information à coder. Dans ce cadre, le codeur peut terminer dans n'importe quel état de sa mémoire (le contenu du registre est alors quelconque). Pour K bits en entrée, on a émis $N = K/R$ bits codés (cas d'un codage sans poinçonnage).

2. **Fermeture de treillis :**
   Dans cette méthode, on ajoute aux K bits d'information ν bits (au maximum), dits bits de fermeture (*tailing bits*), pour rejoindre l'état 'tout-à-zéro' du registre (le registre ne contient que des '0'). Cela entraîne une baisse du rendement R_t car pour K bits d'information, on envoie au plus $(K+\nu)/R_0$, où R_0 et le rendement initial du code. Ici, $R_t = K/2*(K+2)$. Pour un code récursif, les bits de fermeture dépendent de l'état du registre après K bits codés et il ne suffit plus de mettre u train de bits $'0'$ pour retourner à l'état 'tout-à-zéro' du registre.

3. **Fermeture circulaire de treillis, *tail-biting* :**

   Pour ce type de méthode, on réalise un encodage "circulaire" des données ce qui permet de s'affranchir de bits de fermeture et donc d'être plus efficace en rendement. Pour un code convolutif non récursif, ce codage est réalisé en recopiant les ν bits de fin de mot d'information au début (sorte de préfixe assurant le caractère cyclique de la convolution entre la séquence d'information et la réponse impulsionelle du filtre associée au codeur). L'état initiale du registre est alors donné par les ν bits de fin de mot d'information. On assure ainsi que le codeur terminera dans le même état après K bits d'information codés. Pour un code récursif, la procédure est un peu plus sophistiquée. Mais l'idée est toujours la même : assurer que le codeur débute et termine le codage dans le même état. Pour K bits en entrée, on a émis $N = K/R$ bits codés (cas d'un codage sans poinçonnage).

3.2 Codes convolutifs : représentations

On considère de manière générale un codeur avec k entrées et n sorties définissant un code convolutif de rendement $R = k/n$.

3.2.1 Représentation par filtrage

Si on considère le cas simple des codes convolutifs non récursifs, on peut aisément définir chaque sorties comme étant des opérations de filtrage des différentes entrées à l'aide des réponses impulsionnelles finies associées aux fonctions de transfert entre la sorti i et la sortie k. Ainsi si on note $\{u^{(1)}[n], u^{(2)}[n], \ldots, u^{(k)}[n]]\}$ les différents flux en entrée du code de rendement $R = k/n$, on caractérise le transfert entre la sortie j et l'entrée i par $\{g_i^{(j)}[n]\}$. On a alors la sortie $c^{(j)}[n]$ qui sera donnée par

$$c^{(j)}[n] = \sum_{i=1}^{k} g_i^{(j)} \circledast u^{(i)}[n].$$

Même si elle s'entend pour les codes récursifs qui ont une réponse impulsionnelle de transfert infinie, cette notation est moins commode que celle qui suit qui utilise la représentation polynomiale.

3.2.2 Représentation vectorielle polynomiale

soient $\underline{u}(D) = [u^{(1)}(D), u^{(2)}(D), \ldots, u^{(k)}(D)]$, $\underline{c}(D) = [c^{(1)}(D), c^{(2)}(D), \ldots, c^{(n)}(D)]$, les représentations matricielles des transformées en $\mathcal{D}$. En prenant la transformée en $\mathcal{D}$ de $g_i^{(j)}[n]$ donnée par $\mathcal{D}\{g_i^{(j)}[n]\} = g_i^{(j)}(D)$, il vient alors

$$\underline{c}(D) = \sum_{i=1}^{k} u^{(i)}(D)\underline{g}_i(D)$$

où $\underline{g}_i(D) = [g_i^{(1)}(D), g_i^{(2)}(D), \ldots g_i^{(n)}(D)]$ et $g_i^{(j)}(D)$ représente le transfert entre l'entrée i et la sortie j. On peut alors donner la défintion suivante d'une matrice génératrice polynômiale associée à un code convolutif.

Définition 23 : *Matrice énératriceg polynomiale*

On appelle matrice génératrice polynômiale la matrice $G(D)$ de taille $k \times n$ telle que

$$\underline{c}(D) = \underline{u}(D)G(D)$$

avec

$$G(D) = \begin{pmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \cdots & g_1^{(n)}(D) \\ g_2^{(1)}(D) & g_2^{(2)}(D) & \cdots & g_2^{(n)}(D) \\ \vdots & \vdots & \ldots & \vdots \\ g_{k-1}^{(1)}(D) & g_{k-1}^{(2)}(D) & \cdots & g_{k-1}^{(n)}(D) \\ g_k^{(1)}(D) & g_k^{(2)}(D) & \cdots & g_k^{(n)}(D) \end{pmatrix}$$

Définition 24 : *Matrice de parité polynomiale*

On peut également pour un code convolutif définir une matrice de parité définie par

$$\underline{c}(D)H^T(D) = \mathbf{0}$$

et donc $G(D)H^T(D) = \mathbf{0}$

Ces représentations polynomiales joueront le même rôle que les matrices génératrice et de parité dans le cas de codes en bloc.

3.2.3 Codes convolutifs récursifs

Définition 25 : *Transfert récursif*

Certaines relations entrées-sorties peuvent être définies par un transfert de type récursif (ie. à réponse impulsionnelle infinie). Une implémentation simple (Type I) entre l'entrée i et la sortie j est donnée par

$$g_i^{(j)}(D) = \frac{b_0 + b_1 D + b_2 D^2 + \ldots + b_m D^m}{1 + a_1 D + a_2 D^2 + \ldots + a_m D^m}$$

dont la représentation sous forme de registre est donnée en figure 3.2.

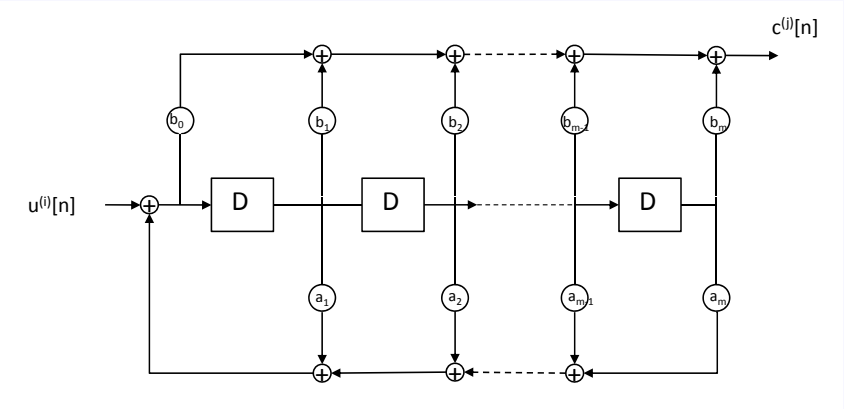


FIGURE 3.2 – Implémentation d'un transfert récursif par réalisation d'un filtre récursif à réponse impulsionnelle infinie de Type I.

Dans ce cas, on obtient la relation entrée-sortie suivante

$$c^{(j)}(D) = g_i^{(j)}(D) u^{(i)}(D)$$

ce qui correspond à une écriture plus aisée de l'équation de convolution avec la réponse impulsionnelle infinie donnée par $c^{(j)}[n] = g_i^{(j)} \circledast u^{(i)}[n]$. Cela correspond en fait à la réalisation à l'aide de l'équation aux différences associées donnée par

$$c^{(j)}[n] = \sum_{l=1}^{m} a_l \, c^{(j)}[n-l] + \sum_{k=0}^{m} b_k \, u^{(i)}[n-k]$$

On retrouve là encore le même formalisme que pour le filtrage en trainement numérique du signal. Un code convolutif défini par une matrice génératrice ayant des éléments de transfert récursifs est appelé code convolutif récursif.

Par exemple, le code récursif systématique $(1, 5/7)_8$ a pour matrice génératrice polynomiale

$$G(D) = [1 \frac{1 + D^2}{1 + D + D^2}]$$

Sa représentation est donnée en figure 3.3.

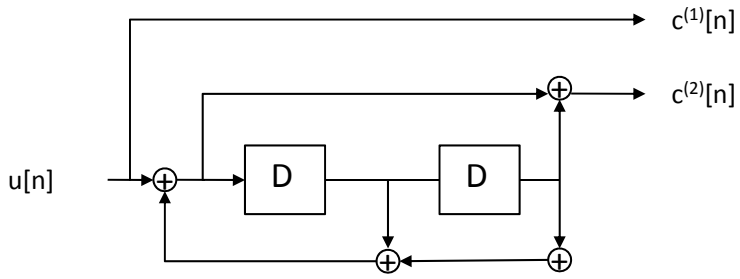


FIGURE 3.3 – représentation du code récursif (1,5/7)

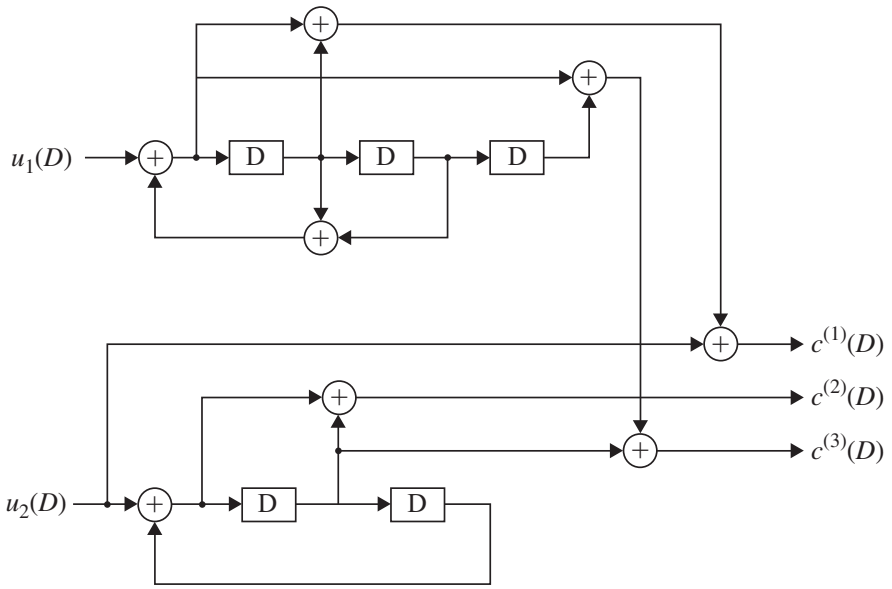


FIGURE 3.4 – Un exemple de code convolutif de rendement R=2/3.

