Thèse

présentée pour obtenir le grade de docteur

**de l'Ecole Nationale Supérieure
des Télécommunications**

Spécialité : Electronique et Communications

Sandrine Vialle

Construction et analyse de nouvelles structures de codage de canal adaptées au traitement itératif

Soutenance prévue le 19 décembre 2000 devant le jury composé de

Gérard Battail	**Examinateurs**
Sergio Benedetto	
Claude Berrou	
Marc Fossorier	
Guillaume Vivier	
Joseph Boutros	**Directeur de thèse**

Ecole Nationale Supérieure des Télécommunications, Paris

Table des matières

Liste des Abréviations

La signification d'une abréviation est donnée lors de sa première apparition dans le texte. Il existe la plupart du temps une abréviation en français et en anglais. Seule l'abréviation la plus usuelle, souvent l'abréviation en anglais, sera employée. Voici la liste des acronymes utilisés dans ce rapport :

APP	A Posteriori Probability, probabilité a posteriori
AWGN	Additive White Gaussian Noise, bruit blanc additif gaussien
BCH	Bose-Chaudhuri-Hocquenghem (code)
BPSK	Binary Phase Shift Keying, modulation de phase à 2 états
DMC	Discrete Memoryless Channel, canal discret sans mémoire
iid	indépendant et identiquement distribué
GLD	Generalized Low Density, code à faible densité généralisé
GMD	Generalized Minimum Distance
IOWEF	Input-Output Weight Enumerator Function, polynôme énumérateur de poids entrée-sortie
LDPC	Low Density Parity Check, code à faible densité
LR	Log-Ratio, rapport logarithmique de probabilités
MAP	Maximum *a posteriori* critère du maximum a posteriori
ML	Maximum Likelihood, critère du maximum de vraisemblance
NRNS	Non Recursive Non Systematic, code convolutif non récursif non systématique
PAM	Pulse Amplitude Modulation, modulation d'amplitude en phase
PCCC	Parallel Concatenation of convolutional codes, turbo-code parallèle
pce	parity check equation, équation de parité
RSC	Recursive Systematic Convolutional, code convolutif récursif systématique
SCCC	Serial Concatenation of convolutional codes, turbo-code série
SISO	Soft-Input Soft-Output, décodeur à entrée et à sortie souples
TB	Tail Biting Codes, codes convolutifs à treillis circulaire
v.a.	Variable aléatoire

Liste des Notations

Les listes suivantes regroupent la majeure partie des notations employées dans les différents chapitres de ce document. Certaines notations, qui apparaissent ponctuellement ont été omises.

Performances des codes optimaux avec décodage ML

$h(x)$	information intrinsèque de la v.a. x
$h(x,y)$	information intrinsèque d'une paire de v.a. x et y
$h(x\|y)$	information intrinsèque conditionnelle des v.a. x et y
$i(x;y)$	information mutuelle des v.a. x et y
$H(X)$	entropie de la v.a. X
$H(X,Y)$	entropie conjointe des v.a. X et Y
$H(X\|Y)$	entropie conditionnelle des v.a. X et Y
$I(X,Y)$	information mutuelle moyenne
C	capacité du canal
$P(x)$	probabilité de la v.a. discrète x
$p(x)$	densité de probabilité de la v.a. continue x
$\mathcal{C}(N,K)$	code de longueur N et de dimension K
$\mathbf{c} = (c_1, \ldots, c_N)$	mot de code de $\mathcal{C}$
c_i	i-ème composante du mot de code $\mathbf{c}$
R	rendement du code $\mathcal{C}$
$d_{\mathrm{H}min}$	distance de Hamming minimale du code $\mathcal{C}$
$E(R)$	exposant de Gallager
α	évanouissement de Rayleigh
L	ordre de diversité
M	nombre total de mots de code de $\mathcal{C}$
Q	nombre de symboles d'une modulation
$E[x]$	espérance mathématique de la v.a. x
$E_x[\ ]$	espérance mathématique conditionnellement à la v.a. x
H_2	entropie binaire
E	énergie par mot de code
E_c	énergie moyenne par bit codé
E_b	énergie moyenne par bit d'information
$\mathbf{y} = (y_1, \ldots, y_N)$	sortie du canal AWGN
$\mathbf{z} = (z_1, \ldots, z_N)$	bruit gaussien de moyenne nulle et de variance N_0 par composante

$\mathrm{Sph}(O, N)$	sphère N-dimensionnelle de centre O et de rayon $\sqrt{2NE_c}$
$V(\mathbf{c})$	région de Voronoï de $\mathbf{c}$
$\Omega(\theta)$	angle solide de la région de Voronoï $V(\mathbf{c})$
$\Omega(\pi) = \Omega_N$	angle solide total d'une sphère en dimension N
θ	demi-angle au sommet d'un cône d'angle solide $\Omega(\theta)$
$\Lambda(\mathbf{c})$	cône circulaire de même angle solide $\Omega(\theta)$ que $V(\mathbf{c})$
$\mathrm{Sph}(\mathbf{a}, N-1)$	sphère de centre $\mathbf{a}$ et de dimension $N-1$
$\mathcal{H}$	hyperplan de dimension $N-1$
$\mathbf{a}$	projection de $\mathbf{c}$ sur l'hyperplan $\mathcal{H}$
$P\mathrm{e}_{\mathrm{w}}$	probabilité d'erreur moyenne par mot
$P\mathrm{e}_{\mathrm{b}}$	probabilité d'erreur moyenne par bit
$Pe_{\mathbf{c}}$	probabilité d'erreur par mot sachant que $\mathbf{c}$ a été émis
$P\mathrm{e}_{\mathrm{w}\mid\boldsymbol{\alpha}}$	probabilité d'erreur moyenne par mot pour un vecteur $\boldsymbol{\alpha}$ donné
$Q(\theta)$	probabilité qu'un mot de code situé sur l'axe de $\Lambda(\mathbf{c})$ soit déplacé à l'extérieur de $\Lambda(\mathbf{c})$
$\mathrm{Vol}(r, N)$	volume d'une sphère à N dimensions de rayon r
$\mathrm{S}(r, N)$	surface d'une sphère à N dimensions de rayon r
$\boldsymbol{\alpha} \odot \mathbf{c}$	vecteur issu du produit composante à composante des vecteur $\boldsymbol{\alpha}$ et $\mathbf{c}$
$\mathcal{F}_i$	facette de la région de Voronoï entre les mots de code $\mathbf{c}$ et $\mathbf{c_i}$
M_i	milieu du segment $[\mathbf{c}\ \mathbf{c_i}]$
λ, γ	multiplicateurs de Lagrange

