TS226 - Codes convolutifs et codes concaténés associés.

Romain Tajan

8 octobre 2018

Table des matières

- Chapitre 1 -

Introduction

Contents

1.1 Notations

Dans ce cours, nous essayerons de garder les notations suivantes :
— Les scalaires sont représentés en minuscules : x
— Les vecteurs sont représentés en en gras : $\mathbf{x}$
— Soit un vecteur $\mathbf{x} = (x_0, x_1, \dots x_{n-1})$, le vecteur $\mathbf{x}_i^j$ représente le vecteur extrait $\mathbf{x}_i^j = (x_i, x_{i+1}, \dots x_j)$
— les variables aléatoires sont représentées en majuscules : X,
— les vecteurs aléatoires sont représentées en gras et en majuscules : $\mathbf{X}$,
— les ensembles seront notés : $\mathcal{X}, \mathcal{Y}$

1.2 Rappels et définitions

1.2.1 Canal et chaîne de communication

La chaîne de communication considérée dans ce cours est celle présentée en Figure 1.1. On souhaite transmettre un message W choisi dans un ensemble de M messages $\{0, \dots M-1\}$. Le message est d'abord transformé en un signal $\mathbf{X}(W)$ qui sera transmis dans le canal. Le signal reçu $\mathbf{Y}$ est une version dégradée

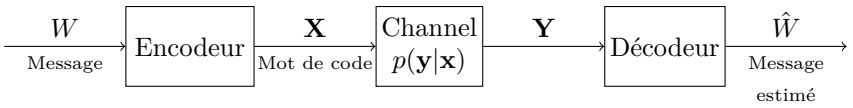

Figure 1.1 – *Chaîne de communication*

(déformée, bruitée ...) de $\mathbf{X}$. Cette dégradation est décrite par la probabilité conditionnelle $p(\mathbf{y}|\mathbf{x})$. Le récepteur estime W par une règle qui lui est propre : $\hat{W} = g(\mathbf{Y})$.

Dans ce cours, un canal sera représenté par sa probabilité de transition. Ainsi, un canal à entrées dans $\mathcal{X}$ et à sorties dans $\mathcal{Y}$ seras représenté par la donnée des probabilités conditionnelles $p(y|x)$ pour $x \in \mathcal{X}$ et $y \in \mathcal{Y}$. Ce canal sera noté $(\mathcal{X}, \mathcal{Y}, p(y|x))$.

Pour un entier n, considérons le canal $(\mathcal{X}^n, \mathcal{Y}^n, p(\mathbf{y}|\mathbf{x}))$. Ce canal est dit **sans mémoire** si sa probabilité de transition vérifie

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=0}^{n-1} p(y_i|x_i). \tag{1.1}$$

Le canal $(\mathcal{X}^n, \mathcal{Y}^n, p(\mathbf{y}|\mathbf{x}))$ est appelé **extension d'ordre** n du canal sans mémoire $(\mathcal{X}, \mathcal{Y}, p(y|x))$. Dans ce cours, nous n'aborderons que les canaux sans mémoire. Soit un canal sans mémoire $(\mathcal{X}^n, \mathcal{Y}^n, p(\mathbf{y}|\mathbf{x}))$, si de plus $\mathcal{X}$ et $\mathcal{Y}$ sont des ensembles finis, ce canal est dit canal discret sans mémoire (DMC : Discrete Memoryless Channel).

1.2.2 Code (M,n)

Définissons maintenant les concepts tels que **code**, **message**, **encodeur**, **mot de code**, **décodeur** et **rendement**. Un code (M, n) pour le canal $(\mathcal{X}^n, \mathcal{Y}^n, p(\mathbf{y}|\mathbf{x}))$ est composé de 3 éléments :

1. Un ensemble de M **messages** indexés. On les notera cet ensemble $\mathcal{M} = \{0, 1, \ldots, M-1\}$.
2. Un **encodeur** : il s'agit d'une fonction de l'ensemble des messages $\mathcal{M}$ vers $\mathcal{X}^n$. On notera $\mathbf{X}$ cette fonction d'encodage :

$$\begin{aligned} \mathbf{X} \quad : \quad & \mathcal{M} \to \mathcal{X}^n \\ & W \mapsto \mathbf{X}(W) \end{aligned}$$

   Les vecteurs $\mathbf{x}(0)$, $\mathbf{x}(1)$, ..., $\mathbf{x}(M-1)$ sont appelés **mots de code**.
3. Un **décodeur** : il s'agit de d'une fonction de $\mathcal{Y}^n$ vers $\mathcal{M}$

$$\begin{aligned} g \quad : \quad & \mathcal{Y}^n \to \mathcal{M}. \\ & \mathbf{Y} \mapsto \hat{W} = g(\mathbf{Y}) \end{aligned}$$

1.2.3 Probabilité d'erreur

Supposons que le message envoyé soit le message $w \in \mathcal{M}$ $(W = w)$. Si le message estimé $\hat{W}$ est différent de w, on dira que le récepteur comment une

erreur. Formellement, cet évènement est caractérisé par sa probabilité :

$$\lambda_w = \mathbb{P}\left[g(\mathbf{Y}) \neq w | \mathbf{X} = \mathbf{x}(w)\right]. \tag{1.2}$$

On introduit alors deux caractérisations de la probabilité d'erreur :

— La **probabilité d'erreur paquet maximale** définie comme :

$$\lambda^{(n)} = \max_{w \in \mathcal{M}} \lambda_w \tag{1.3}$$

Cette probabilité représente un "pire cas", en effet, elle représente la probabilité d'erreur maximale obtenue parmi tous les mots de code.

— Une autre probabilité d'erreur couramment utilisée est la **probabilité d'erreur paquet moyenne**(arithmétique) ; elle est définie comme

$$P_e^{(n)} = \frac{1}{M} \sum_{w=0}^{M-1} \lambda_w. \tag{1.4}$$

Cette probabilité représente la probabilité d'erreur moyenne lorsque W est uniformément distribuée sur $\mathcal{M}$. Dans tout ce cours nous considèrerons que W est uniformément distribué sur $\mathcal{M}$. Il convient alors de remarquer que $P_e^{(n)} \leq \lambda^{(n)}$

Les deux probabilités d'erreur précédentes sont des probabilités d'erreur "paquet" car elles concernent la probabilité que le message estimé soit différent du message émis. Dans le cas où les messages sont représentés par des vecteurs binaires, une erreur paquet n'implique pas forcément que tous les bits du message sont faux.

Afin de simplifier cette partie, nous supposerons qu'il existe K tel que $M = 2^K$ et que chaque message $w \in \mathcal{M}$ est représenté par une étiquette binaire $\mathbf{u}(w) = (u_0(w), \dots u_{K-1}(w))$. La **probabilité d'erreur binaire moyenne** P_b est mathématiquement définie comme

$$P_b = \frac{1}{K} \sum_{i=0}^{K-1} \mathbb{P}\left(u_i(g(\mathbf{Y})) \neq u_i(W)\right) \tag{1.5}$$

L'hypothèse que $\mathbb{P}\left(u_i(g(\mathbf{Y})) \neq u_i(W)\right)$ ne dépend pas de i est souvent vérifiée. Dans ce cas, P_b est donnée par

$$P_b = \mathbb{P}\left(u_0(g(\mathbf{Y})) \neq u_0(W)\right) \tag{1.6}$$

1.2.4 Rendement de code

Le **rendement du code** d'un code (M, n) est défini comme le rapport suivant :

$$R = \frac{\log_2(M)}{n}. \tag{1.7}$$

On peut remarquer ici que $\log_2(M)$ représente le nombre de bits moyens nécessaires pour représenter tous les mots de code. R peut alors être compris comme le nombre de bits de message transmis par symbole envoyé.

1.2.5 Capacité

Nous nous intéressons maintenant à la notion de **capacité** du canal sans mémoire $(\mathcal{X}, \mathcal{Y}, p(y|x))$. Un rendement est dit **atteignable** si il existe une séquence de codes $(\lceil 2^{nR} \rceil, n)$ tels que $\lambda_m^{(n)} \to 0$ lorsque $n \to +\infty$. La **capacité** d'un canal est alors définie comme le supremum des rendements atteignables. Shannon relie la capacité du canal sans mémoire $(\mathcal{X}, \mathcal{Y}, p(y|x))$ au maximum de l'information mutuelle

$$C = \max_{p(x)} I(X;Y) \tag{1.8}$$

en montrant le théorème suivant

Theorem 1.1. *Pour un canal sans mémoire, tout les rendements R strictement inférieurs à C sont atteignables. Précisément, pour tout rendement $R < C$, il existe une séquence de codes $(2^{nR}, n)$ telle que $\lambda^{(n)} \to 0$.*

Réciproquement, toute séquence de codes $(2^{nR}, n)$ telle que $\lambda^{(n)} \to 0$ vérifie $R \leq C$.

La capacité d'un canal est utile car elle est un indicateur du débit maximal qui peut être transmis par ce canal.

1.3 Canaux classiques

Dans ce cours, nous allons considérer principalement trois sortes de canaux différents : le Canal Binaire à Effacement (BEC : Binary Errasure Channel), le Canal Binaire Symétrique (BSC : Binary Symetric Channel) et le Canal Additif à Bruit Blanc Gaussien (AWGN : Additive White Gaussian Noise).

1.3.1 Canal Binaire à Effacement

Le canal BEC est le plus simple au sens où il introduit des erreurs par l'intermédiaire d'un symbole d'erreur dont on connaît les positions. Son entrée est binaire $X = \{0, 1\}$ et sa sortie peut prendre 3 valeurs $Y = \{0, 1, \epsilon\}$ où ϵ est un symbole d'erreur. La probabilité de commettre une erreur est notée p et est définie comme $p = p(\epsilon|0) = p(\epsilon|1)$.

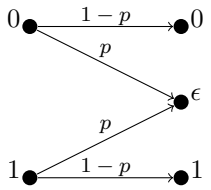

Figure 1.2 – *Canal binaire à effacement*

La capacité du canal BEC est la suivante

$$C_{BEC} = 1 - p, \tag{1.9}$$

elle est obtenue pour une distribution d'entrée uniforme.

1.3.2 Canal Binaire Symétrique

Le canal BSC est à entrées binaires $X = \{0, 1\}$ et à sorties binaires $Y = \{0, 1\}$ et est défini grace à un paramètre p étant la probabilité de transition $p(1|0) = p(0|1) = p$.

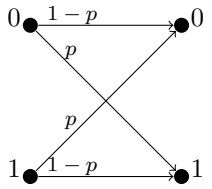

Figure 1.3 – *Canal binaire symétrique*

La capacité du canal BSC est la suivante

$$\begin{aligned} C_{BSC} &= 1 + p\log_2(p) + (1-p)\log_2(1-p) \\ &= 1 - H(p) \end{aligned} \tag{1.10}$$

elle est obtenue pour une distribution d'entrée uniforme.