3.2.4 Exemple générale

On considère le code convolutif donné par la figure 3.4. Montrer que la matrice génératrice polynomiale peut s'écrire comme suit

$$G(D) = \begin{pmatrix} \frac{1+D}{1+D+D^2} & 0 & 1+D \\ 1 & \frac{1}{1+D} & \frac{D}{1+D^2} \end{pmatrix}$$

$$\underline{u}(D) = [u_1(D), u_2(D)], \underline{c}(D) = [c_1(D), c_2(D), c_3(D)]$$

3.2.5 Représentation par machine à états finis et diagramme d'état associé.

Sans perte de généralité, on considérera ici un code de rendement $R = 1/n$ pour faciliter l'exposé. Pour un code convolutif, on peut donner une représentation dîte d'état du codeur (automate associé) en définissant un état à partir d'un vecteur représentant le contenu des registres du codeur (la mémoire du codeur). Ainsi si le codeur est dans un état donné $\underline{S}_n$ (ses registres sont initialisé avec certaines valeurs), la donnée des bits d'entrée permet de donner

de manière déterministe les sorties. Ce qui est formellement représenté par le diagramme suivant :

$$\{u_k\} \rightarrow \boxed{\underline{S}_k} \rightarrow \{c_k\}$$

où $c_k = [c_k^{(1)}, c_k^{(2)}, c_k^{(n)}]$ et $\underline{S}_k$ représente l'état interne du registre à décalage.

On peut alors d'écrire ce modèle d'état à l'aide d'équations fonctionnelles

- Equation d'évolution :
  Par cette équation, on peut décrire formellement/analytiquement le passage d'un état $\underline{S}_{k-1}$ à $\underline{S}_k$ en fonction de l'entrée u_k. On écrit cette relation comme

$$\underline{S}_k = F_1(\underline{S}_{k-1}, u_k)$$

- Equation d'observation :
  Cette équation traduit la génération des sorties *observables* du modèle et qui sont ici les bits codés c_k. On écrit cette relation comme

$$c_k = F_2(\underline{S}_{k-1}, u_k)$$

Par exemple, pour le code $(7,5)_8$, on a $\underline{S}_k = [u_k, u_{k-1}]$. Cette automate peut être visualiser à l'aide d'un diagramme d'état qui est un graphe qui représente les différents états possibles et qui visualise les transitions entre ses états et les sorties associées. Un exemple est donné pour le code convolutif $(7,5)_8$ à la figure 3.5. Les états représentés par des cercles sont associés au contenu du/des registre(s). Les flèches représentent les transitions entre états et les labels indiquent les bits entrant dans le(s) registre(s) et les bits codés associés.

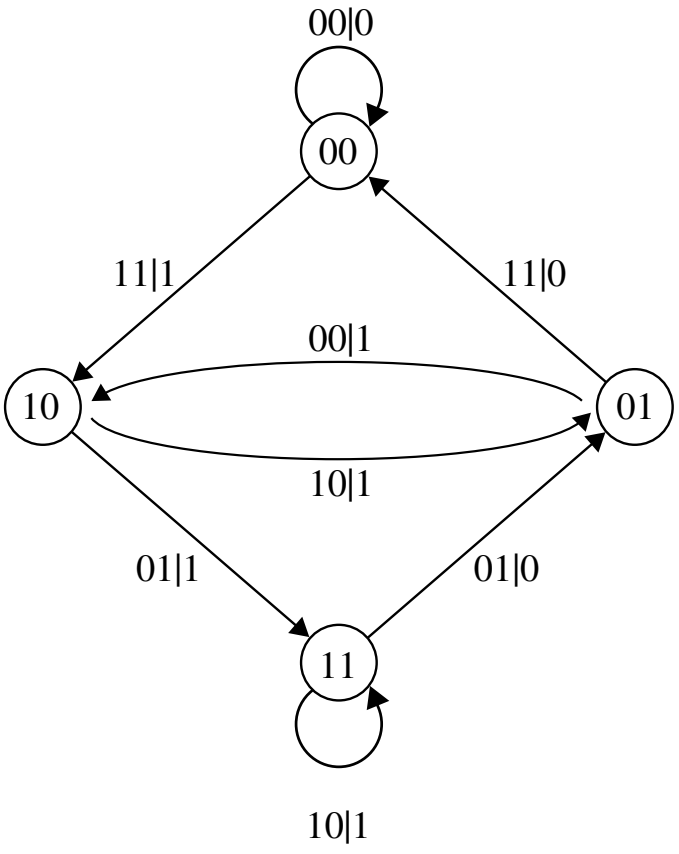


FIGURE 3.5 – Diagramme d'état du code convolutif $(7,5)_8$

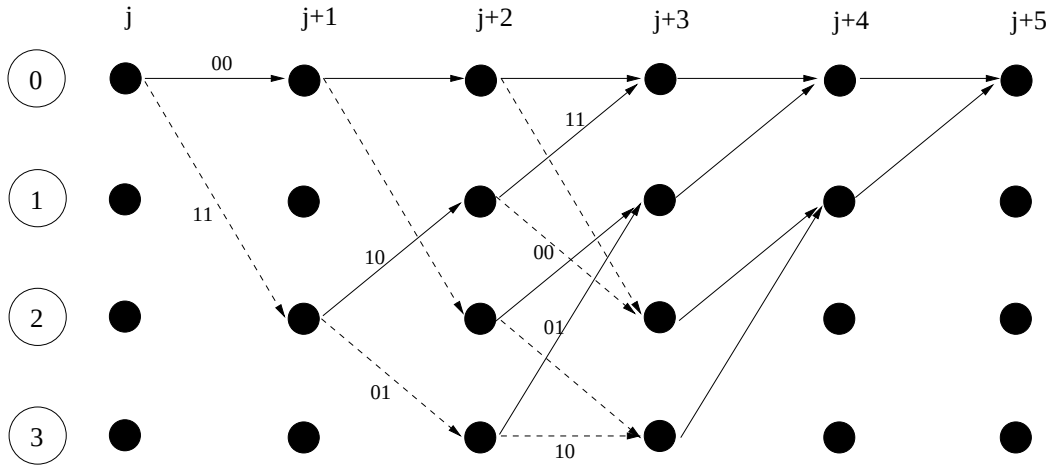


FIGURE 3.6 – Treillis du code (7,5)

3.2.6 Représentation graphique par un treillis

Définition 26 : *Treillis*

Le treillis est la représentation graphique du code dans son espace d'état en considérant la dimension temporelle. On définit alors les éléments suivants

1. **Noeuds :** noeuds du graphe associé à un état $\underline{S}_k$.
2. **Transitions :** branches entre deux noeuds associées à $F_1(.)$.
3. **Etiquettes/labels :** informations portées par les branches (u_k, c_k) et données par $F_2(.)$

Propriétés 7 : *Propriétés associées à un treillis de code convolutif*

1. Pour un treillis de longueur L, tout chemin du treillis de $\underline{S}_0$ à $\underline{S}_L$ est mot de code obtenu par concaténation des étiquettes c_k du chemin.
2. Le chemin où tous les $\underline{S}_k$ sont l'état **0** (représentation binaire naturelle) est le mot de code nul.
3. *Événement minimal :* chemin qui par de l'état $\underline{S}_k = \mathbf{0}$ et ne revient à cet état que à $\underline{S}_{k+l}$ pour un certain l. On lui associe un poids de Hamming grâce aux étiquettes du chemin.
4. $d_{\min}$ est donnée par le poids minimal d'un évênement minimal.

Un exemple est donné pour le code $(7, 5)_8$ figure 4.4.

3.3 Décodage par Maximum de Vraisemblance

Dans cette section, on s'intéresse au décodage par maximum de vraisemblance (ou encore appelé *MLSE*) comme défini auparavant. On explicite ici comment ce type de décodage s'instanti dans le cas des codes convolutifs.