Principes des codes composites

$\mathcal{C}(N, K, d)$	code composite de longueur N, de dimension K et de distance minimale de Hamming d
R	rendement du code $\mathcal{C}(N, K)$
H	matrice de parité du code $\mathcal{C}(N, K)$
J	nombre de codes constituants dans le code composite $\mathcal{C}$
$C^1, \ldots, C^J$	codes constituants du code composite $\mathcal{C}$
$C^i(N_i, K_i, d_i)$	code linéaire de longueur N_i, de dimension K_i et de distance de Hamming minimale d_i
R_i	rendement du code $C^i(N_i, K_i, d_i)$
$C_c^i(n_i, k_i, d_i)$	code convolutif de distance libre d_i, k_i(resp. n_i) bits en entrée (resp. en sortie) du codeur
r_i	rendement du code convolutif $C_c^i(n_i, k_i, d_i)$
$(g_1, \ldots, g_n)$	polynômes générateurs du code convolutif $C_c^i(n_i, k_i, d_i)$ en octal
$S(t)$	état d'un codeur convolutif à l'instant t
S_c	état circulant d'un code convolutif à treillis circulaire
H^i	matrice de parité du code C^i
$LDPC(N, j, \ell)$	code LDPC de longueur N dont la matrice de parité a j "1" par colonne et ℓ "1" par ligne
$H^1, \ldots, H^J$	matrices de parité des codes constituants
$\pi_1, \ldots, \pi_J$	permutations aléatoires entre J codes constituants d'un code composite
$\mathcal{C}_0(n_0, k_0)$	code élémentaire (i.e., équation de parité pour les LDPC ou code de Hamming pour les GLD blocs) de longueur n_0 et de dimension k_0

H_0	matrice de parité du code élémentaire C_0
C^*	somme directe des codes élémentaires C_0
r_0	rendement d'un code élémentaire C_0
$\overline{A(w)}$	nombre moyen de mots de code de poids w du code C
$A^C(w)$	nombre de mots de code de poids w du code C
$A^C(I,Z)$	polynôme énumérateur de poids du code C
$A^C(i,z)$	nombre de mots de code de C correspondant à un poids i en entrée et un poids z en sortie
$i_{\min}$	poids minimal de l'information en entrée d'un code convolutif pour que le treillis soit fermé
$z_{\min}$	poids minimal de la parité en sortie d'un code convolutif
$z_{\min}^{(2)}$	poids minimal de la parité en sortie d'un code convolutif pour un poids en entrée $i_{\min} = 2$
d_{eff}	distance effective d'un turbo-code parallèle
N_π	longueur de l'entrelaceur d'un turbo-code
δ	distance minimale normalisée
$g(s)$	moment générateur d'un code en blocs
$R(w)$	probabilité qu'un mot de poids w soit un mot du code constituant
$Q(x)$	fonction erreur
$\mathbf{c} = (c_1, \ldots, c_N)$	mot de code
$\mathbf{x} = (x_1, \ldots, x_N)$	mot de code $\mathbf{c}$ modulé
$\mathbf{y} = (y_1, \ldots, y_N)$	sortie du canal
N_0	variance par composante du bruit gaussien du canal AWGN
N_{nc}	nombre de noeuds contraintes dans le graphe de Tanner d'un code composite
W	nombre de branches dans le treillis d'un code convolutif
L	longueur de contrainte d'un code convolutif
ν	mémoire d'un code convolutif
M	nombre d'états dans le treillis d'un code convolutif
$APP(c_i)$	probabilité *a posteriori du bit codé* c_i
$Ext(c_i)$	information extrinsèque du bit codé c_i
$\Pr(y_i \| x_i)$	observation du canal
$P(c_i)$	probabilité *a priori du bit codé* c_i
$\sigma(m, m')$	probabilité *a posteriori de la transition de l'état* m *vers l'état* m'
$\gamma(m, m')$	métrique de branche de l'algorithme aller-retour entre les états m et m'
$\alpha(m)$	métrique aller de l'algorithme aller-retour pour l'état m
$\beta(m)$	métrique retour de l'algorithme aller-retour pour l'état m
$p(m, m')$	probabilité *a priori de la transition* m *de l'état* m *vers l'état* m'
$T = (V, E, A)$	treillis d'un code en blocs : V est l'ensemble des états, E l'ensemble des arêtes, et A l'ensemble des étiquettes sur les branches
h_i	i-ème colonne de la matrice de parité H
LV_i	vraisemblance du symbole reçu y_i
G	matrice génératrice d'un code en blocs
$\lambda_1,\ \lambda_2$	permutation dans l'algorithme de Fossorier-Lin
$d_E(\mathbf{u}, \mathbf{v})$	distance euclidienne entre les vecteurs $\mathbf{u}$ et $\mathbf{v}$