1.3.3 Canal Additif à Bruit Blanc Gaussien

Classiquement, le canal Additif à Bruit Blanc Gaussien réel (AWGN : additive White Gaussian Noise) est tel que

$$Y = X + Z$$

où $Z \sim \mathcal{N}(0, \sigma_Z^2)$. Ainsi, $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$ et la probabilité de transition $p(y|x)$ est donnée par l'équation suivante

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma_Z^2}} e^{\frac{-1}{2\sigma_Z^2}(y-x)^2}. \tag{1.11}$$

La capacité du canal AWGN avec la contrainte de puissance $\mathbb{E}\left[X^2\right] \leq \sigma_X^2$ est

$$C_{AWGN} = \frac{1}{2}\log_2\left(1 + \frac{\sigma_X^2}{\sigma_Z^2}\right). \tag{1.12}$$

Elle est obtenue pour la distribution gaussienne $X \sim \mathcal{N}(0, \sigma_X^2)$.

Dans ce cours, nous allons nous intéresser au canal AWGN à "entrée binaire" noté BI-AWGN. Ce canal est tel que $\mathcal{X} = \{-1, 1\}$, $\mathcal{Y}$ et $p(y|x)$ restant inchangés. La capacité du canal BI-AWGN est obtenue pour une entrée uniforme et est donnée par

$$C_{BI-AWGN} = -\int_{-\infty}^{+\infty} p(y)\log_2\left(p(y)\right)dy - \frac{1}{2}\log_2\left(2\pi e\sigma_Z^2\right) \tag{1.13}$$

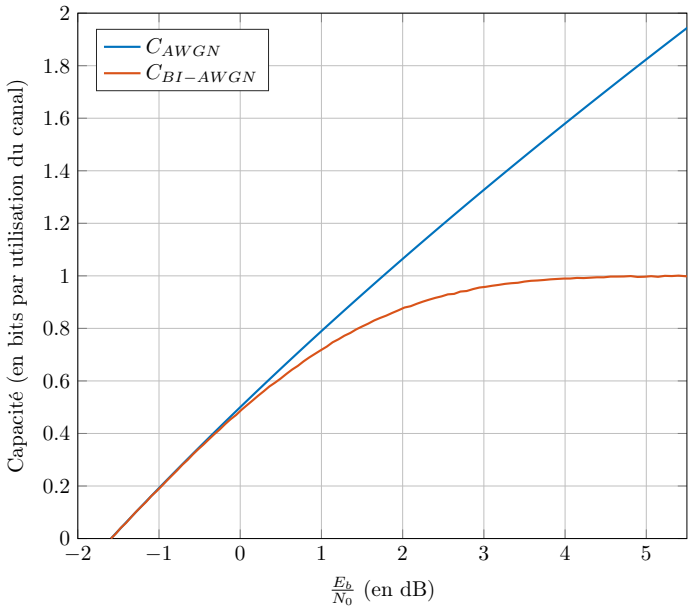

Figure 1.4 – *C_{AWGN} et $C_{BI-AWGN}$ en fonction de $\frac{E_b}{N_0}$*

1.4 Remarques concernant le décodage

1.4.1 Décodeur MAP "séquence", décodeur ML "séquence"

Dans cette section, nous nous intéressons au décodage du Maximum A Posteriori (MAP) pour les canaux BSC et AWGN sans mémoire. Le premier décodeur considéré est le décodeur du Maximum A Posteriori (MAP) séquence. Ce décodeur est considéré car c'est celui qui permet de minimiser $P_e^{(n)}$. Mathématiquement, le critère de décodage MAP s'écrit comme

$$\hat{w} = \arg\max_{w \in \mathcal{M}} p(\mathbf{x}(w)|\mathbf{y}). \tag{1.14}$$

Autrement dit, décoder en utilisant le critère MAP revient à choisir le mot de code le plus probable sachant le signal reçu. En utilisant la règle de Bayes, nous avons

$$\hat{w} = \arg\max_{w \in \mathcal{M}} p(\mathbf{y}|\mathbf{x}(w)) \frac{p(\mathbf{y})}{p(\mathbf{x}(w))} \tag{1.15}$$

Si de plus les mots de codes sont tirés avec une probabilité uniforme : $p(\mathbf{x}(w)) = 1/M$, en utilisant le fait que $p(\mathbf{y})$ ne dépends pas de w nous obtenons

$$\hat{w} = \arg\max_{w \in \mathcal{M}} p(\mathbf{y}|\mathbf{x}(w)). \tag{1.16}$$

Ce critère s'appelle le critère du maximum de vraisemblance (ML) séquence. Le terme séquence signifie que l'on cherche à décoder tout le mot de code en "un coup".

Cas du canal BSC

Considérons dans un premier temps, le canal BSC de probabilité de transition p. Le décodage ML séquence d'un code (M, n) s'écrit

$$\hat{w} = \arg\max_{w\in\mathcal{M}} p(\mathbf{y}|\mathbf{x}(w)) \tag{1.17}$$

$$= \arg\max_{w\in\mathcal{M}} \prod_{i=0}^{n-1} p(y_i|x_i(w)) \tag{1.18}$$

L'équation (1.18) est due au fait que le canal considéré est sans mémoire. En utilisant la monotonie de la fonction log, on a

$$\hat{w} = \arg\max_{w\in\mathcal{M}} \log\left(\prod_{i=0}^{n-1} p(y_i|x_i(w))\right) \tag{1.19}$$

$$= \arg\max_{w\in\mathcal{M}} \sum_{i=0}^{n-1} \log\left(p(y_i|x_i(w))\right) \tag{1.20}$$

On remarque que $p(y_i|x_i(w)) = p$ pour tous les cas où $y_i \neq x_i(w)$ et $p(y_i|x_i(w)) = 1-p$ pour tous les cas où $y_i = x_i(w)$. En utilisant la distance de Hamming définie pour deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ par

$$d_H(\mathbf{x}, \mathbf{y}) = \text{Nombre de positions où } \mathbf{x} \text{ diffère de } \mathbf{y}, \tag{1.21}$$

nous réécrivons l'équation (1.20) comme

$$\hat{w} = \arg\max_{w\in\mathcal{M}} d_H(\mathbf{y}, \mathbf{x}(w)) \log(p) + (n - d_H(\mathbf{y}, \mathbf{x}(w))) \log(1-p) \tag{1.22}$$

$$= \arg\max_{w\in\mathcal{M}} d_H(\mathbf{y}, \mathbf{x}(w)) \log\left(\frac{p}{1-p}\right) + n\log(1-p) \tag{1.23}$$

Dans les cas qui nous intéressent $p \leq 0.5$, donc $1-p \geq p$ et $\log\left(\frac{p}{1-p}\right) < 0$. Le terme $n\log(1-p)$ est constant, on obtient finalement

$$\hat{w} = \arg\min_{w\in\mathcal{M}} d_H(\mathbf{y}, \mathbf{x}(w)) \tag{1.24}$$

En conclusion, dans le cas d'un canal BSC sans mémoire, le récepteur ML séquence est équivalent à trouver le mot de code le plus proche du signal reçu en utilisant la distance de Hamming.

Cas du canal Gaussien

Intéressons nous maintenant au canal AWGN de variance σ^2. Le décodage ML séquence d'un code (M, n) s'écrit toujours en utilisant l'équation (1.20) car nous n'avons utilisé que la monotonie du log et l'absence de mémoire du canal

$$\hat{w} = \arg\max_{w\in\mathcal{M}} \sum_{i=0}^{n-1} \log\left(p(y_i|x_i(w))\right)$$

On rappelle que pour un canal AWGN de variance σ^2, $p(y_i|x_i(w))$ est donné par l'équation (1.11) rappelée ici par souci de clarté

$$p(y_i|x_i(w)) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - x_i)^2}{2\sigma^2}}.$$

En substituant $p(y_i|x_i(w))$ dans l'équation (1.20), on a

$$\begin{aligned}\hat{w} &= \arg\max_{w\in\mathcal{M}} -\frac{1}{2\sigma^2}\sum_{i=0}^{n-1}\left(y_i - x_i(w)\right)^2 - n\log(\sqrt{2\pi\sigma^2}) \\ &= \arg\min_{w\in\mathcal{M}} \sum_{i=0}^{n-1}\left(y_i - x_i(w)\right)^2 \end{aligned} \tag{1.25}$$

L'équation (1.25) est obtenue en remarquant que le terme $-n\log(\sqrt{2\pi\sigma^2})$ est constant et que le terme $-\frac{1}{2\sigma^2}$ est constant et négatif. En introduisant la distance euclidienne $d_E(\mathbf{x},\mathbf{y})$

$$d_E(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=0}^{n-1}(x_i - y_i)^2}$$

on obtient finalement

$$\hat{w} = \arg\min_{w\in\mathcal{M}} d_E(\mathbf{y},\mathbf{x}(w))^2. \tag{1.26}$$

En conclusion, dans le cas d'un canal AWGN sans mémoire, le récepteur ML séquence est équivalent à trouver le mot de code le plus proche du signal reçu en utilisant la distance Euclidienne.

1.4.2 Décodeur MAP "bit"

Le décodeur MAP "bit" s'écrit

$$\hat{u}_i = \arg\max_{u_i\in\{0,1\}} p(u_i|\mathbf{y}) \tag{1.27}$$

1.5 Historique

Le codage de canal a été introduit en 1950 par Claude Elwood Shannon alors qu'il travaille pour les Bell Laoratories. L'idée sous-jacente au codage de canal est de rajouter de la redondance sur l'information transmise afin de la protéger contre des erreurs introduites par le canal de propagation.

- Chapitre 2 -

Codes convolutifs

Contents

Les codes convolutifs ont été introduits par Peter Elias en 1955. Il ont été utilisés dans nombre de standards de télécommunication depuis les années 1970. On les retrouve en particulier dans les normes de communications avec les satellites voyager et Digital Video Broadcasting (DVB-S2). Dans ces normes, les codes codes convolutifs sont souvent associés à un autre code comme le code Reed Solomon dans un schéma appelé concaténé.

Depuis les années 1990, les codes convolutifs sont utilisés comme brique de base dans les turbocodes. Ces turbocodes sont considérés dans des standards tels que le standard de 4e génération "Long Term Evolution" (LTE).

2.1 Présentation

2.1.1 Code $(7,5)_8$

Les codes convolutifs sont des codes linéaires qui permettent d'encoder les bits "à la volée" à partir de registres à décalage. Le code convolutif le plus classiquement présenté s'appelle code $(7,5)_8$; l'architecture de ce code est donnée en Figure 2.1. Toutes les sommes présentes dans la Figure 2.1 sont à comprendre dans $\mathbb{F}_2$ (corps de Galois de 2 éléments) ; autrement dit, les additions sont à réaliser "modulo 2".