3.3.1 Critère de décodage MLSE

Pour rappel, le critère MLSE se décline toujours sous la forme

$$\begin{aligned} \hat{\mathbf{c}} &= \arg\max_{\mathbf{c}} p(\mathbf{y}|\mathbf{c}) \\ &= \arg\max_{\mathbf{c}} \prod_{n} p(y_n|c_n) \end{aligned}$$

La dernière égalité venant du fait que l'on considère une canal sans mémoire à entrées antipodales (BPSK) pour première présentation. Si on considère un code convolutif avec fermeture de treillis avec un treillis comportant L sections, et en reprenant les notations précédentes sur les codes convolutits, on peur alors écrire

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} \prod_{k=1}^{L} p(\mathbf{y_k}|\mathbf{c_k})$$

où $\mathbf{y_k} = (y_k^{(1)}, \cdots, y_k^{(n_c)})$ et $\mathbf{c_k} = (c_k^{(1)}, \cdots, c_k^{(n_c)})$.
On aura donc finalement

$$p(\mathbf{y_k}|\mathbf{c_k}) = \prod_{n=1}^{n_c} p(y_k^{(n)}|c_k^{(n)}).$$

En posant $\lambda_k(\mathbf{c_k}) = -log(p(\mathbf{y_k}|\mathbf{c_k}))$, toujours positif, il vient

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} \sum_{k=1}^{L} \lambda_k(\mathbf{c_k})$$

3.3.2 Algorithme de Viterbi

L'estimation de la séquence la plus vraisemblable sera réalisée efficacement à l'aide de l'algorithme de Viterbi, premier à proposer cette solution. L'idée originale de Viterbi est d'utiliser le treillis associé au code convolutif pour énumérer efficacement l'ensemble des chemins possibles. En utilisant, le treillis, cet algorithme permet à chaque section de treillis de ne garder que les $|\mathcal{S}|$ chemins les plus vraisemblables terminant dans chaque état. Rappelons que, de par la représentation d'état, on a $\mathbf{c_k}(\mathbf{s_{k-1}}, \mathbf{s_k})$, ie. les labels des bits codés émis $\mathbf{c_k}$ qui sont une fonction déterministe de l'état de départ s_{k-1} et l'état d'arrivée s_k. On a alors les propriétés suivantes :

Propriétés 8 : *propriétés*

1. Correspondance entre espace des séquences et des états

$\{c[n]\|n=1\cdots N\}$	$\Longleftrightarrow$	$\{s[n]\|n=0\cdots L\}$
Espace des séquences codées		Espaces des Etats

2. Chaque chemin sur le treillis représente une séquence de bits codés émis possibles :

séquence MLSE	$\Leftrightarrow$	chemin le plus problable, ie de plus petite métrique cumulée sur le treillis

En effet, en utilisant le fait que $\{c[n]|n=1\cdots N\} \Longleftrightarrow \{\mathbf{c_k}|k=1\cdots L\}$ un simple jeu de réécriture permet d'avoir

$$\begin{aligned}
\hat{\mathbf{c}} &= \arg\min_{\mathbf{c}} \sum_{k=1}^{L} \lambda_k(\mathbf{c_k}) \\
&\Updownarrow \\
\hat{\mathbf{c}} &= \arg\min_{s_k} \sum_{k=1}^{L} \lambda_k(\mathbf{c_k}(s_{k-1}, s_k)) \\
&= \arg\min_{s_k} \sum_{k=1}^{L} \lambda_k(s_{k-1}, s_k)
\end{aligned}$$

Pour la section de treillis n, à l'état $s_n = s$, on peut alors écrire

$$\begin{aligned}
\Lambda_n(s_n) &= \min_{\{s_0,s_1,\cdots,s_{n-1},s_n\}} \sum_{k=0}^{n} \lambda_k(s_{k-1}, s_k) \\
&= \min_{\{s_{n-1}\to s_n\}} \left\{ \left\{ \min_{\{s_0,\cdots,s_{n-1}\}} \sum_{k=0}^{n-1} \lambda_k(s_{k-1}, s_k) \right\} + \lambda_n(s_{n-1}, s_n) \right\} \\
&= \min_{\underbrace{\{s_{n-1}\to s_n\}}_{\text{transitions possibles}}} \{\Lambda_{n-1}(s_{n-1}) + \lambda_n(s_{n-1}, s_n)\}
\end{aligned}$$

La procédure complète est alors donnée par

Définition 27 : *Algorithme de Viterbi*

— Pour chaque section n ($n = 1 \cdots N$), pour chaque état $s_n = s$ ($s = 0 \cdots |\mathcal{S}|$) :

1. calculer Λ_n tel que

$$\Lambda_n(s_n) = \min_{\{s_{n-1} \to s_n\}} \{\Lambda_{n-1}(s_{n-1}) + \lambda_n(s_{n-1}, s_n)\}$$

2. stocker l'état précédent s_{n-1} : pour chaque état s_n, on peut donc associer une séquence *survivante* $\{s_0, \cdots, s_n\}$ de métrique cumulée associée $\Lambda_n(s_n)$)

— A la fin du treillis, il ne reste plus que $|\mathcal{S}|$ chemins possibles, alors par parcours arrière (*traceback*) des états du treillis

$$\hat{\mathbf{s}} = \left\{ s_0, s_1, \cdots, s_{L-1}, s_L | \underset{s_L}{\text{argmin}} \{\Lambda_L(s_L)\} \right\}$$

ce qui équivaut compte tenu que

$$\hat{\mathbf{s}} = \{s_0, s_1, \cdots, s_{L-1}, s_K\} \leftrightarrow \hat{\mathbf{u}} = \{u_1, s_2, \cdots, u_{K-1}, u_K\}$$

à sélectionner la séquence d'information la plus probable (Hard Output, HO) en utilisant $(s_{n-1}, s_n) \to u_n$.

Si le treillis est fermé lors de l'étape de codage (retour à zéro), le dernier état est donné systématiquement par $\Lambda_L(s_0)$.

Cas du décodage à entrées dures (Hard Inputs (HI))

Le modèle de canal est alors donné par

$$y_n = c_n \oplus e_n, \forall n = 1, \cdots, N$$

Dans le cadre d'un décodage dure, on a alors

$$\begin{aligned} \lambda_k(s_{k-1}, s_k) &= d_H(\mathbf{y_k}, \mathbf{c_k}) \\ &= \sum_{m=1}^{n_c} d_H(y_k^{(m)}, c_k^{(m)}) \\ &= \sum_{m=1}^{n_c} y_k^{(m)} \oplus c_k^{(m)} \end{aligned} \tag{3.1}$$

Le mot de code sélectionné sera donc bien le mot de distance de Hamming cumulée la plus faible.

Cas du décodage à entrées "souples" (Soft Inputs (SI)), modulation antipodales (BPSK) sur canal Gaussien

Le modèle de canal est alors donné par

$$y_n = c_n + b_n, \quad \forall n = 1, \cdots, N$$

où $b_n \sim \mathcal{N}(0, \sigma_b^2)$.

et donc

$$p(y_n|c_n) \propto e^{-\frac{(y_n-c_n)^2}{2\sigma_b^2}}$$

Dans le cadre d'un décodage souple, après calcul direct, on a alors

$$\begin{aligned} \lambda_k(s_{k-1}, s_k) &= d_E(\mathbf{y_k}, \mathbf{c_k}) \\ &= \sum_{m=1}^{n_c} |y_k^{(m)} - c_k^{(m)}|^2 \end{aligned} \tag{3.2}$$

Le mot de code sélectionné sera donc bien le mot de distance euclidienne cumulée la plus faible.

On peut encore simplifié le problème en développant les termes $|.|^2$, en supprimant les termes constants et/ou communs à chaque métrique de branche de la section courante du trellis, on obtient une nouvelle métrique :

$$\tilde{\lambda}_k(s_{k-1}, s_k) = -\sum_{m=1}^{n_c} y_k^{(m)} c_k^{(m)} \tag{3.3}$$

On définit alors finalement par

$$\Gamma_k(s_{k-1}, s_k) = \sum_{m=1}^{n_c} y_k^{(m)} c_k^{(m)} \tag{3.4}$$

la métrique de branche comme étant la corrélation déterministe partielle entre la séquence observée et le mot de codes émis. Comme on a changé le signe de cette métrique, il convient alors de remplacer l'opération de min par max dans l'algorithme de Viterbi présenté. On interprète alors cela comme une maximisation de l'intercorrélation entre le mot émis et la séquence observée.

Cas du décodage à entrées "souples" (Soft Inputs (SI)), modulation M-aire sur canal Gaussien

On s'intéresse maintenant au cas où le mot émis est mappé sur une constellation d'ordre élevée. Pour commencer, on considère que l'on a un ordre de constellation $M = 2^{n_c}$, ie. chaque label $\mathbf{c_k}$ est **directement** mappé sur un symbole de la constellation. On note $x_k = \mathcal{M}(\mathbf{c_k})$ où $\mathcal{M}(.)$ est l'opérateur de mapping.

Le modèle de canal est alors donné par

$$y_n = x_n + b_n, \quad \forall n = 1, \cdots, L$$

où b_n est un bruit blanc Gaussien complexe circulaire suivant une loi $\mathcal{CN}(0, \sigma_b^2)$, ie. $b_n = b_{I,n} + ib_{Q,n}$ où $b_{I,n}$ et $b_{Q,n}$ suivent une loi normale réelle $\mathcal{N}(0, \sigma_b^2/2)$. On a donc

$$\begin{aligned} p(y_n|x_n) &= p(b_n) \\ &= p(b_{I,n}, b_{Q,n}) \\ &\propto e^{-\frac{|y_n-x_n|^2}{\sigma_b^2}} \end{aligned} \tag{3.5}$$

Le décodage ML s'écrit comme précédemment

$$\begin{aligned}\hat{\mathbf{c}} &= \arg\max_{\mathbf{c}} p(\mathbf{y}|\mathbf{c}) \\ &= \arg\max_{\mathbf{c}} \prod_n p(y_n|c_n) \\ &= \arg\max_{\mathbf{x}} \prod_n p(y_n|x_n)\end{aligned}$$

Comparé au cas binaire précédent, on a directement accès à $p(y_n|x_n), \forall n$. Dans ce cas la métrique de branche se simplifie comme suit

$$\begin{aligned}\lambda_k(s_{k-1}, s_k) &= d_E(y_k, x_k) \\ &= |y_k - x_k|^2 \end{aligned} \tag{3.6}$$

On retrouve là encore le fait que la séquence retenue au final sera la séquence la plus proche de la séquence reçue. On peut également simplifier le décodage en considérant la métrique

$$\tilde{\lambda}_k(s_{k-1}, s_k) = -2\mathrm{Re}\{y_k x_k^*\} + |x_k|^2 \tag{3.7}$$

Ceci s'interprète comme une corrélation "corrigée" par la puissance du symbole considéré.