Codes GLD construits à partir de codes convolutifs

$\mathcal{C}(N, K, d_{\mathrm{H\,min}})$	code GLD convolutif de longueur N, de dimension K et de distance de Hamming minimale $d_{\mathrm{H\,min}}$
J	nombre de codes constituants d'un code GLD convolutif
$C_c^j(n_j, k_j, d_j)$	code convolutif de longueur n_j, de dimension k_j et de distance libre d_j
$C^j(N_j, K_j)$	code en blocs obtenu à partir de C_c^j par fermeture de son treillis
H^j	matrice de parité du code C^j
$C^1, \dots, C^J$	codes constituants d'un code GLD convolutif
$\pi_1, \dots, \pi_J$	permutations aléatoires d'un code GLD convolutif
n	nombre de bits en sortie d'un codeur convolutif
k	nombre de bits en entrée d'un codeur convolutif
L	longueur de contrainte d'un code convolutif
ν	mémoire d'un code convolutif
r	rendement d'un code convolutif
$g_0, \dots, g_n$	polynômes générateurs d'un code convolutif
$H_{RSC\infty}$	matrice de parité de dimensions infinies d'un code convolutif RSC
H_{RSC}	matrice de parité du même code convolutif RSC dont le treillis est fermé
C_{ex}^{RSC}	code convolutif RSC de rendement $R = \frac{3}{4}$, de longueur de contrainte $L = 4$, et de polynômes générateurs en octal $(15, 16, 13, 17)$
C_{ex}^{NRNS}	code convolutif NRNS de rendement $R = \frac{3}{4}$, de longueur de contrainte $L = 2$, et de polynômes générateurs en octal $(13, 25, 61, 47)$
$\mathcal{C}_{\mathrm{PCCC}}$	concaténation parallèle de codes convolutifs (turbo-code parallèle)
$\mathcal{C}_{\mathrm{SCCC}}$	concaténation série de codes convolutifs (turbo-code série)
$m[n]$	m modulo n
S	symbole quaternaire

Limites asymptotiques des codes composites décodés itérativement

$\mathcal{C}(N, K)$	code composite de longueur N, de dimension K
J	nombre de codes constituants dans le code composite
R	rendement du code composite
$C^1, \dots, C^J$	codes constituants du code composite
$\pi_2, \dots, \pi_J$	permutations aléatoires du code composite
$C_0(n_0, k_0)$	code élémentaire d'un code GLD blocs
$C(n, k)$	code constituant du code composite, de longueur n et de dimension k
r	rendement du code $C(n, k)$
W	longueur de la fenêtre de dépendance d'un code
N_0	variance du bruit gaussien
$N_0(ap)$	variance de l'information extrinsèque (approximation gaussienne)
d_b	degré d'un bit dans le graphe de Tanner d'un code composite
d_c	degré d'un code dans le graphe de Tanner d'un code composite
d_s	degré d'un symbole dans le graphe de Tanner d'un code composite
m	itération de décodage courante
$\delta_a(C)$	degré d'anisotropie d'un code C
$A(x)$	polynôme énumérateur de poids d'un code

$A_i^0(x)$	polynôme énumérateur de poids conditionnel associé à un bit c_i égal à 0
$A_i^1(x)$	polynôme énumérateur de poids conditionnel associé à un bit c_i égal à 1
$A_{i\ell}^0$ $(A_{i\ell}^1)$	nombre de mots de code de poids ℓ tel que $c_i = 0$ $(c_i = 1)$
$\mathcal{A}(c_i)$	APP partielle d'un bit codé c_i
$appLR(c_i)$	rapport logarithmique des APP d'un bit codé c_i
$\mathcal{B}(c_i)$	rapport logarithmique des APP partielles d'un bit codé c_i
$extLR(c_i)$	rapport logarithmique des informations extrinsèques d'un bit codé c_i
$obsLR(c_i)$	rapport de log-vraisemblance d'un bit codé c_i
$P\mathrm{e}_m$	probabilité d'erreur à l'itération m calculée à partir des APP à l'itération m
$P\mathrm{e}'_m$	probabilité d'erreur calculée à partir des APP partielles à l'itération m
$P\mathrm{e}'$	point fixe de la probabilité d'erreur dans la méthode de la fonction de transfert
p_m	distribution de probabilité de l'APP d'un bit à l'itération m
H	fonction de transfert d'un sommet code dans le graphe de Tanner du code composite
G	fonction de transfert d'un sommet bit dans le graphe de Tanner du code composite
$P\mathrm{e}'_{\mathrm{in}}$	probabilité d'erreur à l'entrée d'un sommet dans le graphe
$P\mathrm{e}'_{\mathrm{out}}$	probabilité d'erreur à la sortie d'un sommet dans le graphe
$\mathcal{C}_{\mathcal{I}}(N_I, K_I)$	turbo-code irrégulier de longueur N_I et de dimension K_I
R_I	rendement du turbo-code irrégulier $\mathcal{C}_{\mathcal{I}}(N_I, K_I)$
$\mathcal{C}_{\mathcal{R}}(N_R, K_R)$	turbo-code régulier de même rendement et de même polynômes générateurs que $\mathcal{C}_{\mathcal{I}}(N_I, K_I)$
$\{f_d\}_{d=1,\dots,D}$	profil de degrés d'un turbo-code irrégulier
$\bar{d}$	degré moyen d'un turbo-code irrégulier
μ_d	probabilité qu'un sommet soit de degré d dans le graphe de Tanner
$P\mathrm{e}_m(d)$	probabilité d'erreur des bits de degré d à l'itération m
u_d	bit de degré d dans le graphe de Tanner

Table des figures

Liste des tableaux

Introduction

Le codage de canal est une fonction essentielle dans les systèmes de transmission de l'information. Il a acquis à ce titre une importance considérable durant ces dernières années. La plupart des systèmes actuels de transmission numérique en sont pourvus, des systèmes radio-mobiles (GSM, EDGE, UMTS, ...) aux systèmes d'enregistrement des données (disque compact, ...). La finalité du codage est de protéger les données contre les perturbations introduites par le canal de transmission et de proposer ainsi une meilleure qualité de service. Les avancées réalisées dans le domaine de l'électronique et du traitement du signal ont permis l'utilisation de schémas de codage de plus en plus performants dans les équipements de transmission.