Soit $\mathbf{u} = (u_0, u_1, \dots u_{K-1})$ la séquence binaire à transmettre. Les sorties $\mathbf{c}^{(1)}$ et $\mathbf{c}^{(2)}$ de ce code convolutif sont obtenues par filtrage de la séquence $\mathbf{u}$ par les filtres de réponses impulsionnelles $\mathbf{g}^{(1)} = (1,1,1)$ et $\mathbf{g}^{(2)} = (1,0,1)$.

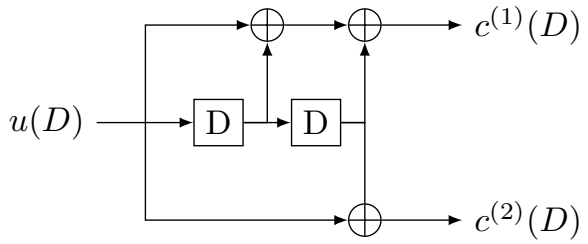

Figure 2.1 – *Code convolutif utilisant un encodeur non-récursif non-systématique* $(7,5)_8$

Plus généralement, la sortie $\mathbf{c}^{(i)}$ d'un code convolutif est obtenue comme

$$\mathbf{c}^{(i)} = \mathbf{u} \star \mathbf{g}^{(i)}, \tag{2.1}$$

$$c_n^{(i)} = \sum_{l=0}^{\nu-1} g_l^{(i)} u_{n-l}, \tag{2.2}$$

où toutes les valeurs contenues de $\mathbf{g}^{(i)}$ sont binaires et toutes les opérations sont à réaliser dans $\mathbb{F}_2$.

Dans le domaine de la transformée en D, l'équation (2.2) devient

$$C^{(i)}(D) = U(D)G^{(i)}(D). \tag{2.3}$$

Il est à noter que les coefficients des polynômes présents dans l'équation (2.3) sont à valeur dans $\mathbb{F}_2$. De façon encore plus compacte, on peut utiliser la notation matricielle suivante :

$$\mathbf{C}(D) = U(D)\mathbf{G}(D), \tag{2.4}$$

où $\mathbf{C}(D) = \begin{bmatrix} C^{(1)}(D) & C^{(2)}(D) \end{bmatrix}$ et $\mathbf{G}(D) = \begin{bmatrix} G^{(1)}(D) & G^{(2)}(D) \end{bmatrix}$. Ce code possède un rendement $R = 1/2$ car pour K bits encodés, il possède 2^K mots de codes et tous ces mots de codes sont de longueur $N = 2K$.

2.1.2 Encodeurs récursifs

On introduit les encodeurs **récursifs** en généralisant $G^{(i)}(D)$ aux fractions rationnelles :

$$G^{(i)}(D) = \frac{A^{(i)}(D)}{B^{(i)}(D)} \tag{2.5}$$

$$= \frac{a_0^{(i)} + a_1^{(i)} D + \cdots + a_m^{(i)} D^m}{1 + b_1^{(i)} D + \cdots + b_m^{(i)} D^m} \tag{2.6}$$

avec $b_0 = 1$. Dans la représentation octale des codes convolutifs, on écrira $(\frac{A^{(1)}}{B^{(1)}}, \frac{A^{(2)}}{B^{(2)}})_8$. Le code convolutif $(1, 5/7)_8$ correspond aux fractions rationnelles suivantes

$$G^{(1)}(D) = 1$$

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

son implémentation est donnée en Figure 2.2.

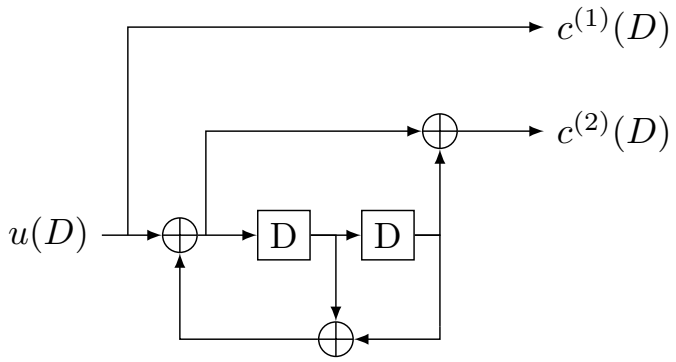

Figure 2.2 – *Code convolutif utilisant un encodeur récursif systématique* $(1, \frac{5}{7})_8$

2.1.3 Classification des encodeurs

Dans ce cours on distinguera les classes d'encodeurs suivantes :

— **récursif** (resp. **non-récursif**) : un encodeur est dit récursif si il possède une voie de retour. Dans l'expression de la transformée en D, ceci se traduit par l'apparition d'un dénominateur non 1 dans la fonction de transfert de l'encodeur.

— **systématique** (resp. **non-systématique**) : un encodeur est dit systématique s'il possède une sortie identique à l'entrée.

Les encodeurs pourront donc appartenir aux 4 catégories suivantes : récursif systématiques, récursifs non-systématiques, non-récursifs systématique et non-récursif non-systématiques.

Les encodeurs récursifs systématiques seront les briques de bases des turbo-codes, ils seront donc ceux que nous allons étudier dans la suite de ce cours. Le code $(1, 5/7)_8$ sera celui considéré comme illustration, néanmoins, les résultats des sections suivantes peuvent être aisément étendus.

2.1.4 Représentations graphiques des codes convolutifs

Il existe plusieurs façons de représenter les codes convolutifs. Dans ce cours, nous présentons deux représentations apparaissant comme étant dexu des plus utilisées :

— le diagramme d'états,

— le treillis.

Représentation en machine à états

L'état d'un encodeur s_n est défini comme le contenu de ses registres à décalages.

$$\begin{cases} \mathbf{c}_n = F_1(s_n, u_n) \\ s_{n+1} = F_2(s_n, u_n) \end{cases} \tag{2.7}$$

Les fonctions F_1 et F_2 peuvent être représentées synthétiquement sous la forme d'un *diagramme d'états*. Le diagramme d'états pour l'encodeur $(1, \frac{5}{7})_8$ est donné en exemple dans la Figure 2.3.

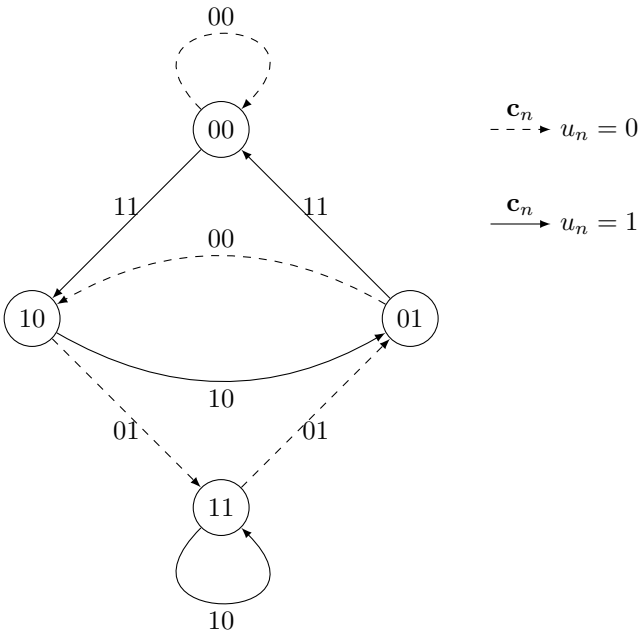

Figure 2.3 – *Représentation en diagramme d'états du code convolutif* $(1, 5/7)_8$

La représentation en diagramme d'états est utilisée pour caractériser le spectre de distances d'un code convolutif, c'est à dire l'ensemble des couples (d, A_d) où d est un poids d'un mot de code et A_d est le nombre de mots de codes possédant le poids d. Ce spectre de distances sera utilisé pour obtenir une borne sur la probabilité d'erreur d'un code convolutif.

Représentation en treillis

La représentation en treillis d'un code convolutif est similaire à celle en diagramme d'états. La principale différence entre ces deux représentations est que la représentation en treillis fait apparaitre directement la dimension temporelle. Le treillis correspondant à un code $(1, 5/7)_8$ est représenté en Figure 2.4. Dans

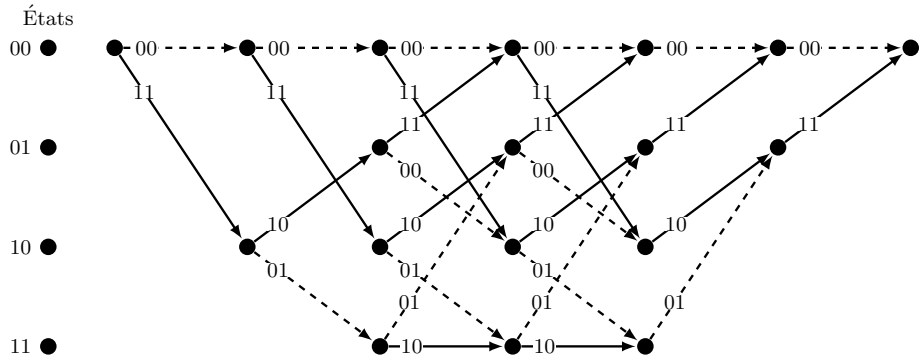

Figure 2.4 – *Code convolutif utilisant un encodeur récursif systématique* $(1, \frac{5}{7})_8$

ce treillis, nous avons supposé qu'une fermeture a été réalisée. Le codeur est ainsi laissé dans l'état 00 à la fin de l'encodage du mot de code. Pour réaliser cette fermeture, $\nu - 1$ bits supplémentaires sont nécessaires. Le rendement de code est donc inférieur à 1/2 et est donné par

$$R = \frac{K}{2 * (K + \nu - 1)}. \tag{2.8}$$

Il convient de remarquer que pour une grande taille de mots de code ($K \gg \nu$) $R \approx \frac{1}{2}$.

La représentation en treillis présentée dans cette section est particulièrement utile pour créer des algorithmes de décodage efficaces pour les codes convolutifs. La prochaine section de ce cours est dédiée à ces algorithmes de décodages.

2.2 Décodage

2.2.1 Critère ML séquence

Algorithme de Viterbi

En 1967, Viterbi introduit un algorithme permettant d'implémenter un décodeur ML séquence pour les codes convolutifs, et ce, pour une complexité largement résuite comparée à l'exploration de tous les mots de codes. Ce décodeur s'appuie sur la représentation en treillis présentée dans la section précédente et utilise un argument de programmation dynamique afin de construire de façon séquentielle le mot de code.