Si la modulation est à module constant, on pourra simplifier en

$$\Gamma_k(s_{k-1}, s_k) = 2\mathrm{Re}\{y_k x_k^*\} \tag{3.8}$$

Dans ce cas, on revient à une maximisation de corrélation entre séquence de symboles reçus et séquence de symboles émis.

Cas du décodage à entrées "souples" (Soft Inputs (SI)), modulation M-aire à bits entrelacés sur canal Gaussien

Pour simplifier la complexité induite par l'utilisation d'un code adapté à l'ordre de modulation (modulation codées en treillis dont le cas précédent est un cas trivial), on considère un schéma pour lequel on place un entrelaceur entre le code et la modulation. Le mot de code $\mathbf{c} = [c_1, \cdots, c_N]$ est entrelacé par l'entrelaceur Π. On obtient alors une séquence de symboles $\mathbf{x} = [\mathbf{x_1}, \cdots, \mathbf{x_{N_s}}]$ obtenus en regroupant les bits codés entrelacés en groupes de m bits tels que $\mathbf{x_n} = [x_n^1, \cdots, x_n^m]$. $[x_n^1, \cdots, x_n^m]$ définit l'étiquette binaire pour le mapping sur le symbole $\mathbf{s_n} = \mathcal{M}(\mathbf{x_n}) \in \mathbb{C}$. L'entrelaceur Π définit une relation bijective

$$\begin{aligned}\pi : [1, \cdots, N] &\mapsto [1, \cdots N_s] \times [1, \cdots m] \\ k &\qquad (k', i)\end{aligned} \tag{3.9}$$

entre le bit codé $c_k \in \{0, 1\}$ et le i-ème bit $x_{k'}^i$ du symbole M-aire $\mathbf{x_{k'}}$. On a $N_s = N/m$.

$$\mathcal{M} : x \to s$$

définit la fonction bijective de mapping. Au niveau du récepteur, un décodage conjoint code-modulation n'est plus possible. On réalise alors une démodulation souple qui va estimer de manière souple les bits du mots de codes. Cette information souple est donnée sous la forme d'une log-vraisemblance ou d'un log-rapport de vraisemblance (LLR). La quantité obtenue traduit le fait que l'on a transformé un canal vectoriel M-aire en m canaux binaires équivalents dont les caractéristiques moyennes dépendent de la modulation et du mapping utilisés. La définition suivante donne les quantités *souples* que l'on peut estimer sur les bits codés émis.

Définition 28 : *Démodulation souple*

On suppose la transmission de symboles M-aire sur canal Gaussien avec $M = 2^m$. Les vecteurs binaires $x[n] = [x^1[n], \cdots, x^m[n]] \in \mathcal{X} = \{0,1\}^m$ sont "mappés" sur des symboles $s[n] \in \mathcal{S} \subset \mathbb{C}$. On suppose que les symboles $x[n]$ (et donc $s[n]$) sont équi-distribués. Les symboles observés sont donnés par $\mathbf{y} = [\mathbf{y_1}, \cdots, \mathbf{y_{N_s}}]$. On supposera le canal sans mémoire. On peut définir alors deux quantités relatives à une décision *souple*.

1. *log-vraisemblance/métrique bit ML :*

$$\lambda^i(y_{k'}, b) = \log \sum_{\mathbf{x} \in \mathcal{X}_b^i} p(\mathbf{y_{k'}}|\mathbf{x}),\ \forall i = 1, \cdots, m.$$

   $\mathcal{X}_b^i$ est le sous-ensemble des symboles $\mathbf{x} \in \mathcal{X}$ ayant $x_i[n] = b \in \{0,1\}$ dans leur étiquette binaire associée.

2. *LLR associé au bit $x_i[n]$:*

L'expression $\lambda^i(y_{k'}, b)$ est en faîte dérivée de la vraisemblance de l'obersvation $\mathbf{y_{k'}}$ sachant le bit de label $x^i_{k'}$. Ainsi on a

$$\begin{aligned} p(\mathbf{y_{k'}}|x^i = b) &= \sum_{\mathbf{x} \in \mathcal{X}} p(\mathbf{y_{k'}}|\mathbf{x}) p(\mathbf{x}|x^i = b) \\ &= \frac{1}{2^{m-1}} \sum_{\mathbf{x} \in \mathcal{X}_b^i} p(\mathbf{y_{k'}}|\mathbf{x}) \end{aligned} \tag{3.10}$$

en notant que

$$p(\mathbf{x})|x^i = b) = \begin{cases} 2^{-(m-1)} & , \text{ si } x^i = b \\ 0 & , \text{ sinon} \end{cases}$$

En passant au log et en ne gardant pas le terme constant, on obtient la métrique voulue.

Le décodage ML du mot de code émis est alors réalisé à partir des métriques bit supposées indépendfantes après entrelacement supposé parfait. La règle de décision est alors donnée par

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} \sum_{n=1}^{N} \lambda^i(\mathbf{y_{k'}}, c_n) \tag{3.11}$$

Pour un code convolutif, si on considère un décodage par Viterbi sur le treillis cela revient à considérer le probleme suivant

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} \sum_{k=1}^{L} \lambda_k(s_{k-1}, s_k) \tag{3.12}$$

où, avec quelques abus de notation, la métrique de branche s'exprime comme

$$\lambda_k(s_{k_1}, s_k) = \sum_{l=1}^{n_c} \lambda^i(y_{k'}, c_k^l)$$

où $\lambda^i(y_{k'}, c_k^l)$ représente la log-vraisemblance associée au bit codé c_k^l quand on redéfinit l'entrelacement comme étant la fonction

$$\begin{array}{ccc} \pi : [1, \cdots, L] \times [1, \cdots, n_c] & \mapsto & [1, \cdots, N_s] \times [1, \cdots, m] \\ (k, l) & & (k', i) \end{array}$$

où N_s est le nombre de symboles modulés codés émis sur le canal ($N_s = (L.n_c)/m$)

3.4 Décodage MAP symbole

3.4.1 Critère MAP bit

On s'intéresse ici au décodage des codes convolutifs suivant un critère MAP symbole, le symbole étant ici un bit. Ce critère diffère du MAP séquence ou ML. L'idée ici est de calculer la probabilité a posteriori d'un bit d'information (ou codé comme on le verra après) sachant l'ensemble de observations. Ceci est donné par la définition suivante si on considère une séquence de bits d'information $u_n \in \{0,1\}$ de taille K

Définition 29 : *Critère MAP bit*

$$\begin{aligned} \hat{u}_n &= \arg\max_{u_n} p(u_n|\mathbf{y}) \\ &= \frac{1-\text{signe}(L(u_n))}{2} \end{aligned} \tag{3.13}$$

avec

$$L(u_n) = \log\left[\frac{p(u_n=0|\mathbf{y})}{p(u_n=1|\mathbf{y})}\right]$$

La quantité $L(u_n)$ est ce que l'on appelle le LLR (*Log-Likelihood Ratio*) associé au bit u_n.

Le critère est une conséquence du fait que l'on considère une variable binaire. En effet, si on considère le bit u_n, alors le test de décision statistique binaire associé donne

$$\hat{u}_n = \arg\max_{u_n} p(u_n|\mathbf{y}) = 0$$

si

$$p(u_n=0|\mathbf{y}) > p(u_n=1|\mathbf{y}).$$

Par passage au log et définition du LLR associé à u_n on obtient alors $L(u_n) > 0$. De la même manière, on aura $\hat{u}_n = 1$ si $L(u_n) < 0$. On a donc la relation $\text{signe}(L(u_n)) = 1 - 2u_n$ qui nous définit un mapping bijectif entre le signe du LLR et la valeur binaire. Ce mapping bijectif correspond d'ailleurs à un mapping de type BPSK : $\{`0' \leftrightarrow +1, `1' \leftrightarrow -1\}$. Compte tenu de cette relation directe, il est souvent plus commode de directement considérer la version "mappée/modulé" du bit (cas des canaux souples), de sorte que l'on considérera de manière indifférenciée $u_n = 0/1$ et $u_n = \pm 1$ dès lors le contexte en suffisamment clair. Ainsi, on pourra adopter la règle MAP suivante

$$\begin{aligned} \hat{u}_n &= \arg\max_{u_n} p(u_n|\mathbf{y}) \\ &= \text{signe}(L(u_n)) \end{aligned} \tag{3.14}$$

Le LLR associé à u_n peut être associé à une mesure de fiabilité de la décision. En effet, $\text{signe}(L(u_n))$ est identifiable à une décision "dure" du bit à laquelle on peut associer une mesure de fiabilité de cette décision grâce à $|L(u_n)|$. Plus cette dernière valeur est élevée, plus la décision est fiable. On peut également noter que la connaisance du LLR permet de remonter facilement aux probabilité en remarquant que

$$p(u_n=0|\mathbf{y}) + p(u_n=1|\mathbf{y}) = 1 \tag{3.15}$$

$$L(u_n) = \log\left(\frac{p(u_n=0|\mathbf{y})}{p(u_n=1|\mathbf{y})}\right) \tag{3.16}$$

On obtient alors les relations suivantes

$$p(u_n = 0|\mathbf{y}) = \frac{e^{L(u_n)}}{1 + e^{L(u_n)}} \tag{3.17}$$

$$p(u_n = 1|\mathbf{y}) = \frac{1}{1 + e^{L(u_n)}} \tag{3.18}$$

La quantité $L(u_n)$ est souvent appelée *information souple*. Un algorithme qui fournit en sortie des informations homogènes à des LLRs (ou symboles bruités) est appelé *algorithme à sorties souples (Soft Outputs, SO)* par opposition à des algorithmes qui fournissent une version décidée des symboles dénommés *algorithme à sorties dures (Hard Outputs, HO)*. Par analogie, si les entrées sont des symboles décidés, on parle d'algorithem à entrées dures (Hard Input, HI) par opposition à des algorithmes à entrées souples (Soft Inputs, SI) prenant en compte des symboles bruités ou des entrèes représentant des statistiques suffisantes (A DEFINIR).