La théorie du codage doit son origine à la théorie de l'information introduite en 1948 par Shannon [72]. Cette dernière fixe les limites ultimes en termes de codage correcteur d'erreurs par l'intermédiaire du théorème fondamental du codage. Celui-ci énonce qu'il est possible de transmettre un message codé sur un canal perturbé et de le restituer sans erreur après décodage, à condition que le rendement du code soit inférieur à la *capacité* du canal. L'obtention d'une probabilité d'erreur aussi faible que souhaitée est cependant sujette à l'emploi d'un code approprié, pour lequel Shannon ne suggère qu'une construction aléatoire. Le décodage est par ailleurs supposé à vraisemblance maximale (décodage ML, *Maximum Likelihood*) afin de minimiser la probabilité d'erreur par symbole [63]. Or, si la construction d'un code aléatoire semble aisée, il n'en est pas de même de la mise en œuvre de son décodage qui ne peut être qu'exhaustif et dont la complexité devient rapidement prohibitive lorsque la dimension du code augmente. Ainsi pendant de nombreuses années, l'existence d'un codage et d'un décodage explicites (*i.e.*, réalisables en pratique) semblait être incompatible avec les limites proposées par Shannon. De nombreux codes proposés durant ces années étaient munis d'une structure algébrique facilitant leur décodage mais non justifiée par la théorie de l'information.

En 1993, l'invention des turbo-codes par Berrou, Glavieux et Thitimajshima [11] apporte un regard nouveau sur l'existence de codes "réalistes" approchant les limites données par la théorie de l'information. Les turbo-codes s'appuient sur la concaténation parallèle de codes élémentaires liés par un entrelaceur aléatoire. Malgré une distance minimale relativement faible, leurs performances ne sont éloignées que de 0.5 dB des performances ultimes proposées par Shannon. Autrement dit, bien que ces codes soient de mauvais codes du point de vue des critères habituels de la théorie du codage (le rapport de leur distance minimale à leur longueur tend vers zéro lorsque celle-ci tend vers l'infini), ils atteignent des performances très proches des performances optimales. Leur distribution de distances présente, en effet, des similitudes avec la distribution de distances d'un code aléatoire, ce qui apparaît comme une explication de leurs excellentes performances [3].

Les turbo-codes font partie d'une catégorie plus vaste de codes imitant le codage aléatoire, appelés *codes composites*. Ces codes sont construits à partir de codes élémentaires simples (codes en blocs ou codes convolutifs) par concaténation, intersection ou produit. Un décodage à maximum de vraisemblance ne peut être envisagé pour ces codes du fait d'une trop grande complexité. Aussi, pour approcher les performances d'un décodage optimal, le décodage de ces codes consiste en un processus itératif fondé sur le décodage optimal successif de chaque code élémentaire. La perte d'information est minimisée par l'échange d'informations souples entre les différents décodeurs.

Le chapitre 1 de ce document débute par une présentation rapide des principes fondamentaux de la théorie de l'information. Le théorème fondamental du codage est introduit ainsi que la notion de capacité d'un canal. Les limites imposées par la capacité sont étudiées et analysées pour différents canaux. Partant du constat qu'il s'agit de limites asymptotiques en termes de longueur du code, les performances optimales d'un code sont ensuite envisagées pour une longueur et un rendement fixés. Cette étude est fondée sur une approche géométrique proposée par Shannon en 1959 pour le canal gaussien. Nous comparons les performances optimales ainsi évaluées avec les performances des turbo-codes pour différentes longueurs. Nous proposons ensuite d'observer l'évolution de ces probabilités d'erreur optimales en présence d'évanouissements, en étendant le raisonnement de Shannon au canal de Rayleigh.

Le chapitre 2 est dédié à l'introduction des codes composites qui apparaissent comme des codes performants permettant d'approcher les limites de Shannon évaluées au chapitre 1. Les différentes structures de codes composites sont présentées ainsi que leurs performances analytiques sous l'hypothèse d'un décodage à maximum de vraisemblance. Le processus de décodage itératif de ces codes est alors expliqué. Deux algorithmes de décodage à entrée souple sont envisagés : l'algorithme aller-retour [2] appliqué sur le treillis du code élémentaire et l'algorithme de Fossorier-Lin [37] pour les codes en blocs linéaires. Nous proposons d'adapter ce dernier afin qu'il puisse délivrer une sortie souple et être intégré dans le processus de décodage itératif d'un code composite construit à partir de codes en blocs. Nous traitons plus particulièrement le cas des codes produits, dont la structure permet d'envisager un décodage parallèle des codes constituants, et qui présentent de très bonnes performances, notamment pour de petites longueurs. Lorsque la longueur du code augmente, nous proposons d'utiliser des codes produits construits à partir de codes convolutifs à treillis circulaire afin d'améliorer leurs performances.

Dans le chapitre 3, nous introduisons un nouveau schéma de code composite, généralisant la structure des codes à faible densité proposés par Gallager en 1963 [42]. Ces codes sont appelés *codes à faible densité généralisés* et sont construits à partir de l'intersection de codes convolutifs. Nous analysons leurs performances théoriques, sous l'hypothèse d'un décodage ML, en observant leur distribution de poids moyenne et leur distance minimale. Nous déduisons de cette étude que ces codes sont asymptotiquement bons, c'est-à-dire que leur distance minimale est une fonction linéaire de leur longueur. Nous envisageons ensuite le décodage itératif des codes à faible densité généralisés et présentons leurs performances sur le canal gaussien.

Nous nous intéressons dans le chapitre 4 à l'évaluation des performances asymptotiques des codes composites décodés itérativement. En effet, les analyses des performances présentées aux chapitres précédents reposaient sur l'hypothèse d'un décodage à vraisemblance maximale. Nous proposons dans ce chapitre d'étudier les limites du décodage itératif des différents codes composites envisagés. Pour cela, nous introduisons deux méthodes différentes : l'une est fondée sur l'étude de la propagation des informations souples au cours du décodage ; l'autre observe graphiquement l'évolution de la probabilité d'erreur au fil des itérations. Ces deux méthodes nous permettent de déduire des critères de conception pour les codes composites, en prenant en compte les limites imposées par le décodage itératif.