Soit $\mathbf{u} = (u_0, u_1, \cdots, u_{K-1})$ le mot d'information de K bits a encoder. En supposant que le treillis a été fermé, mot de code $\mathbf{c} = (\mathbf{c}_0 \dots \mathbf{c}_{L-1})$ correspondant est de longueur $N = nL$ où $L = K + \nu - 1$ où n représente le nombre de sorties de l'encodeur. Dans le Chapitre 1, nous avons vu que le décodage au sens du maximum de vraisemblance s'écrit :

- $\hat{w} = \arg\min_w d_H(\mathbf{x}(w)|\mathbf{y})$ pour le canal BSC, en prenant $\mathbf{x} = \mathbf{c}$
- $\hat{w} = \arg\min_w d_E^2(\mathbf{x}(w)|\mathbf{y})$ pour le canal (BI-)AWGN en prenant $\mathbf{x} = 1 - 2\mathbf{c}$

Dans les deux cas, la métrique a minimiser s'écrit sous la forme suivante

$$\Gamma_L = \sum_{l=0}^{L-1} \psi_l \tag{2.9}$$

où ψ_l est la fonction

- $\psi_l = \sum_{j=0}^{n-1} d_H(y_l^{(j)}, x_l^{(j)})$ pour le canal BSC (dans ce cas $\mathbf{x} = \mathbf{c}$)
- $\psi_l = \sum_{j=0}^{n-1} \left| y_l^{(j)} - x_l^{(j)} \right|^2$ pour le canal (BI-)AWGN, avec $x_l^{(j)} = 1 - 2c_l^{(j)}$.

Pour le reste de cette section, $\Gamma_L(s)$ est calculé comme Γ_L à la différence que l'état final est $s_L = s$. Si le treillis a été fermé, le décodage ML devient

$$\hat{w} = \arg\min_w \Gamma_L(0). \tag{2.10}$$

Afin de réduire la complexité algorithmique inhérente au décodeur du maximum de vraisemblance, Viterbi utilise un argument de programmation dynamique qui

peut être résumé comme suit. Soit $(s_0^\star, s_1^\star, \ldots, s_L^\star)$ les nœuds du treillis participant au chemin minimisant $\Gamma_L(0)$, alors $(s_0^\star, s_1^\star, \ldots, s_l^\star)$ pour $l < L$ doit minimiser $\Gamma_l(s_l^\star)$. La démonstration de cette propriété peut être réalisée par l'absurde. Supposons que $(s_0^\star, s_1^\star, \ldots, s_L^\star)$ minimise $\Gamma_L(0)$. Remarquons d'abord que pour ce chemin $\Gamma_L(0)$ s'écrit comme

$$\Gamma_L(0) = \Gamma_l(s_l^\star) + \sum_{n=l+1}^{L-1} \psi_l. \tag{2.11}$$

Si $(s_0^\star, s_1^\star, \ldots, s_l^\star)$ ne minimise pas $\Gamma_l(s_l^\star)$, alors, en remplaçant $(s_0^\star, s_1^\star, \ldots, s_l^\star)$ par une séquence minimisant $\Gamma_l(s_l^\star)$, on améliore $\Gamma_L(0)$, ce qui est en contradiction avec le fait que $(s_0^\star, s_1^\star, \ldots, s_l^\star)$ minimise $\Gamma_L(0)$.

L'algorithme de Viterbi exploite cette propriété est s'écrit, pour chaque $l \leq L$ et chaque état s

$$\Gamma_l(s) = \min_{s'} \Gamma_{l-1}(s') + \psi_{l-1}(s', s) \tag{2.12}$$

où $\psi_{l-1}(s', s)$ est défini comme précédemment mais en considérant que les $x_l^{(j)}$ sont ceux obtenus par la transition entre l'état s' et l'état s. Lorsque $n = 2$, l'algorithme de Viterbi est représenté schématiquement sur le treillis donné en Figure 2.5.

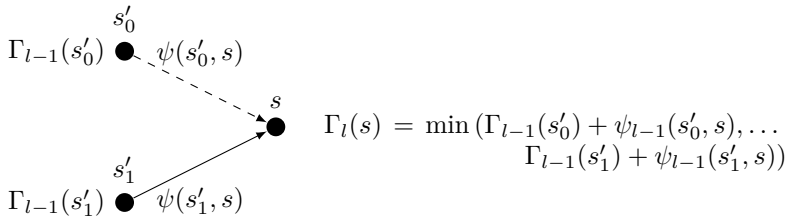

Figure 2.5 – *Algorithme de Viterbi*

Probabilité d'erreur du décodage de Viterbi (canal BI-AWGN)

Dans cette section, nous abordons le calcul de la probabilité d'erreur du décodage ML "séquence". Pour le canal BI-AWGN, il convient de remarquer que $p(y|x) = p(-y| - x)$ et que le code convolutif est linéaire. Ces deux propriétés réunies impliquent que $\mathbb{P}(erreur|\mathbf{x}) = \mathbb{P}(erreur|\mathbf{x}')$ pour tous mots de codes émis $\mathbf{x}$ et $\mathbf{x}'$. En reprenant les étapes du calcul de P_e effectuées dans le Chapitre 1, nous pouvons considérer uniquement le mot de code identiquement nul (noté $\mathbf{x_0}$)

$$\begin{aligned}
P_e &= \sum_{\mathbf{x}} \mathbb{P}(erreur|\mathbf{x}) p(\mathbf{x}) && (2.13)\\
&= \sum_{\mathbf{x}} \mathbb{P}(\bigcup_{\hat{\mathbf{x}} \neq \mathbf{x}} \textit{Décider } \hat{\mathbf{x}}|\mathbf{x}) p(\mathbf{x}) && (2.14)\\
&\leq \sum_{\mathbf{x}} \sum_{\hat{\mathbf{x}} \neq \mathbf{x}} \mathbb{P}(\textit{Décider } \hat{\mathbf{x}}|\mathbf{x}) p(\mathbf{x}) && (2.15)\\
&= \sum_{\hat{\mathbf{x}} \neq \mathbf{x_0}} \mathbb{P}(\textit{Décider } \hat{\mathbf{x}}|\mathbf{x_0}) && (2.16)
\end{aligned}$$

Le mot de code transmis étant $\mathbf{x_0} = (A, A, \dots A)$ et le mot de code reçu étant $\hat{\mathbf{x}} = (-A, A, \cdots -A)$, nous avons $d_E(\mathbf{x_0}, \hat{\mathbf{x}}) = 2A\sqrt{d(\hat{\mathbf{x}})} = 2\sqrt{d(\hat{\mathbf{x}})RE_b}$ où $d(\hat{\mathbf{x}})$ est le poids de Hamming du mot de code $\hat{\mathbf{c}}$ correspondant à $\hat{\mathbf{x}}$, $E_b = \frac{A^2}{R}$ est l'énergie par bit d'information transmis et R le rendement du code. Comme sur chaque dimension, le vecteur $\mathbf{x_0}$ est bruité par un bruit de variance $\sigma^2 = \frac{N_0}{2}$, P_e s'écrit

$$P_e = \sum_{\hat{\mathbf{x}} \neq \mathbf{x_0}} Q\left(\frac{d_E(\mathbf{x_0}, \hat{\mathbf{x}})}{2\sigma}\right) = \sum_{\hat{\mathbf{x}} \neq \mathbf{x_0}} Q\left(\sqrt{2d(\hat{\mathbf{x}})R\frac{E_b}{N_0}}\right) \tag{2.17}$$

Soit A_d le nombre de mots de codes tels que $d(\hat{\mathbf{x}}) = d$, on obtient finalement

$$P_e = \sum_d A_d Q\left(\sqrt{2dR\frac{E_b}{N_0}}\right). \tag{2.18}$$

On retrouve ici le spectre en distance (d, A_d) du code convolutif. Ce spectre peut être obtenu en parcourant tout les chemins de longueur L commençant dans l'étant 0 et finissant dans ce même état dans le diagramme d'état du code.

2.2.2 Critère MAP bit et algorithme BCJR

Dans la section précédente, nous avons vu le décodeur ML séquence, il minimise la probabilité d'erreur paquet. Dans cette section, nous considèrerons le critère MAP "bit" qui minimise la probabilité d'erreur binaire. Ce décodeur sera utilisé en pratique pour le décodage des turbocodes. Aussi, dans cette section, nous nous focaliserons sur le canal BI-AWGN.

Comme nous l'avons vu dans le Chapitre 1, le critère MAP "bit" est défini par

$$\hat{u_l} = \arg \max_{u_l \in \{0,1\}} p(u_l|\mathbf{y}). \tag{2.19}$$

La probabilité $p(u_l|\mathbf{y})$ est appelée probabilité *a posteriori* (APP) du bit u_l sachant le signal reçu $\mathbf{y}$. Considérons maintenant le logarithme du ratio des APP

$$L(u_l) = \log\left(\frac{p(u_l = 0|\mathbf{y})}{p(u_l = 1|\mathbf{y})}\right) \tag{2.20}$$

cette quantité est appelée Logarithme du ratio des vraisemblance noté LLR (Log Likelihood Ratio). A partir des LLR, la décision sur u_l s'écrit

$$\hat{u_l} = \begin{cases} 0 \text{ si } L(u_l) \geq 0 \\ 1 \text{ si } L(u_l) < 0 \end{cases} \tag{2.21}$$

Pour la suite de cette section, nous nous concentrons sur un code récursif systématique (RSC) de rendement $R \approx 1/2$ (avec une entrée et 2 sorties). Les mots d'informations sont de longueur K et L est la longueur incluant la fermeture du canal, $\mathbf{u} = (u_0, u_1, \dots u_{L-1})$. Les mots de codes s'écrivent $\mathbf{x} = (x_0^u x_0^p, x_1^u x_1^p, \dots x_{L-1}^u x_{L-1}^p)$ avec $x_l^u = 1 - 2u_l$ et $x_l^p = 1 - 2p_l$. Le signal reçu $\mathbf{y} = \mathbf{x} + \mathbf{z}$ s'écrit alors $\mathbf{y} = (\mathbf{y}_0, \mathbf{y}_1 \dots \mathbf{y}_{L-1})$ où $\mathbf{y}_l = (y_l^u, y_l^p)$ et $\mathbf{z}$ est iid gaussien de moyenne nulle et de variance σ^2.