3.4.2 Algorithme BCJR/Forward-Backward

Notations

Dans la suite, nous adopterons les notations suivantes :

— Sans perte de généralité, et pour simplifier l'exposition, on considérera un code de rendement à 1 entrée et nc sorties. Le rendement initiale avant fermeture est donc $R_0 = 1/n_c$.

— La longueur totale de treillis considéré est $L = K + Nf$ (K bits d'info. + Nf bits de fermeture).

— Pour simplifier l'écriture, on considérera des notations vectorielles. Ainsi le mot de code émis est $\mathbf{c} = [c_1, c_2, \cdots c_L]$ avec $c_k = [c_k^{(1)}, c_k^{(2)}, \cdots, c_k^{(n_c)}]$ le label codé émis de bits codés à l'instant k. De même, on a le vecteur de symboles bruités reçus noté par $\mathbf{y} = [y_1, y_2, \ldots y_L]$ avec $y_k = [y_k^{(1)}, y_k^{(2)}, \ldots, y_k^{(n_c)}]$. y_k est le symbole reçu issu d'un canal à entrées binaires et sans mémoire. Si le canal est additif, on aura $y_k^{(j)} = c_k^{(j)} + b_k^{(j)}$, en supposant $c_k^{(j)}$ la version modulée du bit codé. On notera $\mathbf{y}_k^l = [y_k, y_{k+1}, \cdots y_{l-1}, y_l]$ la suite d'observations entre les sections k et l.

LLR MAP revisité

Dans le cadre des codes convolutifs, on va maintenant dériver l'expression du critère MAP bit qui permet de calculer le critère

C'est un algorithme de type bloc qui considère un traitement par bloc des données. Pour ceci, on utilisera la représentation markovienne du code à l'aide son espace d'état. On notera par s_n l'état d'arrivée du codeur quand le bit u_n a été émis. On a alors une correspondance bijective : $u_n \leftrightarrow (s_{n-1}, s_n)$. En parliculier, on aura besoin de définir les ensembles suivants :

1. Ensemble des transitions associées à $u_n = +1$ sur une section de treillis :

$$\mathcal{S}^+ = \{(s', s) \text{ où } (s_{n-1} = s') \mapsto (s_n = s)|u_n = +1\}$$

2. Ensemble des transitions associées à $u_n = -1$ sur une section de treillis :

$$\mathcal{S}^- = \{(s', s) \text{ où } (s_{n-1} = s') \mapsto (s_n = s)|u_n = -1\}$$

On peut alors dériver l'expression du LLR $L(u_n)$ associé à la décision MAP comme suit

$$\begin{aligned} L(u_n) &= \log\left[\frac{p(u_n=+1|\mathbf{y})}{p(u_n=-1|\mathbf{y})}\right] \\ &= \log\left[\frac{p(u_n=+1,\mathbf{y})}{p(u_n=-1,\mathbf{y})}\right] \end{aligned}$$

On peut noter alors que pour $u_n = +1$ (résultat similaire pour $u_n = -1$), on a

$$\begin{aligned} p(u_n=+1,\mathbf{y}) &= \sum_{s'}\sum_{s} p(u_n=+1, s_{n-1}=s', s_n=s, \mathbf{y}) \\ &= \sum_{\mathcal{S}^+} p(s_{n-1}=s', s_n=s, \mathbf{y}) \end{aligned} \tag{3.19}$$

et donc au final, on peut écrire

$$L(u_n) = \log\left[\frac{\sum_{\mathcal{S}^+} p(s_{n-1}=s', s_n=s, \mathbf{y})}{\sum_{\mathcal{S}^-} p(s_{n-1}=s', s_n=s, \mathbf{y})}\right] \tag{3.20}$$

Il nous reste alors à calculer le terme $p(s_{n-1}=s', s_n=s, \mathbf{y})$.

Factorisation de $p(s', s, \mathbf{y})$

Par application de la règle de Bayes et en utilisant les propriétés markoviennes du code, on peut écrire la factorisation suivante

$$p(s_{n-1}=s', s_n=s, \mathbf{y}) = \alpha_{n-1}(s')\gamma_n(s', s)\beta_n(s) \tag{3.21}$$

où

$$\begin{aligned} \alpha_n(s) &= p(s_n=s, \mathbf{y}_1^n) \\ \beta_n(s) &= p(\mathbf{y}_{n+1}^L | s_n=s) \\ \gamma_n(s', s) &= p(s_n=s, y_n | s_{n-1}=s') \end{aligned}$$

$\alpha_n(s)$ représente alors la probabilité conjointe d'être dans l'état $s_n = s$ et d'avoir les observations associées aux n premières trnasitions du treillis (information du passé). cette quantité est souvent appelée *métrique forward*. De la même manière, $\beta_n(s)$ représente alors la probabilité d'observer les les observations futures sachant l'état $s_n = s$ (information liée aux observations futures). cette quantité est souvent appelée *métrique backward*. Enfin, $\gamma_n(s', s)$ souvent appelé prbabilité/métrique de transition correspond à la vraisemblance liée à l'obervation courante. Elle dépend explicitement du modèle de canal. Ainsi, de manière très grossière, le calcul du MAP pour une section n du treillis (on veut décdier le bit u_n) se réalise par le produit de probabilités associées à une information du passé du future et de l'observation courante. On va voir maintenant comment calculer efficacement ces quantités.

Récursions forward-backward

En utilisant les mêmes ressorts que précédemment, on peut montrer facilement que l'on a les relations suivantes

1. Passe Forward :

$$\alpha_n(s) = \sum_{s'} \gamma_n(s', s)\alpha_{n-1}(s') \tag{3.22}$$

Le calcul de $\alpha_n(s)$ se réalise donc de manière récursive suivant les indices de temps croissants (Phase forward/aller). Le calcul se réalise sur les transitions possibles entre les états de départ s_{n-1} et d'arrivée s_n.

2. Passe Backward :

$$\beta_{n-1}(s') = \sum_{s} \gamma_n(s', s)\beta_n(s) \tag{3.23}$$

Le calcul de $\beta_n(s)$ se réalise donc de manière récursive suivant les indices de temps décroissants (Phase backward/retour). Le calcul se réalise sur les transitions possibles entre les états de départ s_{n+1} et d'arrivée s_n.

Calcul des probabilités de transitions

Il ne reste plus qu'à calculer les propabiltés de transition. De manière directe, le calcul donne

$$\begin{aligned} \gamma_n(s', s) &= p(s_n = s, y_n | s_{n-1} = s') && (3.24)\\ &= p(y_n | s', s).p(s | s') && (3.25)\\ & && (3.26) \end{aligned}$$

avec

$$p(s|s') = \begin{cases} 0 & , \text{ si } \{s' \to s\} \text{ non valide} \\ \pi(u_n) & , \text{ sinon} \end{cases}$$

On peut alors écrire

$$\begin{aligned} \gamma_n(s', s) &= p(y_n | c_n(s', s))\pi(u_n)\mathbb{1}_{s' \to s} \\ &= \prod_{m=1}^{n_c} p(y_n^{(m)} | c_n^{(m)(s',s)})\pi(u_n)\mathbb{1}_{s' \to s} \end{aligned} \tag{3.27}$$

avec $\pi(u_n)$ la probabilité à priori de u_n et $c_n(s', s)$ les bits associés à l'étiquette $(s' \to s)$. La dernière inégalité est obtenue en considérant que le canal est un canal binaire sans mémoire.

Dans le cas où l'on considère une transmission codée sur un canal sans mémoire additif Gaussien, on a

Exemple 8 : *Cas Gaussien*

$$\forall n, \forall m \in \{1, \cdots, n_c\},\ y_n^{(m)} = c_n^{(m)} + b_n^{(m)}, b_n^{(m)} \sim \mathcal{N}(0, \sigma_b^2)$$

$$\begin{aligned} \gamma_n(s', s) &= \prod_{m=1}^{n_c} p(y_n^{(m)} | c_n^{(m)}(s', s)) \\ &= \frac{1}{\left(\sqrt{2\pi\sigma_b^2}\right)^{n_c}} \exp\left(-\frac{\sum_{m=1}^{n_c} |y_n^{(m)} - c_n^{(m)}(s', s)|^2}{2\sigma_b^2}\right)\pi(u_n)\mathbb{1}_{s' \to s} \end{aligned}$$

Initialisation des récursions forward-backward

Les phases forward backward nécessite une initialisation pour les états $\alpha_0(s)$ et $\beta_L(s)$. Cette initialisation dépend du type de fermeture appliquée au code convolutif.