Enfin nous tirerons quelques conclusions sur les études menées et indiquerons quelques perspectives pour la suite des travaux.

Chapitre 1

Performances des codes optimaux avec décodage ML *

La théorie de l'information définit les frontières du possible en matière de communications et plus particulièrement de codage. La connaissance de ces limites théoriques a poussé la communauté du codage à élaborer des schémas de plus en plus performants pour s'en approcher. Ainsi, en 1993, Berrou, Glavieux et Thitimajshima [11] ont introduit les turbo-codes, dont les performances se situent à 0.5 dB des bornes données par la théorie de Shannon [72].

Les limites proposées par la théorie de l'information sont fondées sur le théorème de codage [72] et constituent des limites asymptotiques pour ce qui est de la longueur du code. La mise au point des systèmes de codage nécessaires pour atteindre ces limites présente un degré de complexité non négligeable, malgré de nombreuses avancées dans ce domaine. Par ailleurs, certains domaines d'application du codage requièrent des longueurs de codes beaucoup plus petites, afin de réduire la complexité des systèmes et les délais en réception. C'est le cas par exemple de la transmission de la voix pour les communications mobiles. La question du choix de bons codes pour ces longueurs se pose alors. Les limites proposées par la théorie de l'information ne sont plus pertinentes dans le sens où les longueurs considérées ne tendent plus vers l'infini. Il faut ainsi étudier de nouvelles bornes prenant en compte la longueur finie du code. Une approche géométrique proposée par Shannon [73] permet de déterminer les performances des codes optimaux de dimension et de rendement fixés.

Après une brève présentation des concepts essentiels sur lesquels s'appuie la théorie de l'information, nous énonçons le théorème fondamental du codage et définissons la notion de capacité d'un canal. Cette dernière permet de poser les limites asymptotiques d'un système codé. Nous développons les expressions de la capacité pour différents canaux classiques, et les représentons en fonction du rapport signal-à-bruit du système. Nous nous intéressons dans une troisième partie aux limites des codes optimaux pour un rendement R et une longueur N fixés. Ces performances sont étudiées pour le canal gaussien en reprenant l'approche géométrique proposée par Shannon. Puis nous étudions l'évolution de ces performances pour le canal de Rayleigh. Nous comparons

*Le contenu de ce chapitre a été présenté en partie à la *37th Allerton Conference on Communication, Control and Computing*, Allerton, Etats-Unis, 24-27 septembre 1999.

alors ces résultats aux limites asymptotiques données par la théorie de l'information et aux performances obtenues en utilisant des systèmes de codage pratiques, tels les turbo-codes.

1.1 Concepts fondamentaux de la théorie de l'information

Cette section introduit les principales notions de la théorie de l'information, autrement dit les outils permettant de disposer d'une mesure *qualitative* de l'information et leurs propriétés. En théorie de l'information, l'intérêt se porte sur l'étude de canaux et de sources aléatoires. Par conséquent, l'imprévisibilité est un attribut essentiel de l'information. Aussi, la mesure qualitative de l'information va-t-elle se traduire par une mesure de l'imprévu et de l'improbable [3].

Le canal et la source sont modélisés par des variables aléatoires. Nous distinguons par la suite le cas où ces variables aléatoires sont discrètes du cas où elles sont continues. Si x est une variable aléatoire discrète, alors $P(x)$ désigne sa probabilité. Si x est une variable aléatoire continue, sa densité de probabilité est notée $p(x)$.

1.1.1 Cas discret

Soient x et y deux variables aléatoires discrètes de probabilités respectives $P(x)$ et $P(y)$. Nous rappelons les définitions suivantes :

- **information intrinsèque ou information propre** : incertitude *a priori* de l'événement x ou information nécessaire pour résoudre cette incertitude :
$$h(x) = -\log P(x) \ .$$
- **information intrinsèque d'une paire de variables aléatoires** (x,y) :
$$h(x,y) = -\log P(x,y) \ .$$
La quantité $P(x,y)$ désigne la probabilité conjointe des événements x et y. Les deux variables aléatoires x et y peuvent être mutuellement dépendantes.
- **information conditionnelle** : information devant être fournie à un observateur pour spécifier la variable x après observation de la variable y, ou incertitude restant sur la variable x après observation de la variable y :
$$h(x|y) = -\log P(x|y) \ .$$
$P(x|y)$ est la probabilité de x conditionnellement à y.
- **information mutuelle** : information sur l'événement x fournie par l'événement y :
$$i(x;y) = \log \frac{P(x|y)}{P(x)} \ .$$
On peut alors écrire :
$$i(x;y) = h(x) - h(x|y) = i(y;x) \ .$$

Ces définitions s'appliquent à des événements individuels. Dans un contexte probabilisé, les moyennes sont souvent plus importantes que les événements individuels. Nous considérons donc par la suite une source aléatoire, discrète, finie, stationnaire, obéissant à une loi de probabilité connue. Soit $\{x_1, x_2, \ldots, x_n\}$ l'ensemble des réalisations possibles de cette source : cet ensemble constitue un alphabet de n symboles. Chaque émission de la source est décrite par la variable aléatoire X, qui est égale au symbole x_i avec la probabilité :

$$P(X = x_i) = p_i,\ i = 1, \ldots, n \qquad \text{et} \qquad \sum_{i=1}^{n} p_i = 1 \ .$$

De même, soit Y une variable aléatoire prenant ses valeurs dans l'alphabet de m symboles $\{y_1, \ldots, y_m\}$ avec la probabilité :

$$P(Y = y_i) = q_i,\ i = 1, \ldots, m \qquad \text{et} \qquad \sum_{i=1}^{m} q_i = 1 \ .$$

En moyennant sur l'ensemble des réalisations des variables aléatoires X et Y les quantités précédemment introduites, nous obtenons les définitions suivantes :

– **entropie de la source (ou de X)** : valeur moyenne de l'information intrinsèque :

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \ .$$

L'entropie de X est une quantité positive vérifiant :

$$H(X) \leq \log(n) \tag{1.1}$$

– **entropie conjointe** :

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) \log P(x_i, y_j) \ .$$

– **entropie conditionnelle d'un ensemble** :

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) \log P(x_i|y_j) \ .$$

– **information mutuelle moyenne** :

$$I(X;Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) \log \frac{P(x_i|y_j)}{P(x_i)} \ .$$

L'information mutuelle moyenne entre les variables X et Y est positive ou nulle et concave.