L'algorithme BCJR (Bahl, Cocke, Jelinek, Raviv) a été introduit en 1974 et utilise le treillis afin de factoriser astucieusement l'expression des LLR $L(u_l)$. Tout d'abord, on peut remarquer que $L(u_l)$ peut se ré-écrire comme

$$L(u_l) = \log\left(\frac{\sum_{\mathcal{U}^0} p(s_l = s, s_{l+1} = s', \mathbf{y})}{\sum_{\mathcal{U}^1} p(s_l = s, s_{l+1} = s', \mathbf{y})}\right) \tag{2.22}$$

où $\mathcal{U}^0$ (resp $\mathcal{U}^1$) est l'ensemble des paires (s, s') telles que la transition $(s_l = s) \rightarrow (s_{l+1} = s')$ est engendrée par $u_l = 0$ (resp. $u_l = 1$). Pour la suite de cette section, on notera $p(s, s', \mathbf{y}) = p(s_l = s, s_{l+1} = s', \mathbf{y})$. Le point crucial du développement du BCJR repose dans la factorisation de $p(s, s', \mathbf{y})$ de la façon suivante

$$p(s, s', \mathbf{y}) = \alpha_l(s)\gamma_l(s, s')\beta_{l+1}(s') \tag{2.23}$$

où $\alpha_l(s)$, $\gamma_l(s, s')$ et $\beta_l(s)$ sont définis comme

$$\begin{aligned} \alpha_l(s) &= p(s_l = s, \mathbf{y}_0^{l-1}) && (2.24)\\ \gamma_l(s, s') &= p(s_{l+1} = s', \mathbf{y}_l | s_l = s) && (2.25)\\ \beta_l(s) &= p(\mathbf{y}_l^{L-1} | s_l = s) && (2.26) \end{aligned}$$

En effet, nous pouvons écrire

$$\begin{aligned} p(s, s', \mathbf{y}) &= p(s_l = s, s_{l+1} = s', \mathbf{y})\\ &= p(s_l = s, s_{l+1} = s', \mathbf{y}_0^{l-1}, \mathbf{y}_l, \mathbf{y}_{l+1}^{L-1})\\ &= p(\mathbf{y}_{l+1}^{L-1} | s, s', \mathbf{y}_0^{l-1}, \mathbf{y}_l) p(s, s', \mathbf{y}_0^{l-1}, \mathbf{y}_l)\\ &= p(\mathbf{y}_{l+1}^{L-1} | s, s', \mathbf{y}_0^{l-1}, \mathbf{y}_l) p(s', \mathbf{y}_l | s, \mathbf{y}_0^{l-1}) p(s, \mathbf{y}_0^{l-1})\\ &= p(\mathbf{y}_{l+1}^{L-1} | s') p(s', \mathbf{y}_l | s) p(s, \mathbf{y}_0^{l-1})\\ &= \beta_{l+1}(s')\gamma_l(s, s')\alpha_l(s) \end{aligned}$$

Il faut maintenant déterminer $\alpha_l(s)$, $\beta_l(s)$ et $\gamma_l(s, s')$ pour tous l, s et s'. Pour cela, nous allons démontrer les propriétés suivantes :

a) Les probabilités $\alpha_l(s)$ sont obtenues à l'aide de la **récursion "aller"** suivante

$$\alpha_{l+1}(s') = \sum_s \gamma_l(s, s')\alpha_l(s) \tag{2.27}$$

b) Les probabilités $\beta_l(s)$ sont obtenues à l'aide de la récursion "retour" suivante

$$\beta_l(s) = \sum_{s'} \gamma_l(s, s')\beta_{l+1}(s') \tag{2.28}$$

c) Les probabilités $\gamma_l(s, s')$ sont définies comme

$$\gamma_l(s, s') = \frac{p(u_l)}{2\pi\sigma^2} \exp\left(-\frac{||\mathbf{y}_l - \mathbf{x}_l(s, s')||^2}{2\sigma^2}\right) \tag{2.29}$$

Les probabilités $\alpha_l(s)$ et $\beta_l(s)$ sont parfois appelées **métriques de nœuds** alors que $\gamma_l(s, s')$ sont appelées **métriques de branches**.

Démonstration de la propriété a)
Afin de démontrer cette propriété, nous repartons de la définition

$$\begin{aligned}
\alpha_{l+1}(s') &= p(s_{l+1}=s', \mathbf{y}_0^l) \\
&= \sum_s p(s_l=s, s_{l+1}=s', \mathbf{y}_0^l) \\
&= \sum_s p(s_{l+1}=s', \mathbf{y}_l | s_l=s, \mathbf{y}_0^{l-1}) p(s_l=s, \mathbf{y}_0^{l-1}) \\
&= \sum_s p(s_{l+1}=s', \mathbf{y}_l | s_l=s) p(s_l=s, \mathbf{y}_0^{l-1}) \\
&= \sum_s \gamma_l(s,s') \alpha_l(s)
\end{aligned}$$

La quatrième ligne est obtenue à partir de la troisième en utilisant le fait que $\mathbf{Y}_0^{l-1}$ est indépendant de S_{l+1} et $\mathbf{Y}_l$ conditionnellement à S_l.

Démonstration de la propriété b)
Il en va de même sur la démonstration de la propriété b)

$$\begin{aligned}
\beta_l(s) &= p(\mathbf{y}_l^{L-1} | s_l=s) \\
&= \sum_{s'} p(\mathbf{y}_l^{L-1}, s_{l+1}=s' | s_l=s) \\
&= \sum_{s'} p(\mathbf{y}_{l+1}^{L-1} | s_l=s, s_{l+1}=s', \mathbf{y}_l) p(s_{l+1}=s', \mathbf{y}_l | s_l=s) \\
&= \sum_{s'} p(\mathbf{y}_{l+1}^{L-1} | s_{l+1}=s') p(s_{l+1}=s', \mathbf{y}_l | s_l=s) \\
&= \sum_{s'} \beta_{l+1}(s') \gamma_l(s,s')
\end{aligned}$$

La quatrième ligne est obtenue à partir de la troisième en utilisant le fait que $\mathbf{Y}_l$ est indépendant de S_{l+1} et $\mathbf{Y}_{l+1}^{L-1}$ conditionnellement à S_{l+1}.

Ces deux propriétés doivent être convenablement initialisées. Ceci est réalisé en initialisant $\alpha_0(s)$ comme suit

$$\alpha_0(s) = \begin{cases} 1, \text{ si } s=0, \\ 0, \text{ si } s \neq 0, \end{cases} \tag{2.30}$$

et $\beta_L(s)$ comme suit

$$\beta_L(s) = \begin{cases} 1, \text{ si } s=0, \\ 0, \text{ si } s \neq 0. \end{cases} \tag{2.31}$$

Ces deux initialisations supposent que l'état de l'encodeur est initialement l'état 0 et que le treillis a été fermé de sorte que l'état final soit aussi l'état 0.

Démonstration de la propriété c)
Nous démontrons maintenant la propriété *c*). La définition de $\gamma_l(s, s')$ donne

$$\begin{aligned}\gamma_l(s, s') &= p(s', \mathbf{y}_l|s) \\ &= \frac{p(s, s')}{p(s)} \frac{p(s', \mathbf{y}_l, s)}{p(s, s')} \\ &= p(s'|s)p(\mathbf{y}_l|s, s')\end{aligned} \tag{2.32}$$

Cette dernière ligne doit être intrprétée comme suit : s'il n'existe pas de transition entre s et s', $p(s'|s) = 0$ et $\gamma_l(s, s') = 0$; s'il existe un transition entre s et s' alors $\gamma_l(s, s')$ devient

$$\begin{aligned}\gamma_l(s, s') &= p(u_l)p(\mathbf{y}_l|\mathbf{x}_l(s, s')) \\ &= p(u_l)\frac{1}{2\pi\sigma^2} \exp\left(-\frac{||\mathbf{y}_l - \mathbf{x}_l(s, s')||^2}{2\sigma^2}\right)\end{aligned}$$

La dernière équation étant obtenue pour un canal AWGN de variance σ^2. Ceci conclut la démonstration de la propriété c). Néanmoins, dans le cas où la distribution *a priori* de u_l est uniforme, $p(u_l) = 1/2$. Dans les turbocodes, cette probabilité *a priori* ne sera plus uniforme et sera fourni par un second décodeur.

En résumé, la décision sur u_l se fait sur le signe de $L(u_l)$. $L(u_l)$ se calcule comme

$$L(u_l) = \log\left(\frac{\sum_{\mathcal{U}^0} p(s_l = s, s_{l+1} = s', \mathbf{y})}{\sum_{\mathcal{U}^1} p(s_l = s, s_{l+1} = s', \mathbf{y})}\right) \tag{2.33}$$

où $p(s, s', \mathbf{y}) = \beta_{l+1}(s')\gamma_l(s, s')\alpha_l(s)$ et les $\alpha_l(s)$, $\beta_{l+1}(s')$ et $\gamma_l(s, s')$ sont obtenus, sur le treillis, grâce aux équations suivantes

$$\alpha_{l+1}(s') = \sum_s \gamma_l(s, s')\alpha_l(s), \tag{2.34}$$

$$\beta_l(s) = \sum_s \gamma_l(s, s')\beta_{l+1}(s'), \tag{2.35}$$

$$\gamma_l(s, s') = \frac{1}{4\pi\sigma^2} \exp\left(-\frac{||\mathbf{y}_l - \mathbf{x}_l(s, s')||^2}{2\sigma^2}\right). \tag{2.36}$$

Les équations (2.33)-(2.36) définissent l'algorithme BCJR. Ces équation sont numériquement instables pour des mots de codes de grandes tailles. En conséquences, nous présentons dans la prochaine section, une version logarithmique de l'algorithme BCJR appelée Log-MAP.

2.2.3 Algorithme Log MAP

Les instabilités numériques évoquées précédemment proviennent de faibles valeurs de $\alpha_l(s)$, $\beta_l(s)$ et $\gamma_l(s, s')$ pouvant être considérées comme 0 à cause de quantifications. En conséquence, $\alpha_l(s)$ sera remplacé par la métrique suivante

$$\tilde{\alpha}_l(s) = \log\left(\alpha_l(s)\right) \tag{2.37}$$

qui vérifie la récursion aller

$$\tilde{\alpha}_{l+1}(s') = \log\left(\sum_{s} \exp\left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s,s')\right)\right) \tag{2.38}$$

avec $\tilde{\alpha}_0(s)$ défini par

$$\tilde{\alpha}_0(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases} \tag{2.39}$$

$\beta_l(s)$ sera remplacé par

$$\tilde{\beta}_l(s) = \log\left(\beta_l(s)\right) \tag{2.40}$$

qui vérifie la récursion retour suivante

$$\tilde{\beta}_l(s) = \log\left(\sum_{s'} \exp\left(\tilde{\beta}_{l+1}(s') + \tilde{\gamma}_l(s,s')\right)\right) \tag{2.41}$$

avec $\tilde{\beta}_L(s)$ défini par

$$\tilde{\beta}_L(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases} \tag{2.42}$$

$\gamma_l(s,s')$ sera remplacé par la métrique suivante

$$\begin{aligned} \tilde{\gamma}_l(s,s') &= \log\left(\gamma_l(s,s')\right) \\ &= -\log(p(u_l)) - \log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left|\left|\mathbf{y}_l - \mathbf{x}_l\right|\right|^2 \\ &= -\log(4\pi\sigma^2) - \frac{1}{2\sigma^2}\left|\left|\mathbf{y}_l - \mathbf{x}_l\right|\right|^2 \end{aligned} \tag{2.43}$$

La dernière ligne étant obtenu pour un a priori uniforme sur u_l.