1. **Sans Fermeture de treillis :**

   Dans ce mode, on peut suppoeser le codeur initialisé dans l'état $s_0 = 0$. Seul un état est donc possible. Comme il n'y a pas de fermeture de treillis, tous les états finaux sont donc équi-probables. Dans ce cadre, il la mémoire du code est la mémoire du code ν, on obtient alors

$$\alpha_0(0) = 1 \quad , \quad \alpha_0(s) = 0 \text{ sinon}$$

$$\beta_L(0) = 1/2^{\nu} \quad , \quad \forall s$$

2. **Fermeture de treillis :**

   Dans ce cas, les états de départs et d'arrivée sont connus (on suppose qu'ils sont égaux à $s_0 = 0$).

$$\alpha_0(0) = 1 \quad , \quad \alpha_0(s) = 0 \text{ sinon}$$

$$\beta_L(0) = 1 \quad , \quad \beta_L(s) = 0 \text{ sinon}$$

L'algorithme complet peut alors être énoncé comme suit

Définition 30 : *Algorithme BCJR*

1. **Initialisation (avec fermeture treillis) :**

$$\alpha_0(0) = 1 \quad , \quad \alpha_0(s) = 0 \text{ sinon} \tag{3.28}$$

$$\beta_L(0) = 1 \quad , \quad \beta_L(s) = 0 \text{ sinon} \tag{3.29}$$

2. **Phase forward-backward :**

$$\forall n = 1, \cdots, L-1, \ \alpha_n(s) = \sum_{s'} \gamma_n(s', s)\alpha_{n-1}(s') \tag{3.30}$$

$$\forall n = L-1, \cdots, 1, \ \beta_{n-1}(s') = \sum_{s} \gamma_n(s', s)\beta_n(s) \tag{3.31}$$

3. **Calcul de $L(u_n)$:** $\forall n = 1, \cdots, K$

$$L(u_n) = \log\left[\frac{\sum_{\mathcal{S}^+} p(s_{n-1} = s', s_n = s, \mathbf{y})}{\sum_{\mathcal{S}^-} p(s_{n-1} = s', s_n = s, \mathbf{y})}\right] \tag{3.32}$$

Probabilité des états

On peut également accéder à la probabilité des états, relation utile dans certains (codeur à faible nombre d'état) car elle évite l'utilisation d'un calcul directe de $p(s_{n-1} = s', s_n = s, \mathbf{y})$. En effet, un calcul rapide permet d'écrire

$$\lambda_n(s) = p(s, \mathbf{y}) = \alpha_n(s)\beta_n(s), \forall s, n$$

TRAITER l'exemple d'un accumulateur

3.4.3 Algorithme BCJR dans le domaine logarithmique

Pour des raisons pratiques, il est souvent plus comode d'exprimer l'algorithme BCJR dans le domaine logarithmique. SI on redéfinit les équations précédentes en considérant le logarithme des quantités précédentes, on se retrouve à devoir dériver les récurrence pour les nouvelles quantités données par

$$\begin{aligned}\tilde{\alpha}_n(s) &\triangleq \log(\alpha_n(s)) \\ \tilde{\beta}_n(s) &\triangleq \log(\beta_n(s)) \\ \tilde{\gamma}_n(s', s) &\triangleq \log(\gamma_n(s', s))\end{aligned}$$

Récursions dans le domaine logarithmique

En utilisant les définitions ci-dessus, après un change changement de variable, il vient

$$\begin{aligned}\tilde{\alpha}_n(s) &\triangleq \log(\alpha_n(s)) \\ &= \log \sum_{s'} \exp(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s)) \\ \tilde{\beta}_{n-1}(s') &\triangleq \log(\beta_n(s')) \\ &= \log \sum_{s} \exp(\tilde{\beta}_n(s) + \tilde{\gamma}_n(s', s))\end{aligned} \tag{3.33}$$

Calcul de $L(u_n)$

$$L(u_n) = \log\Big(\sum_{\mathcal{S}^+} \exp(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s) + \tilde{\beta}_n(s))\Big) - \log\Big(\sum_{\mathcal{S}^-} \exp(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s) + \tilde{\beta}_n(s))\Big)$$

L'opérateur $\max^*$

On définit ici un opérateur qui permettra de réécrire les équations précédentes et et d'en déduire des algorithmes faibles complexité. On définit l'opérateur $\max^*(x, y)$ comme suit :

Définition 31 : *Opérateur* $\max^*(x, y)$

$$\begin{aligned}max^*(x, y) &\triangleq \log(e^x + e^y) \\ &= \max(x, y) - \log(1 + e^{-|x-y|})\end{aligned} \tag{3.34}$$

Cette définition dérive directement du fait que l'on peut écrire $\max(x, y)$ comme suit

$$\max(x, y) = \log\left(\frac{e^x + e^y}{1 + e^{-|x-y|}}\right)$$

De la définition de l'opérateur, on peut déduire la propriété suivante :

Propriétés 9 : *Propriétés de l'opérateur* $\max^*$

$$\begin{aligned} max^*(x, y, z) &\triangleq \log\left(e^x + e^y + e^z\right) \\ &= \overset{*}{\max}(\overset{*}{\max}(x, y), z) \end{aligned} \tag{3.35}$$

cette expression se généralise à un nombre de termes quelconque.

Algorithme Log-MAP

A partir des expressions précédentes, on peut alors donner une version de l'agorithme MAP dans le domaine logarithmique.

Définition 32 : *Algorithme Log-MAP*

1. **Initialisation (avec fermeture treillis) :**

$$\tilde{\alpha}_0(0) = 0 \quad , \quad \alpha_0(s) = -\infty \text{ sinon} \tag{3.36}$$

$$\beta_L(0) = 0 \quad , \quad \beta_L(s) = -\infty \text{ sinon} \tag{3.37}$$

2. **Phase forward-backward :**

$$\forall n = 1, \cdots, L-1,\ \tilde{\alpha}_n(s) = \max_{s'}{}^*(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s)) \tag{3.38}$$

$$\forall n = L-1, \cdots, 1,\ \tilde{\beta}_{n-1}(s') = \max_{s}{}^*(\tilde{\beta}_n(s) + \tilde{\gamma}_n(s', s)) \tag{3.39}$$

3. **Calcul de** $L(u_n)$ **:** $\forall n = 1, \cdots, K$

$$\begin{aligned} L(u_n) &= \max_{\mathcal{S}^+}{}^*\Big(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s) + \tilde{\beta}_n(s)\Big) \\ &\quad - \max_{\mathcal{S}^-}{}^*\Big(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s) + \tilde{\beta}_n(s)\Big) \end{aligned} \tag{3.40}$$

où $\max^*_{\mathcal{S}^x}(.) \triangleq \sum_{\mathcal{S}^x} \exp(.)$

D'un point de vue pratique, cette algorithme est implémenté d'une manière simplifié en utilisant l'opérateur max(.) et utilisation de lookup table pour approximé le terme correctif.

Algorithme Max-Log-MAP

On peut réduire la complexité du Log-MaP en considérant le simple remplacement de l'opérateur max*(.) par l'opérateur max(.). On néglige alors le terme correctif.

Définition 33 : *Algorithme Max-Log-MAP*

1. **Initialisation (avec fermeture treillis) :**

$$\tilde{\alpha}_0(0) = 0 \quad , \quad \alpha_0(s) = -\infty \text{ sinon} \tag{3.41}$$

$$\beta_L(0) = 0 \quad , \quad \beta_L(s) = -\infty \text{ sinon} \tag{3.42}$$

2. **Phase forward-backward :**

$$\forall n = 1, \cdots, L-1, \ \tilde{\alpha}_n(s) = \max_{s'}(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s)) \tag{3.43}$$

$$\forall n = L-1, \cdots, 1, \ \tilde{\beta}_{n-1}(s') = \max_{s}(\tilde{\beta}_n(s) + \tilde{\gamma}_n(s', s)) \tag{3.44}$$

3. **Calcul de $L(u_n)$: $\forall n = 1, \cdots, K$**

$$\begin{aligned} L(u_n) = & \max_{\mathcal{S}^+}\Big(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s) + \tilde{\beta}_n(s)\Big) \\ & -\max_{\mathcal{S}^-}\Big(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s) + \tilde{\beta}_n(s)\Big) \end{aligned} \tag{3.45}$$

Log-MAP (log-BCJR) : cas Gaussien

Dans le cas Gaussien, on peut instancier la métrique de branche comme suit :

$$\begin{aligned} \tilde{\gamma}_n(s', s) &= \log\left(\gamma_n(s', s)\right) \\ &= \log(\pi(u_n)) - n_c/2 * \log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{m=1}^{n_c} |y_n^{(m)} - c_n^{(m)}(s', s)|^2 \end{aligned}$$

en se réferant au critère MAP final, la constante peut-être supprimée.

Chapitre 4

Concaténation de codes en treillis

4.1 Turbo-codes parallèles

On présente ici la concaténation parallèle de codes convolutifs introduite par Claude Berrou en 1993. L'idée est

4.1.1 Structure du codeur

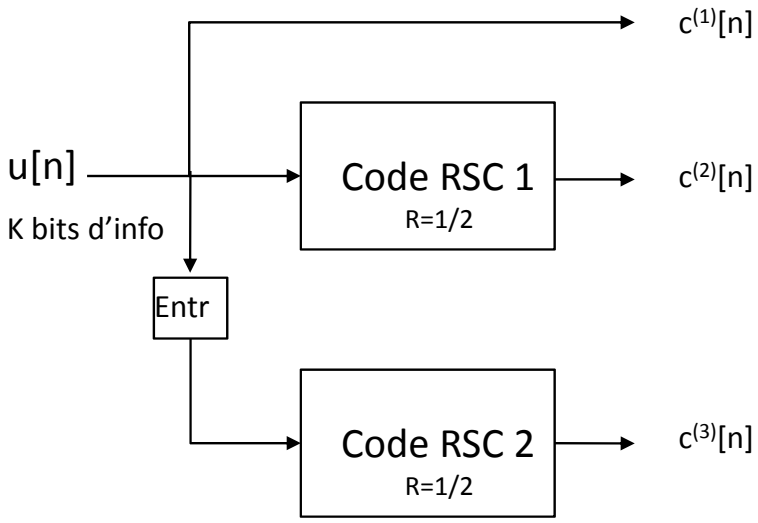


FIGURE 4.1 – Structure d'un turbo code parallèle : chaque code est un code récursif systématique. Ici $R = 1/2$ pour chaque code constituant.