La figure 1.1 illustre les différents concepts que nous venons de présenter ainsi que les relations suivantes [24] :

$$I(X;Y) = H(X) - H(X|Y) \ , \tag{1.2}$$

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y) \ ,$$

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

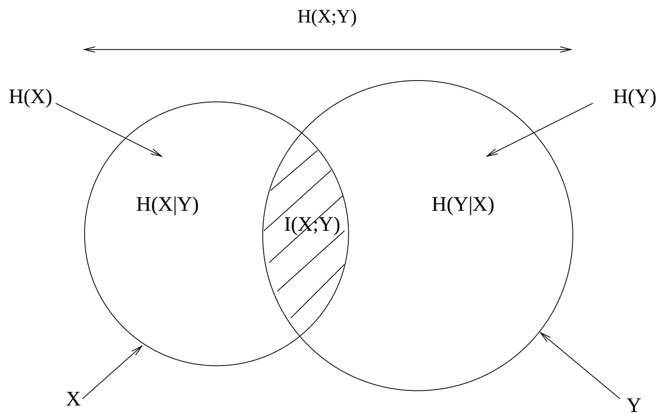


Fig. 1.1 – Relation entre l'information mutuelle et les entropies.

1.1.2 Cas continu

Soient X et Y deux variables aléatoires continues de densités de probabilité respectives $p_X(x)$ et $p_Y(y)$. Par souci de simplicité, $p_X(x)$ et $p_Y(y)$ seront notées $p(x)$ et $p(y)$ par la suite.

Dans le cas continu, la généralisation des grandeurs que nous venons de décrire peut se faire en prenant un minimum de précautions. En effet, l'entropie absolue d'une variable continue est infinie [49] : intuitivement, une variable continue pouvant prendre n'importe quelle valeur dans l'intervalle $]-\infty,+\infty[$, l'incertitude sur cette variable devient très grande. Par conséquent il est nécessaire d'introduire une nouvelle grandeur finie appelée entropie différentielle. L'entropie absolue d'une variable s'écrit alors comme la somme de deux termes, l'un fini, l'entropie différentielle et l'autre infini. Nous pouvons ainsi écrire les définitions suivantes :

– **entropie différentielle** :

$$H(X) = -\int_{-\infty}^{+\infty} p(x) \log p(x) dx \ . \tag{1.3}$$

– **entropie différentielle conjointe** :

$$H(X,Y) = -\int_x \int_y p(x,y) \log p(x,y) dx dy \ .$$

– **entropie différentielle conditionnelle d'un ensemble** :

$$H(X|Y) = -\int_x \int_y p(x,y) \log p(x|y) dx dy \ .$$

Nous avons alors la relation suivante :

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y) \ .$$

Notons que, contrairement aux entropies définies dans le cas discret, les entropies différentielles peuvent être négatives.

– **information mutuelle moyenne** :

$$I(X;Y) = \int_x \int_y p(x,y) \log \frac{p(x|y)}{p(x)} dxdy = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y) \ .$$

L'information mutuelle d'une variable continue est égale à la différence de deux entropies, et ne contient par conséquent plus de terme tendant vers l'infini. Il s'agit de la généralisation directe de l'information mutuelle d'une variable discrète.

1.2 Capacité d'un canal et théorème fondamental du codage

1.2.1 Définition

En l'absence de bruit, la capacité du canal est la quantité d'information maximale moyenne que chacun de ses symboles d'entrée peut apporter à sa sortie. La présence de bruit impose une limite supérieure sur le débit d'information qu'il est possible de transmettre à travers le canal, et conditionne ainsi la capacité.

La capacité d'un canal est définie comme le maximum de l'information mutuelle moyenne $I(X;Y)$, X et Y étant les variables aléatoires associées respectivement à l'entrée et à la sortie du canal. La maximisation est effectuée sur l'ensemble des distributions de probabilité $Pr_X(x)$ à l'entrée du canal[1] :

$$C = \max_{Pr_X(x)} I(X;Y)$$

Afin de faire apparaître l'effet du canal, la formule de l'information mutuelle est réécrite en remplaçant la loi conjointe par le produit de la loi marginale et de la loi conditionnelle :

– **canal discret (entrée et sortie discrètes)** :

$$C = \max_{P(x)} \sum_{x,y} P(x)P(y|x) \log \frac{P(y|x)}{\sum_{x'} P(x')P(y|x')} \ , \tag{1.4}$$

– **canal continu (entrée et sortie continues)** :

$$C = \max_{p(x)} \int_x \int_y p(x)p(y|x) \log \frac{p(y|x)}{\int_{x'} p(x')p(y|x')dx'} dxdy \ . \tag{1.5}$$

La capacité du canal est donc uniquement fonction du canal (plus précisément elle dépend de la probabilité de transition du canal), contrairement à l'information mutuelle moyenne qui dépend également de l'entrée du canal. En général, l'information mutuelle est une fonction strictement concave de la distribution d'entrée et possède ainsi un maximum unique. Ceci garantit l'existence de la capacité.