Enfin, $L(u_l)$ peut être calculé à partir des métriques $\tilde{\alpha}_l(s)$, $\tilde{\beta}_l(s)$ et $\tilde{\gamma}_l(s,s')$ de la façon suivante

$$\begin{aligned} L(u_l) &= \log\left(\sum_{(s,s')\in\mathcal{U}^0} p(s,s',\mathbf{y})\right) - \log\left(\sum_{(s,s')\in\mathcal{U}^1} p(s,s',\mathbf{y})\right) \\ &= \log\left(\sum_{(s,s')\in\mathcal{U}^0} \exp\left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s,s') + \tilde{\beta}_{l+1}(s')\right)\right) \\ &\quad - \log\left(\sum_{(s,s')\in\mathcal{U}^1} \exp\left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s,s') + \tilde{\beta}_{l+1}(s')\right)\right). \end{aligned}$$

Jusqu'a présent, les nouvelles métriques proposées ne changent ni la simplicité ni la stabilité des calculs (à cause des exponentielles). L'astuce utilisée afin d'améliorer la stabilité numérique (et la simplicité d'écriture) réside dans un calcul numériquement stable de la fonction

$$\max^{\star}(x,y) = \log\left(e^x + e^y\right). \tag{2.44}$$

Pour cela, il suffit de remarquer que

$$\max^\star(x,y) = \max(x,y) + \log\left(1 + e^{-|x-y|}\right). \tag{2.45}$$

La fonction $\max^\star$ est alors implémentée comme un max et on utilise une table pour calculer efficacement le terme correctif. Cette fonction peut être étendue à 3 variables ou plus en remarquant que

$$\begin{aligned} \max^\star(x,y,z) &= \log\left(e^x + e^y + e^z\right) \\ &= \max^\star(\max^\star(x,y),z) \end{aligned} \tag{2.46}$$

En utilisant la fonction $\max^\star$ les équations (2.33)-(2.36) définissant le BCJR deviennent

$$\tilde{\alpha}_{l+1}(s') = \max_s^\star\left(\tilde{\gamma}_l(s,s') + \tilde{\alpha}_l(s)\right), \tag{2.47}$$

$$\tilde{\beta}_l(s) = \max_{s'}^\star\left(\tilde{\gamma}_l(s,s') + \tilde{\beta}_{l+1}(s')\right), \tag{2.48}$$

$$\tilde{\gamma}_l(s,s') = -\frac{1}{2\sigma^2}\left|\left|\mathbf{y}_l - \mathbf{x}_l\right|\right|^2. \tag{2.49}$$

Dans l'expression de $\tilde{\gamma}_l(s,s')$, le terme $-\log(4\pi\sigma^2)$ a été enlevé car, comme nous le verrons, il n'intervient pas dans l'expression de $L(u_l)$. En effet $L(u_l)$ est donné par

$$\begin{aligned} L(u_l) = &\max_{(s,s')\in\mathcal{U}^0}^\star\left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s,s') + \tilde{\beta}_{l+1}(s')\right) \\ &- \max_{(s,s')\in\mathcal{U}^1}^\star\left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s,s') + \tilde{\beta}_{l+1}(s')\right). \end{aligned}$$

le terme $-\log(4\pi\sigma^2)$ étant une constante additive, elle est simplifiée dans la différence. Il convient de remarquer que $\max^\star(x+a,y+a) = a + \max^\star(x,y)$.

Un pseudo-code de l'algorithme BCJR pour le décodeur Log-MAP est synthétisé dans l'encadré intitulé Algorithme 1. L'opérateur $\max^\star$ faisant intervenir les fonctions log et exp, la complexité de cet algorithme est assez élevée. Dans la section suivante, nous donnons un algorithme sous-optimal possédant une complexité réduite.

2.2.4 Algorithme Max-Log MAP

Afin de diminuer la complexité de l'algorithme Log MAP, on peut remplacer l'opérateur $\max^\star$ par l'opérateur max. Cette substitution donne lieu à l'algorithme Max-Log MAP. Il est moins complexe que l'algorithme Log MAP et il peut être exécuté sans connaitre σ^2 ; par contre, l'algorithme Max-Log MAP a de moins bonne performances que l'algorithme Log MAP en terme de BER. Ce décodeur est celui implémenté en pratique.

Algorithme 1 Algorithme Log MAP

Initialiser $\tilde{\alpha}_0(s)$ et $\tilde{\beta}_L(s)$ avec

$$\tilde{\alpha}_0(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

$$\tilde{\beta}_L(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

Pour $l = 0$ à $L - 1$ **faire**
  Pour chaque transition $s \to s'$ valide **faire**
    Calculer $\tilde{\gamma}_l(s, s')$ avec

$$\tilde{\gamma}_l(s, s') = -\frac{1}{2\sigma^2} \left|\left|\mathbf{y}_l - \mathbf{x}_l(s, s')\right|\right|^2$$

  Fin Pour
  Pour chaque état s' **faire**
    Calculer $\tilde{\alpha}_{l+1}(s')$ en utilisant la récursion

$$\tilde{\alpha}_{l+1}(s') = \max_s{}^{\star} \left(\tilde{\gamma}_l(s, s') + \tilde{\alpha}_l(s)\right)$$

  Fin Pour
Fin Pour

Pour $l = L - 1$ à 1 par pas de -1 **faire**
  Pour chaque état s **faire**
    Calculer $\tilde{\beta}_l(s)$ en utilisant la récursion suivante

$$\tilde{\beta}_l(s) = \max_{s'}{}^{\star} \left(\tilde{\gamma}_l(s, s') + \tilde{\beta}_{l+1}(s')\right)$$

  Fin Pour
Fin Pour
Pour $l = 0$ à $L - 1$ **faire**
  Calculer $L(u_l)$ comme

$$\begin{aligned} L(u_l) \quad = \quad & \max_{(s,s') \in \mathcal{U}^0}{}^{\star} \left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s, s') + \tilde{\beta}_{l+1}(s')\right) \\ & - \max_{(s,s') \in \mathcal{U}^1}{}^{\star} \left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s, s') + \tilde{\beta}_{l+1}(s')\right) \end{aligned}$$

  $\mathcal{U}^0$ est l'ensemble des transitions $(s \to s')$ engendrées par $u_l = 0$
  $\mathcal{U}^1$ est l'ensemble des transitions $(s \to s')$ engendrées par $u_l = 1$
Fin Pour

- Chapitre 3 -

Turbocodes

Contents

Les turbocodes ont été introduits en 1993 par C. Berrou, A. Glavieux et P. Thitimajshima. Ils représentent une percée majeure dans l'univers du codage de canal par le fait d'être les premiers codes à pouvoir s'approcher la capacité de Shannon (pour de long mots de codes).

Les turbocodes peuvent être généralement définis comme étant une association en parallèle ou en série de plusieurs encodeurs convolutifs.

3.1 Turbocodes parallèles

3.1.1 Structure de l'encodeur

La concaténation en parallèle de deux encodeurs RSC (RSC1 et RSC2) est représentée en Figure 3.1.

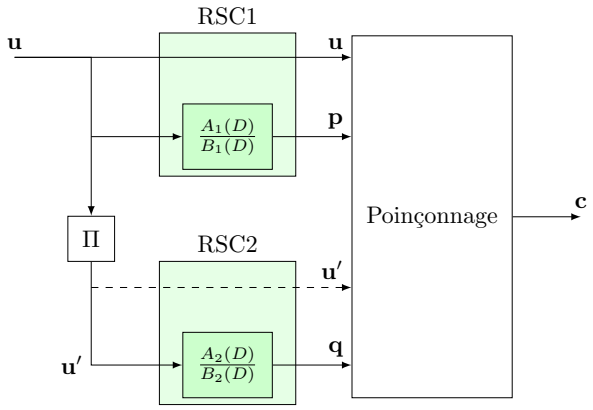

Figure 3.1 – *Schéma de principe d'un code turbo parallèle*

Comme il est montré en Figure 3.1, RSC1 et RSC2 sont séparés par un **entrelaceur**. Celui-ci joue un rôle primordial pour les performances du turbocode. En dépit du fait de "mélanger" le mot d'information **u**, l'entrelaceur permet :

— d'assurer une dispersion temporelle des erreurs produites en rafales et ainsi de les faire apparaitre comme des erreurs isolées pour le second décodeur. L'effet des rafales d'erreur est ainsi amoindri.
— d'assurer une grande distance minimale (si l'entrelaceur est convenablement choisi).

Le dernier bloc de la Figure 3.1 est la **fonction de poinçonnage**. Ce bloc définit quels sont les bits transmis ou non en sortie de l'encodeur. Cette fonction est effectuée périodiquement et est décrite par une matrice binaire P contenant autant de lignes que de sorties de l'encodeur (4 pour la Figure 3.1) et autant de colonnes que la période du poinçonneur. Considérons par exemple la matrice

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

signifie que les éléments de **u** ne sont pas modifiés, les éléments de $\mathbf{u}'$ sont tous poinçonnés, alors que seuls les éléments pairs (resp. impairs) de **p** (resp. **q**) sont transmis. Généralement, les éléments de **u** ne sont pas poinçonnés et les éléments de $\mathbf{u}'$ le sont. La matrice donnée ne concerne alors que **p** et **q** et ne comporte plus que 2 lignes. Une matrice équivalente à P serait

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

Le but du poinçonnage est d'augmenter le rendement du code : en effet, soit R_0 le rendement avant poinçonnage, le rendement du code après poinçonnage est obtenu par la relation suivante

$$R = R_0 \frac{\#\text{ d'éléments dans P}}{\#\text{ d'éléments à 1 dans P}}. \tag{3.1}$$

Pour une concaténation parallèle telle que RSC1 est de rendement R_1 et RSC2 de rendement R_2, le rendement R_0 est donné par

$$R_0 = \frac{R_1 R_2}{R_1 + R_2}. \tag{3.2}$$

3.1.2 Décodage itératif des turbocodes parallèles

Principe du décodage itératif

Le décodage des turbocodes est itératif. Son principe de fonctionnement est illustré dans la Figure 3.2.