Codage de l'information

La structure d'un turbo-code parallèle avec des codes composants de rendement $R_0 = 1/2$ est donnée figure 4.1. La structure du codeur est une structure de codage par bloc. ELle se généralise à plusieurs codeurs avec des propriétés qui peuvent être différentes. Pour la simplicité d'exposition, on se contentera dans ce texte du cas de deux codeurs identiques ayant les mêmes caractéristiques. La procédure est donnée comme suit

1. K bits d'information sont codés avec le codeur convolutif récursif systématqiue RSC1,

2. Les mêmes K bits d'information sont en parallèle entrelacés et codés par le codeur convolutif récursif RSC2,

3. Comme la sortie systèmatique du code RSC1 est déjà disponible, seule la partie redondance est prise en compte sur le codeur RSC2.

En pratique les deux codes sont souvent identiques, cependant rien n'empêche de considérer des codes composants différents et de rendments différents. La seule contrainte originelle est qu'ils partagent la même information. En partant d'un code de rendement $R_0 = 1/2$, on obtient donc un code de rendement $R'_0 = 1/3$. Le code est par construction systématique. On peut faire varier le rendement en définissant comem pour les codes convolutifs des masques périodiques de poinçonnage. On est donc pas limité au seul rendment 1/3. En générale si on considère un code dont les composants sont de rendements R_1 et R_2 le rendement golbale du code est donnée par

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

Ainsi, on peut atteindre différents rendements par changement des codes composants. On peut également employé la technique de poinçonnage comme utilisée avec les codes convolutifs. Par exemple, on peut envoyé, sur deux prériodes d'horloge, chaque bit systématique et poinçonnés alternativement chaque bits de parité ce qui donne la matrice de poinçonnage de période 2 suivante

$$\mathbf{P} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Le code résultant est un code de rendement $R = 1/2$.

Fermeture de treillis

Généralement, un turbo-code est utilisé pour réalisé un codage de type bloc. Dans ce cadre Une terminaison de treillis est souvent réalisée. Cependant la fermeture de treillis ne se réalise pas par un simple ajout de bits $'0'$ pour amener le codeur dans l'état de registre "tout-à-zéro". Dans le cadre de codes convolutifs récursifs, les ν bits à considérer dépendent de l'état du registre près que les K premiers bits aient été codés. Dans ce cas, la connaissance de cet état permet de déterminer la suite de bits à utiliser. On peut considérer soit une table associée ou de manière plus efficcae de considérer la méthode de Divsalar. Celle-ci consiste en la méthode décrite figure **??**. A près K bits d'information, l'entrée du codeur est alors rebouclée sur le retour du code récursif qui donne alors la sortie systématique et à pour effet de conduire le codeur à l'état "tout-à-zéro". La présence de l'entrelaceur prévient toute fermeture conjointe des deux codeurs avec un nombre de bits raisonnable pour un entrelaceur quelconque. Ainsi, on ne considère donc en générale que la fermeture du premier code composant, le second n'étant pas terminé. On verra par la suite comment gérer dans le décodage cette asymmétrie.

Entrelaceur

L'entrelaceur utilisée dans le codage de type turbo est un entrelaceur en bloc pseudo-aléatoire défini comme une permutation de K éléments (les K bits d'information) sans répétition. Un des rôles principaux est de pouvoir décorréler les entrées souples des décodeurs souples associés. Nous verrons par la suite comment sont réaliser ces entrelaceurs.

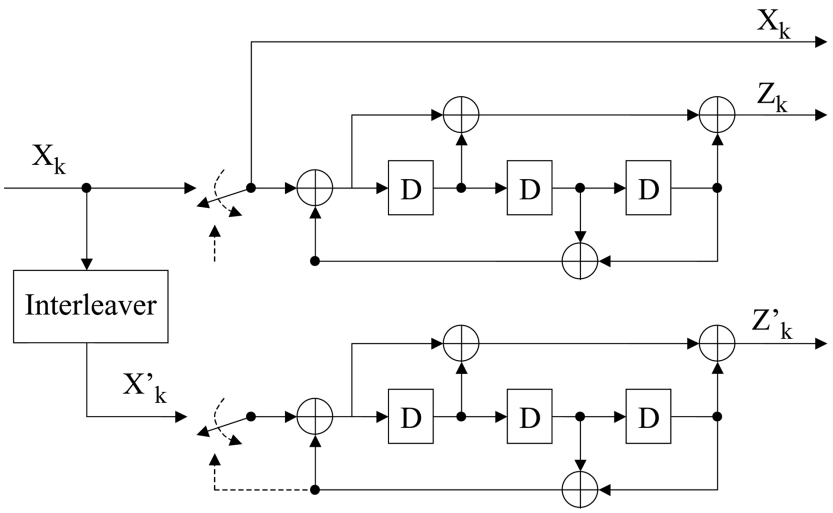


FIGURE 4.2 – Structure d'un turbo code parallèle pour l'UMTS (3GPP)

Exemple :Turbo-code UMTS

4.1.2 Décodage itératif des turbo-codes

Du schéma du codeur, on voit clairement que l'on peut décoder les deux codes constituants de manière indépendantes car on a accès pour les deux codes aux bits systématiques et codés. Cependant, cette stratégie est sous optimale car les bits d'information traités par les deux codeurs ne sont pas indépendants. En effet, les deux codeurs partagent la même information et on peut alors se poser la question d'un décodage conjoint des deux codes convolutifs. Le fait d'avoir un entrelaceur implique une complexité exponentielle du décodage conjoint si l'on voulait considérer un treillis conjoint. L'idée proposée par C. Berrou est de décoder séparément les deux codes constituants. Chaque codeur pourra être à même de renseigner l'autre codeur d'une probabilité a priori pour les bits d'information qu'ils ont en commun, a priori qui pourra alors être mis à profis pour une nouvelle tentative de décodage. On peut alors envisager une sorte de décodage successif collaboratif des deux décodeurs. La difficulté est d'assurer que l'information échangée est suffisament décoréllée entre les deux décodeurs. Pour cela nous avons besoin de définir une quantité appelée *information extinsèque* qui est une quantité qui est liée au gain statistique apporté par le décodeur sur le décodage souple d'un bit hormis la vraisemblance et l'a priori disponible.

Notion d'Information Extrinsèque

On se contentera ici du cas d'un codeur récursif systématique de rendement $R = 1/n_c$, pour simplifier les notations. On supposera également que $c_n^{(1)} = u_n$, ie. le premier bit codé par le codeur RSC1 est le bit systématique. On peut alors à partir du décodage MAP symbole mettre en exergue l'information dîte extinsèque :

Définition 34 : *Information extrinsèque*

Pour un codeur RSC, le LLR associé à un décodage de type MAP peut être décomposé de la manière suivante :

$$\begin{aligned} L(u_n) &= \log\left[\frac{\sum_{\mathcal{S}^+} p(s_{n-1}=s', s_n=s, \mathbf{y})}{\sum_{\mathcal{S}^-} p(s_{n-1}=s', s_n=s, \mathbf{y})}\right] && (4.1)\\ &= \mathbf{L_c(u_n)} + \mathbf{L_a(u_n)} + \mathbf{L_{ext}(u_n)} && (4.2)\end{aligned}$$

avec

1. **Information canal (Vraisemblance) :**

$$L_c(u_n) = \log\left(\frac{p(y_n^{(1)}|u_n=+1)}{p(y_n^{(1)}|u_n=-1)}\right)$$

2. **Information a priori :**

$$L_a(u_n) = \log\left(\frac{p(u_n=+1)}{p(u_n=-1)}\right)$$

3. **Information extrinsèque :**

$$\begin{aligned} L_{ext}(u_n) &= \log\left[\frac{\sum_{\mathcal{S}^+} \alpha_{n-1}(s')\prod_{k=2}^{n_c} p(y_n^{(k)}|c_n^{(k)}(s',s))\beta_n(s)}{\sum_{\mathcal{S}^-} \alpha_{n-1}(s')\prod_{l=2}^{n_c} p(y_n^{(l)}|c_n^{(l)}(s',s))\beta_n(s)}\right]\\ &= \log\left[\frac{\sum_{\mathcal{S}^+} \alpha_{n-1}(s')\gamma_n^e(s',s)\beta_n(s)}{\sum_{\mathcal{S}^-} \alpha_{n-1}(s')\gamma_n^e(s',s)\beta_n(s)}\right] && (4.3)\end{aligned}$$

On peut alors faire l'interprétation suivante. En non codée, nous aurions

$$\begin{aligned} L(u_n) &= \mathbf{L_c(u_n)} + \mathbf{L_a(u_n)}\\ &= \log\left(\frac{p(y_n^{(1)}|u_n=+1)}{p(y_n^{(1)}|u_n=-1)}\right) + \log\left(\frac{p(u_n=+1)}{p(u_n=-1)}\right) && (4.4)\end{aligned}$$

Ce qui correspond à la statistique de test classique de la détection d'une variable binaire en fonction de sa vraisemblance et de l'a priori disponible. Pour le cas codé, la quantité $L_{ext}(u_n)$ représente un terme supplémentaire, qui est lié au code et au processus de décodage. Elle est extrinsèque car ne dépend pas de l'observations directe du bit ni de son apriori, mais de toutes les autres données. C'est cette quantité qui après décodage MAP par l'agorithme BCJR permet d'avoir un gain par rapport au cas non codé. C'est cette quantité qui sera utilisé à la sortie d'un des deux décodeurs comme a priori pour l'autre décodeur.