[1] $Pr_X(x)$ désigne indifféremment la probabilité $P(x)$ dans le cas discret ou la densité de probabilité $p(x)$ dans le cas continu

La maximisation sur la distribution en entrée du canal peut se faire à l'aide de techniques comme le théorème de maximisation sous contraintes de Lagrange ou de méthodes non linéaires comme la méthode du gradient [24]. Dans certains cas, la distribution d'entrée maximisant l'information mutuelle est connue. Pour les canaux discrets symétriques par exemple, le maximum de l'information mutuelle est atteint pour une distribution équiprobable des symboles d'entrée. Cette propriété facilite grandement le calcul de la capacité.

Notons que l'emploi du logarithme en base 2 dans les équations (1.4) et (1.5) permet d'exprimer la capacité en bits par dimension réelle.

1.2.2 Théorème fondamental du codage de canal

L'un des principaux résultats de la théorie de l'information réside dans le théorème de codage formulé par Shannon [72], dont l'énoncé est le suivant :

Théorème 1.2.1 *(Théorème fondamental du codage de canal)*
Pour un taux d'émission $R_b < C$, où C est la capacité du canal en bits/dimension, il existe un code de longueur N tel que sa probabilité d'erreur soit arbitrairement petite lorsque N tend vers l'infini. Cette probabilité d'erreur peut être majorée par une fonction de la forme :

$$Pe \leq 2^{-NE(R_b)} ,$$

où $E(R_b)$ est l'exposant de Gallager ou exposant de codage aléatoire [43]. L'exposant $E(R_b)$ est positif pour $R_b < C$.

Inversement, sur un canal sans mémoire, si $R_b \geq C$, quel que soit le code considéré et quelle que soit sa longueur N, la probabilité d'erreur moyenne par mot ne peut tendre vers 0.

Ce théorème établit que les données peuvent être transmises de manière fiable sur un canal même bruité lorsque le taux d'émission R_b est inférieur à la capacité. Le lecteur peut se référer aux ouvrages [43][24][3] pour une démonstration du théorème et de sa réciproque. Notons que ce résultat suppose que le décodage est à maximum de vraisemblance [63] (ML, *Maximum Likelihood*).

L'étude de la capacité du canal permet donc de fixer les limites asymptotiques d'un système codé. Si Q est la taille de la modulation utilisée et R le rendement du code correcteur d'erreurs, alors le taux d'émission R_b est égal à $R_b = R \times \log_2(Q)$. Pour ce taux d'émission R_b, la connaissance de la capacité du canal fournit la valeur maximale de la variance du bruit, ou de manière équivalente la valeur minimale du rapport signal-à-bruit, pour pouvoir assurer une transmission fiable des données lorsque la longueur du code tend vers l'infini. Les paragraphes suivants donnent l'expression de la capacité par dimension réelle pour les canaux gaussien (à bruit additif blanc gaussien, AWGN, *Additive White Gaussian Noise*) et de Rayleigh. L'influence de la diversité sur la capacité du canal de Rayleigh est également évoquée.

1.2.3 Expression de la capacité pour le canal AWGN

Deux cas sont envisagés pour le calcul de la capacité :
- aucune contrainte n'est imposée sur la distribution de l'entrée du canal : dans ce cas, la distribution gaussienne maximise l'information mutuelle du canal AWGN. L'entrée du canal est donc continue.
- la modulation à l'entrée du canal est fixée (entrée discrète) : il n'y a alors plus de maximisation sur la distribution en entrée du canal, mais nous appelons *capacité* l'information mutuelle obtenue avec cette distribution par abus de langage [80].

La sortie du canal gaussien est supposée réelle (*i.e.*, continue) dans les deux cas.

La première hypothèse représente le véritable calcul de la capacité du canal gaussien, puisque la distribution en entrée du canal est la distribution maximisant l'information mutuelle sur ce canal. La capacité ainsi évaluée donne par conséquent les performances optimales pouvant être atteintes sur le canal AWGN. Elle peut être infiniment grande puisque l'alphabet en entrée du canal est de taille infinie. Inversement, le cas où l'entrée du canal est discrète donne les limites du système codé lorsqu'une modulation finie est considérée en entrée du canal. La capacité est alors limitée par la taille de l'alphabet de la modulation d'après l'inégalité (1.1).

Soient X et Y, deux variables aléatoires représentant respectivement l'entrée et la sortie du canal AWGN. Le bruit, caractérisé par la variable aléatoire Z, est un bruit blanc additif gaussien de variance N_0 par dimension.

Capacité du canal AWGN à entrée et sortie continues

Pour un canal gaussien dont l'entrée et la sortie sont continues, l'information mutuelle moyenne est maximisée par une distribution gaussienne de la variable aléatoire X [43]. La capacité C du canal vérifie alors :

$$C = \frac{1}{2}\log_2\left(1 + 2C\frac{E_b}{N_0}\right) \ , \tag{1.6}$$

où E_b est l'énergie moyenne par bit d'information. Cette capacité s'annule pour $\left(\frac{E_b}{N_0}\right)_{dB} = -1.6$ dB.

Capacité du canal AWGN avec entrée discrète et sortie continue

Supposons en entrée du canal une modulation dont les Q symboles appartiennent à l'alphabet fini $\{x_0, x_1, \ldots, x_{Q-1}\}$. Si tous les symboles sont équiprobables, nous pouvons écrire [80] :

$$C = \sum_{q=0}^{Q-1}\int_y p(y, x_q)\log_2\frac{p(y|x_q)}{\sum_{q'=0}^{Q-1}\frac{1}{Q}p(y|x_{q'})}dy \ , \tag{1.7}$$

avec

$$p(y|x_q) = \frac{1}{\sqrt{2\pi N_0}}\exp\left(-\frac{|y-x_q|^2}{2N_0}\right) \ , \tag{1.8}$$

En combinant les équations (1.7) et (1.8), la capacité du canal s'écrit :

$$C = \log_2 Q - \frac{1}{Q} \sum_{q=0}^{Q-1} E_z \left[\log_2 \left(\sum_{q'=0}^{Q-1} \exp\left(-\frac{|z + x_q - x_{q'}|^2 - |z|^2}{2N_0} \right) \right) \right] , \tag{1.9}$$

où E_z caractérise l'espérance mathématique conditionnellement à la variable z.