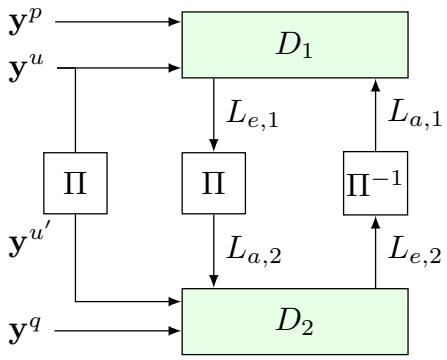

Figure 3.2 – *Schéma de principe du décodage itératif d'un turbocode parallèle*

TODO: Les décodeurs D_1 et D_2 sont des décodeurs à entrées et sorties souples (Log-MAP ou Max-Log-MAP) associés respectivement aux encodeurs RSC1 et RSC2. D_1 reçoit $\mathbf{y}^u$ et $\mathbf{y}^p$ délivrés par le canal ainsi que $L_{a,1}$, une information a-priori délivrée par D_2 sous forme de log-ratio de vraisemblances. Cette information a-priori est issue d'une information extrinsèque générée par D_2. Dans l'autre sens, D_2 reçoit $\mathbf{y}^{u'}$ et $\mathbf{y}^q$ délivrés par le canal ainsi que $L_{a,2}$, une information a-priori délivrée par D_1.
Ce procédé est appelé décodage itératif ; il continue jusqu'à ce que le nombre voulu d'itération soit effectué. À la fin de ce processus, un décodeur peut calculer $L(u_l)$ comme la somme de son information a priori (L_a), son information extrinsèque (L_e) et l'information du canal L_c. En effet, $L(u_l)$ peut s'exprimer ainsi

$$\begin{aligned} L(u_l) &= \log\left(\frac{p(u_l=0|\mathbf{y})}{p(u_l=1|\mathbf{y})}\right) \\ &= \log\left(\frac{p(\mathbf{y}|u_l=0)}{p(\mathbf{y}|u_l=1)}\right) + \log\left(\frac{p(u_l=0)}{p(u_l=1)}\right) \\ &= \underbrace{\log\left(\frac{p(\mathbf{y}_l|u_l=0)}{p(\mathbf{y}_l|u_l=1)}\right)}_{\text{Information du canal}} + \underbrace{\log\left(\frac{p(\mathbf{y}_{\sim l}|u_l=0)}{p(\mathbf{y}_{\sim l}|u_l=1)}\right)}_{\text{Information extrinsèque}} + \underbrace{\log\left(\frac{p(u_l=0)}{p(u_l=1)}\right)}_{\text{Information a priori}} \\ &= L_c(u_l) + L_e(u_l) + L_a(u_l) \end{aligned} \tag{3.3}$$

Le décodage considéré est un décodage MAP-bit effectué selon la règle suivante :

$$\hat{u}_l = \begin{cases} 0, \text{ si } L(u_l) \geq 0 \\ 1, \text{ sinon.} \end{cases}$$

Détails du décodage

Les décodeurs D_1 et D_2 sont tous deux des décodeurs BCJR vus dans le chapitre précédent. Ils utilisent une information a priori afin d'amélierer leur décodage. Cette information a priori $L_a(u_l)$ doit être compris comme un log

ratio de vraisemblance :

$$L_a(u_l) \Leftrightarrow \log\left(\frac{p(u_l=0)}{p(u_l=1)}\right).$$

Nous rappelons que dans l'algorithme Log-MAP, la métrique de branche est définie comme

$$\tilde{\gamma}_l(s,s') = \log\left(p(u_l)\right) - \log\left(2\pi\sigma^2\right) - \frac{||\mathbf{y}_l - \mathbf{x}_l||^2}{2\sigma^2} \tag{3.4}$$

où $\mathbf{x}_l$ et u_l sont obtenus par la transition de l'état s vers l'état s'. Afin d'incorporer $L_a(u_l)$ à la métrique $\tilde{\gamma}_l(s,s')$ il convient de remarquer que $p(u_l)$ s'écrit

$$\begin{aligned} p(u_l) &= \frac{\exp\left(L_a(u_l)/2\right)}{1+\exp\left(L_a(u_l)\right)} \exp\left(x_l^u L_a(u_l)/2\right) &(3.5)\\ &= A_l \exp\left(x_l^u L_a(u_l)/2\right) &(3.6)\end{aligned}$$

où $x_l^u = 1 - 2u_l$. Après suppression des termes constants par rapport à u_l (attention, $L_a(u_l)$ ne dépends pas de u_l mais de sa distribution), il reste

$$\tilde{\gamma}_l(s,s') = x_l^u \frac{L_a(u_l)}{2} + \frac{y_l^u}{\sigma^2} x_l^u + \frac{y_l^p}{\sigma^2} x_l^p \tag{3.7}$$

Le terme $||\mathbf{y}_l - \mathbf{x}_l||^2$ peut être remplacé par $-2(y_l^u x_l^u + y_l^p x_l^p)$ car les termes $||\mathbf{y}_l||^2$ et $||\mathbf{x}_l||^2$ ne dépendent pas de u_l.

En utilisant cette nouvelle métrique de branche, le log ratio de vraisemblance $L(u_l)$ devient

$$\begin{aligned} L(u_l) &= \max_{(s,s')\in\mathcal{U}^0}{}^{\star}\left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s,s') + \tilde{\beta}_{l+1}(s')\right)\\ &\qquad - \max_{(s,s')\in\mathcal{U}^1}{}^{\star}\left(\tilde{\alpha}_l(s) + \tilde{\gamma}_l(s,s') + \tilde{\beta}_{l+1}(s')\right)\\ &= L_a(u_l) + \frac{2y_l^u}{\sigma^2} + \max_{(s,s')\in\mathcal{U}^0}{}^{\star}\left(\tilde{\alpha}_l(s) + \frac{y_l^p}{\sigma^2}x_l^p + \tilde{\beta}_{l+1}(s')\right)\\ &\qquad - \max_{(s,s')\in\mathcal{U}^1}{}^{\star}\left(\tilde{\alpha}_l(s) + \frac{y_l^p}{\sigma^2}x_l^p + \tilde{\beta}_{l+1}(s')\right)\\ &= L_a(u_l) + L_c(u_l) + L_e(u_l)\end{aligned}$$

Le décodeur envoie à son décodeur companion l'information extrinsèque $L_e(u_l)$ calculée comme suit

$$L_e(u_l) = \max_{(s,s')\in\mathcal{U}^0}{}^{\star}\left(\tilde{\alpha}_l(s) + \frac{y_l^p}{\sigma^2}x_l^p + \tilde{\beta}_{l+1}(s')\right) - \max_{(s,s')\in\mathcal{U}^1}{}^{\star}\left(\tilde{\alpha}_l(s) + \frac{y_l^p}{\sigma^2}x_l^p + \tilde{\beta}_{l+1}(s')\right).$$

L'algorithme de décodage itératif est résumé dans l'encart nommé Algorithme 3.

Algorithme 2 Décodage itératif d'un turbocode parallèle

Initialisation

Pour $\mathbf{D_1}$

Initialiser $\tilde{\alpha}_0^{(1)}(s)$ avec :

$$\tilde{\alpha}_0^{(1)}(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

Si le treillis de RSC1 est fermé, initialiser $\tilde{\beta}_L^{(1)}(s)$ comme

$$\tilde{\beta}_L^{(1)}(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

Si le treillis de RSC1 n'est pas fermé, initialiser $\tilde{\beta}_L^{(1)}(s)$ comme

$$\tilde{\beta}_L^{(1)}(s) = -\log N_s^{(1)} \text{ pour } s = 0, 1, \ldots, N_s^{(1)}$$

où $N_s^{(1)}$ est le nombre d'états dans le treillis associé à RSC1.

$L_{a,1}(u_l) = 0$, pour $l = 0, 1 \ldots, L-1$.

Pour $\mathbf{D_2}$
Initialiser $\tilde{\alpha}_0^{(2)}(s)$ avec :

$$\tilde{\alpha}_0^{(2)}(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

Initialiser $\tilde{\beta}_L^{(2)}(s)$ comme

$$\tilde{\beta}_L^{(2)}(s) = -\log N_s^{(2)}$$

où $N_s^{(2)}$ est le nombre d'états dans le treillis associé à RSC2.

$L_{a,2}(u_l)$ n'a pas à être initialisé, il sera défini après la première demi itération.

Itération n :

$\mathbf{D_1}$:
Pour $l = 0$ à $L-1$ **faire**
  Pour chaque transition $s \to s'$ valide **faire**
    Calculer $\tilde{\gamma}_l(s, s')$ avec

$$\tilde{\gamma}_l^{(1)}(s, s') = x_l^u \frac{L_{a,1}(u_l)}{2} + x_l^u \frac{y_l^u}{\sigma^2} + x_l^p \frac{y_l^p}{\sigma^2}$$

  Fin Pour
  Pour chaque état s' **faire**
    Calculer $\tilde{\alpha}_{l+1}^{(1)}(s')$ en utilisant la récursion

$$\tilde{\alpha}_{l+1}^{(1)}(s') = \max_s{}^{\star} \left(\tilde{\gamma}_l^{(1)}(s, s') + \tilde{\alpha}_l^{(1)}(s) \right)$$

  Fin Pour
Fin Pour

Pour $l = L-1$ à 1 par pas de -1 **faire**
  Pour chaque état s **faire**
    Calculer $\tilde{\beta}_l^{(1)}(s)$ en utilisant la récursion suivante

$$\tilde{\beta}_l^{(1)}(s) = \max_{s'}{}^{\star} \left(\tilde{\gamma}_l^{(1)}(s, s') + \tilde{\beta}_{l+1}^{(1)}(s') \right)$$

Fin Pour
Fin Pour
Pour $l = 0$ à $L-1$ **faire**
Calculer $L_{e,1}(u_l)$ comme

$$\begin{aligned} L_{e,1}(u_l) &= \max_{(s,s')\in\mathcal{U}^0}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + \frac{y_l^p}{\sigma^2}x_l^p + \tilde{\beta}_{l+1}^{(1)}(s')\right) \\ &\quad - \max_{(s,s')\in\mathcal{U}^1}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + \frac{y_l^p}{\sigma^2}x_l^p + \tilde{\beta}_{l+1}^{(1)}(s')\right) \end{aligned}$$

$\mathcal{U}^0$ est l'ensemble des transitions $(s \to s')$ engendrées par $u_l = 0$
$\mathcal{U}^1$ est l'ensemble des transitions $(s \to s')$ engendrées par $u_l = 1$
Fin Pour

$L_{a,2}(u'_l) = L_{e,1}(u_{\Pi(l)})$
D₂ :
Pour $l = 0$ à $L-1$ **faire**
Pour chaque transition $s \to s'$ valide **faire**
Calculer $\tilde{\gamma}_l^{(2)}(s,s')$ avec

$$\tilde{\gamma}_l^{(2)}(s,s') = x_l^{u'}\frac{L_{a,2}(u'_l)}{2} + x_l^{u'}\frac{y_l^{u'}}{\sigma^2} + x_l^q\frac{y_l^q}{\sigma^2}$$

Fin Pour
Pour chaque état s' **faire**
Calculer $\tilde{\alpha}_{l+1}^{(2)}(s')$ en utilisant la récursion

$$\tilde{\alpha}_{l+1}^{(2)}(s') = \max_s{}^{\star}\left(\tilde{\gamma}_l^{(2)}(s,s') + \tilde{\alpha}_l^{(2}(s)\right)$$

Fin Pour
Fin Pour

Pour $l = L-1$ à 1 par pas de -1 **faire**
Pour chaque état s **faire**
Calculer $\tilde{\beta}_l^{(2)}(s)$ en utilisant la récursion suivante

$$\tilde{\beta}_l^{(2)}(s) = \max_{s'}{}^{\star}\left(\tilde{\gamma}_l^{(2)}(s,s') + \tilde{\beta}_{l+1}^{(2)}(s')\right)$$