Le décodage MAP d'un code convolutif peut donc se représenter par un bloc de décodage à entrées et sorties souples (bloc SISO). Le schéma bloc est donné ci-après.

Décodage itératif MAP

le schéma en figure 4.3 donne la structure du décodeur itératif turbo. Le principe sous-jacent est le décodage distribué des deux codes composants qui vont avoir un décodage alterné. Les décodeurs MAP des deux codes constituants vont alors pouvoir s'échanger une information *extrinsèque* relative aux bits d'information communs de manière itérative. L'information extrinsèque sur chaque bit en sortie de chaque codeur est alors considérée par l'autre décodeur comme un a priori après entrelacement/désentrelacement, l'entrelacement favorisant la décorrélation de ces quantités. Le décodage s'effectue comme suit. Un premier décodage MAP est effectué sur le premier code RSC1 pour lequel on ne considère pas d'apriori (ie. $L_{a,0}(u[n]) = 0, \forall n$). L'information extrinsèque du premier codeur est alors calculée comme

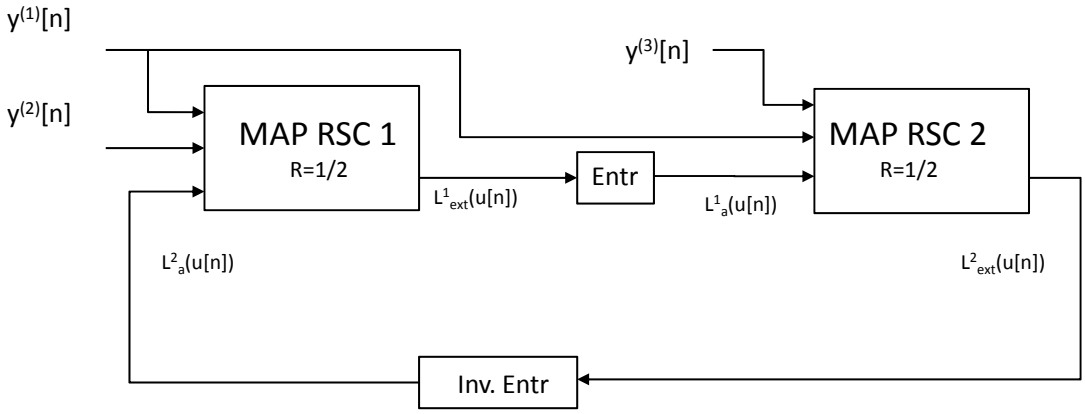


FIGURE 4.3 – Schéma de décodage itératif des turbo-codes parallèle

FIGURE 4.4 – Treillis du code (7,5)

$$L^1_{\text{ext},1}(u[n]) = L^1_1(u[n]) - L_c(u[n]) - L^1_{a,0}(u[n])$$

Le décodage MAP complet est alors donné par la procédure générique suivante :
— Critères MAP au deux décodeurs à l'itération (ℓ) :

$$\begin{aligned} L^{(\ell)}_{RSC_1}(u_n) &= \frac{2}{\sigma^2} y^{(1)}_n + L^{(\ell-1)}_{a,1}(u_n) + L^{(\ell)}_{e,1}(u_n) \\ &= \frac{2}{\sigma^2} y^{(1)}_n + L^{(\ell-1)}_{e,2}(u_{\pi^{-1}(n)}) + L^{(\ell)}_{e,1}(u_n) \end{aligned} \tag{4.5}$$

$$L^{(\ell)}_{RSC_2}(u_{\pi(n)}) = \frac{2}{\sigma^2} \pi(y^{(1)}_{\pi(n)}) + L^{(\ell-1)}_{e,1}(u_{\pi(n)}) + L^{(\ell)}_{e,2}(u_{\pi(n)}) \tag{4.6}$$

avec $\pi(.)$ et π^{-1} représentent les opérations d'entrelacement et de désentrelacement

4.2 Turbo-codes série

La figure 4.5 représente une autre concaténation possible à savoir la concaténtaion série. Le processus à l'émetteur est le suivant

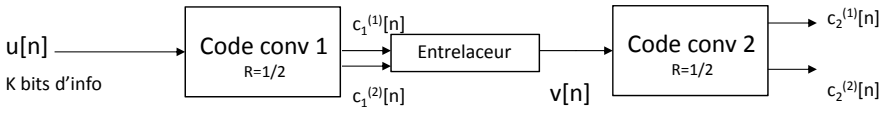


FIGURE 4.5 – Structure d'un turbo-code série : le code 1 fournit le bloc d'information au code 2.

FIGURE 4.6 – Structure du décodeur d'un turbo-code série.

- K bits d'information sont codés avec le codeur 1 (code externe),
- les bits codés sont entrelacés et puis codés par le codeur 2 (code interne).

Dans ce cadre, on a un rendement global (hors poinçonnage) correspondant au produit des rendements des deux codes internes

$$R = R_i R_o$$

où R_i (resp. R_o) est le rendement du code interne (resp. externe).

L'architecture du décodeur est alors donnée par la figure 4.6. Les principales différences par rapport au cas de la concaténation parallèle sont les suivantes :

- les deux décodeurs associés différents du cas parallèle,
- le premier codeur peut le pas être récursif.

Pour le décodage, le décodeur souple associé au code interne est le même que celui d'un turbo-code parallèle, ie. que l'on donne une information souple sur les bits d'"information" du codeur interne. Pour le code interne, le décodeur doit fournir une information extrinsèque sur l'ensemble des bits codés et non plus seulement sur les bits d'information. Le critère MAP associé est alors modifié de la manière suivante

$$\begin{aligned} L_1(c_n^{(m)}) &= \max_{\mathcal{C}^+}{}^{*}\Big(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s',s) + \tilde{\beta}_n(s)\Big) \\ &\quad -\max_{\mathcal{C}^-}{}^{*}\Big(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s',s) + \tilde{\beta}_n(s)\Big) \qquad (4.7) \\ &= L_2^e(c_n^{(m)}) + L_1^e(c_n^{(m)}) \qquad (4.8) \\ \tilde{\gamma}_n(s',s) &= \sum_{m=1}^{n_c} c_n^{(m)}(s',s)\frac{L_2^e(c_n^{(m)}(s',s))}{2} \qquad (4.9) \end{aligned}$$

où l'on a les notations suivantes :

$$\mathcal{C}^+ = \{(s',s) \text{ où } (s_{n-1}=s') \mapsto (s_n = s) | c_n^{(m)} = +1\}$$

$$\mathcal{C}^- = \{(s',s) \text{ où } (s_{n-1}=s') \mapsto (s_n = s) | c_n^{(m)} = -1\}$$

Chapitre 5

Analyse asymptotique : EXIT charts

Le but de ce chapitre est d'intriduire l'analyse de performance des schéma turbo dans le cadre dît asymptotique, ie. les performances de ces schémas quand la raille des mots de codes tend vers l'infini. Le but est de prévoir ce que l'on nomme le seuil de décodage de ces schémas à savoir le point d'opération en E_b/N_0 minimum pour lequel on peut garantir une convergence du schéma itératif vers une propabilité d'erreur arbitrairement petite. En dessous de ce seuil, la probabilité d'erreur sera alors nécessairement non nulle. Ce type d'analyse permet dès lors de prédire les performances du schéma de sélectionner/optimiser les schémas de codage basés sur des concaténations parallèles ou série. De la même façon on pourra alors prédire les performances en fonction du rapport signal à bruit.

5.0.1 Présentation générale

Une courbe EXIT (*EXIT chart*) sert à analyser le comportement entrée-sortie d'un bloc à entrées et sorties souples (SISO). On cherche ainsi à déterminer une relation entrée-sortie pour caractériser un bloc de décodage suivant une figure de mérite donnée. Le bloc opérant sur des entrées homogènes à des probabilités/vraisemblance ou versions log, on caractérisera donc se comportement *en moyenne*. Dans le cas générale, un bloc SISO dépend en entrées du canal (via les observations issues du canal) et de l'a priori (donné par les informations extrinsèques d'un autre bloc SISO). En sortie, on a alors deux sorties, l'une liée à l'information extinsèque, l'autre liée à information a posteriori. Le but d'une courbe EXIT est de modéliser de manière indépendante chaque bloc SISO de la chaîne. Pour ce faire si on peut *mesurer* un certaine quantité en sortie en fonction des paramètres moyens d'entrée, on se doit de définir un modèle statistique a priori pour l'information extrinsèque en entrée du bloc SISO, les données observées et donc simulées suivant elles le modèle de canal prédéfini.

Pour simplifier l'analyse et dans certains cas la mise en équations des processus de décodage, on voudra caractériser chaque bloc par une relation entrée-sortie mono-dimensionnelle de type Output $= F(\text{SNR}, \text{Input})$. La figure de mérite Input en entrée sera associée aux informations a priori entrantes (extrinsèques des autres blocs), de même, la figure de mérite sortante Output sera associée aux informations extrinsèques fournies par le bloc SIS0