Fin Pour
Fin Pour
Pour $l = 0$ à $L-1$ **faire**
Calculer $L_{e,2}(u'_l)$ comme

$$\begin{aligned} L_{e,2}(u'_l) &= \max_{(s,s')\in\mathcal{U}'^0}{}^{\star}\left(\tilde{\alpha}_l^{(2)}(s) + \frac{y_l^q}{\sigma^2}x_l^q + \tilde{\beta}_{l+1}^{(2)}(s')\right) \\ &\quad - \max_{(s,s')\in\mathcal{U}'^1}{}^{\star}\left(\tilde{\alpha}_l^{(2)}(s) + \frac{y_l^q}{\sigma^2}x_l^p + \tilde{\beta}_{l+1}^{(2)}(s')\right) \end{aligned}$$

Fin Pour

$L_{a,1}(u_l) = L_{e,2}(u'_{\Pi^{-1}(l)})$

Après la dernière itération :
Pour $l = 0$ à $L-1$ **faire**
Calculer $L_{u,1}(u_l)$ avec

$$L_{u,1}(u_l) = \frac{2y_l^u}{\sigma^2} + L_{a,1}(u_l) + +L_{e,1}(u_l)$$

et calculer $\hat{u}_l$ en utilisant la règle suivante

$$\hat{u}_l = \begin{cases} 0 \text{ si } L_{u,1}(u_l) \geq 0 \\ 1 \text{ si } L_{u,1}(u_l) < 0 \end{cases}$$

Fin Pour

3.2 Turbocodes séries

3.2.1 Structure de l'encodeur

La concaténation en **série** de deux encodeurs RSC (RSC1 et RSC2) est représentée en Figure 3.3. Un message d'information **u** est encodé par RSC1

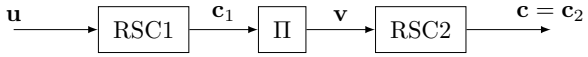

Figure 3.3 – *Schéma de principe d'un code turbo parallèle*

donnant $\mathbf{c}_1 = [u_0, p_0, u_1, p_1, \ldots u_{L-1}]$

3.2.2 Décodage des codes concaténés en série

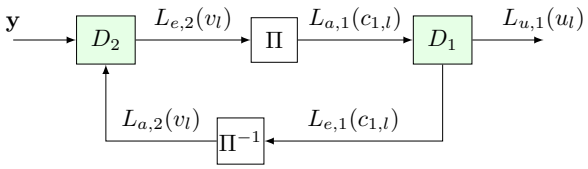

Figure 3.4 – *Schéma de principe d'un code turbo parallèle*

À la différence des codes concaténés en parallèle, les codes concaténés en série ne focalisent pas sur les bits systématiques. En effet, les bits de parités ajoutés par RSC1 sont à considérer comme des bits systématiques pour RSC2. En conséquence, le décodage BCJR doit être modifié afin que D_2 génère aussi bien des extrinsèques sur les bits systématiques **u** que sur les bits de parité **p**.

D_1 doit être modifié aussi, car les observations qu'il possède sur ses bits systématiques (**v**) reposent uniquement sur les sorties souples du décodeur D_2. En effet, il n'y a pas d'information provenant du canal pour D_1.

Algorithme 3 Décodage itératif d'un turbocode série

Initialisation
Pour $\mathbf{D_1}$
Initialiser $\tilde{\alpha}_0^{(1)}(s)$ avec :

$$\tilde{\alpha}_0^{(1)}(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

$$\tilde{\beta}_L^{(1)}(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

$L_{a,1}(u_l)$ sera initialisé après la première demi-itération.

Pour $\mathbf{D_2}$
Initialiser $\tilde{\alpha}_0^{(2)}(s)$ avec :

$$\tilde{\alpha}_0^{(2)}(s) = \begin{cases} 0, \text{ si } s = 0, \\ -\infty, \text{ si } s \neq 0. \end{cases}$$

Initialiser $\tilde{\beta}_{2L}^{(2)}(s)$ comme

$$\tilde{\beta}_{2L}^{(2)}(s) = -\log N_s^{(2)}$$

où $N_s^{(2)}$ est le nombre d'états dans le treillis associé à RSC2.

$L_{a,2}(c_l) = 0$ pour $l = 0, 1, \ldots, 2L-1$.

Itération n :

$\mathbf{D_2}$:
Pour $l = 0$ à $2L-1$ **faire**
  Pour chaque transition $s \to s'$ valide **faire**
    Calculer $\tilde{\gamma}_l(s, s')$ avec

$$\tilde{\gamma}_l^{(2)}(s, s') = x_l^v \frac{L_{a,2}(v_l)}{2} + x_l^v \frac{y_l^v}{\sigma^2} + x_l^q \frac{y_l^q}{\sigma^2}$$

  Fin Pour
  Pour chaque état s' **faire**
    Calculer $\tilde{\alpha}_{l+1}^{(2)}(s')$ en utilisant la récursion

$$\tilde{\alpha}_{l+1}^{(2)}(s') = \max_s{}^{\star} \left(\tilde{\gamma}_l^{(2)}(s, s') + \tilde{\alpha}_l^{(2)}(s) \right)$$

  Fin Pour
Fin Pour

Pour $l = 2L-1$ à 1 par pas de -1 **faire**
  Pour chaque état s **faire**

Calculer $\tilde{\beta}_l^{(2)}(s)$ en utilisant la récursion suivante

$$\tilde{\beta}_l^{(2)}(s) = \max_{s'}{}^{\star} \left(\tilde{\gamma}_l^{(2)}(s, s') + \tilde{\beta}_{l+1}^{(2)}(s) \right)$$

Fin Pour
Fin Pour
Pour $l = 0$ à $2L - 1$ **faire**
Calculer $L_{e,2}(v_l)$ comme

$$\begin{aligned} L_{e,2}(v_l) &= \max_{(s,s')\in\mathcal{V}^0}{}^{\star} \left(\tilde{\alpha}_l^{(2)}(s) + \frac{y_l^v}{\sigma^2} x_l^v + \frac{y_l^q}{\sigma^2} x_l^q + \tilde{\beta}_{l+1}^{(1)}(s') \right) \\ &\quad - \max_{(s,s')\in\mathcal{V}^1}{}^{\star} \left(\tilde{\alpha}_l^{(2)}(s) + \frac{y_l^v}{\sigma^2} x_l^v + \frac{y_l^q}{\sigma^2} x_l^q + \tilde{\beta}_{l+1}^{(1)}(s') \right) \end{aligned}$$

$\mathcal{V}^0$ est l'ensemble des transitions $(s \to s')$ engendrées par $v_l = 0$
$\mathcal{V}^1$ est l'ensemble des transitions $(s \to s')$ engendrées par $v_l = 1$
Fin Pour

$L_{a,1}(c_{1,l}) = L_{e,2}(v_{\Pi(l)})$
$L_{a,1}(u_l) = L_{a,1}(c_{1,2l})$
$L_{a,1}(p_l) = L_{a,1}(c_{1,2l+1})$
D_1 :
Pour $l = 0$ à $L - 1$ **faire**
Pour chaque transition $s \to s'$ valide **faire**
Calculer $\tilde{\gamma}_l^{(2)}(s, s')$ avec

$$\tilde{\gamma}_l^{(1)}(s, s') = x_l^u \frac{L_{a,1}(u_l)}{2} + x_l^p \frac{L_{a,1}(p_l)}{2}$$

Fin Pour
Pour chaque état s' **faire**
Calculer $\tilde{\alpha}_{l+1}^{(1)}(s')$ en utilisant la récursion

$$\tilde{\alpha}_{l+1}^{(1)}(s') = \max_{s}{}^{\star} \left(\tilde{\gamma}_l^{(1)}(s, s') + \tilde{\alpha}_l^{(1}(s) \right)$$

Fin Pour
Fin Pour

Pour $l = L - 1$ à 1 par pas de -1 **faire**
Pour chaque état s **faire**
Calculer $\tilde{\beta}_l^{(1)}(s)$ en utilisant la récursion suivante

$$\tilde{\beta}_l^{(1)}(s) = \max_{s'}{}^{\star} \left(\tilde{\gamma}_l^{(1)}(s, s') + \tilde{\beta}_{l+1}^{(1)}(s) \right)$$

Fin Pour
Fin Pour
Pour $l = 0$ à $L - 1$ **faire**

Calculer $L_{e,1}(u_l) = L_{e,1}(c_{1,2l})$ comme

$$\begin{aligned} L_{e,1}(u_l) &= \max_{(s,s')\in\mathcal{U}^0}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + x_l^p\frac{L_{a,1}(p_l)}{2} + \tilde{\beta}_{l+1}^{(1)}(s')\right) \\ &\quad - \max_{(s,s')\in\mathcal{U}^1}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + x_l^p\frac{L_{a,1}(p_l)}{2} + \tilde{\beta}_{l+1}^{(1)}(s')\right) \end{aligned}$$

Calculer $L_{e,1}(p_l) = L_{e,1}(c_{1,2l+1})$ comme

$$\begin{aligned} L_{e,1}(p_l) &= \max_{(s,s')\in\mathcal{P}^0}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + x_l^u\frac{L_{a,1}(u_l)}{2} + \tilde{\beta}_{l+1}^{(1)}(s')\right) \\ &\quad - \max_{(s,s')\in\mathcal{P}^1}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + x_l^u\frac{L_{a,1}(u_l)}{2} + \tilde{\beta}_{l+1}^{(1)}(s')\right) \end{aligned}$$

Fin Pour
$L_{a,2}(v_l) = L_{e,1}(c_{1,\Pi^{-1}(l)})$

Après la dernière itération :
Calculer $\tilde{\gamma}_l^{(1)}(s,s')$ avec

$$\tilde{\gamma}_l^{(1)}(s,s') = x_l^u\frac{L_{a,1}(u_l)}{2} + x_l^p\frac{L_{a,1}(p_l)}{2}$$

Pour $l = 0$ à $L-1$ **faire**
Calculer $L_{u,1}(u_l)$ avec

$$\begin{aligned} L_{u,1}(u_l) &= \max_{(s,s')\in\mathcal{U}^0}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + \tilde{\gamma}_l^{(1)}(s,s') + \tilde{\beta}_{l+1}^{(1)}(s')\right) \\ &\quad - \max_{(s,s')\in\mathcal{U}^1}{}^{\star}\left(\tilde{\alpha}_l^{(1)}(s) + \tilde{\gamma}_l^{(1)}(s,s') + \tilde{\beta}_{l+1}^{(1)}(s')\right) \end{aligned}$$

et calculer $\hat{u}_l$ en utilisant la règle suivante

$$\hat{u}_l = \begin{cases} 0 \text{ si } L_{u,1}(u_l) \geq 0 \\ 1 \text{ si } L_{u,1}(u_l) < 0 \end{cases}$$

Fin Pour