Les équations (1.6) et (1.9) peuvent être résolues en utilisant la méthode numérique du point fixe [22]. La figure 1.2 représente la capacité d'un canal gaussien avec entrée discrète ou continue. Dans le cas discret, deux modulations ont été envisagées : la modulation de phase binaire (BPSK *Binary Phase Shift Keying*) et la modulation d'impulsion en amplitude à 4 états (4-PAM *4 Pulse Amplitude Modulation*). Nous remarquons sur cette figure que les capacités des canaux ayant une

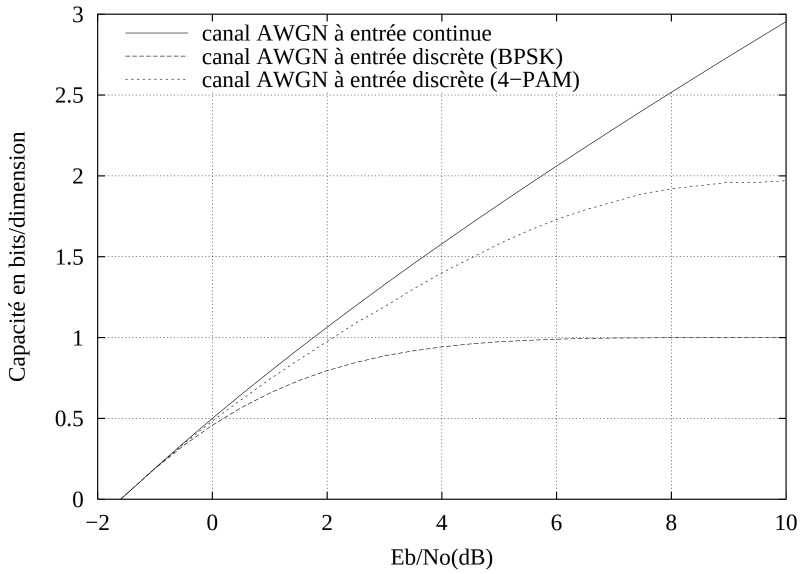


Fig. 1.2 – Capacité du canal AWGN

modulation discrète en entrée sont effectivement limitées par la taille de l'alphabet, $\log_2(2) = 1$ pour la BPSK et $\log_2(4) = 2$ pour la 4-PAM.

1.2.4 Expression de la capacité pour le canal de Rayleigh

Il est intéressant d'estimer l'impact des évanouissements de Rayleigh sur la capacité du canal. Nous faisons les hypothèses suivantes :

- l'état du canal est toujours parfaitement connu du récepteur (estimation du canal parfaite),
- le canal est entrelacé, de telle sorte qu'il peut être considéré sans mémoire (canal DMC, *Discrete Memoryless Channel*),
- les évanouissements de Rayleigh sont non-sélectifs en fréquence, *i.e.*, il n'y a pas d'interférence entre symboles (IES),
- l'évanouissement est supposé constant sur un temps symbole.

A l'instant i, la sortie du canal y_i ne dépend donc que du symbole en entrée du canal x_i, de l'évanouissement α_i et du bruit blanc additif gaussien z_i de variance N_0. Du fait de la détection cohérente, l'étude est restreinte à un évanouissement réel dont la distribution est donnée par :

$$p_\alpha(\alpha) = 2\alpha e^{-\alpha^2} \quad \text{avec} \quad E[\alpha^2] = 1 \ .$$

La capacité du canal de Rayleigh ainsi défini s'obtient en moyennant la capacité du canal gaussien par rapport à la distribution de l'évanouissement [12] [32] [47] [86] :

$$C = E_\alpha \left[C_{AWGN}(\alpha) \right] \ .$$

$C_{AWGN}(\alpha)$ désigne la capacité du canal gaussien obtenue en fixant la valeur de l'évanouissement α. Une démonstration de ce résultat est donnée en annexe A.

Les équations (1.10) et (1.11) donnent les expressions de la capacité d'un canal de Rayleigh à entrée discrète ou continue. La sortie du canal est supposée continue.

Capacité du canal de Rayleigh à entrée et sortie continues

L'information mutuelle est de nouveau maximisée pour une distribution gaussienne des symboles en entrée du canal. La capacité du canal de Rayleigh vérifie alors :

$$C = E_\alpha \left[\frac{1}{2} \log_2 \left(1 + \alpha^2 \frac{C E_b}{N_0} \right) \right] \ , \tag{1.10}$$

où E_α désigne l'espérance mathématique conditionnellement à α.

Capacité du canal à entrée discrète et sortie continue

Supposons en entrée du canal une modulation dont les Q symboles appartiennent à l'alphabet fini $\{x_0, x_1, \dots, x_{Q-1}\}$. Les symboles sont supposés équiprobables. L'expression de la capacité est :

$$C = \log_2 Q - E_\alpha \left[\frac{1}{Q} \sum_{q=0}^{Q-1} E_z \left[\log_2 \left(\sum_{q'=0}^{Q-1} \exp \left(- \frac{|z + \alpha(x_q - x_{q'})|^2 - |z|^2}{2N_0} \right) \right) \right] \right] \ , \tag{1.11}$$

où E_α et E_z désignent respectivement les espérances mathématiques conditionnellement aux variables aléatoires α et z.

En appliquant l'algorithme du point fixe pour résoudre les équations (1.10) et (1.11), nous obtenons les courbes de la figure 1.3. Pour le cas discret, la modulation BPSK a été choisie. La capacité du canal AWGN est également représentée pour référence.

La capacité du canal de Rayleigh s'annule pour $\left(\frac{E_b}{N_0}\right)_{dB} = -1.60$ dB, comme pour le canal AWGN. L'écart entre la courbe de capacité du canal AWGN et celle du canal de Rayleigh se creuse lorsque le rendement du code augmente, puis se stabilise à l'infini [54].

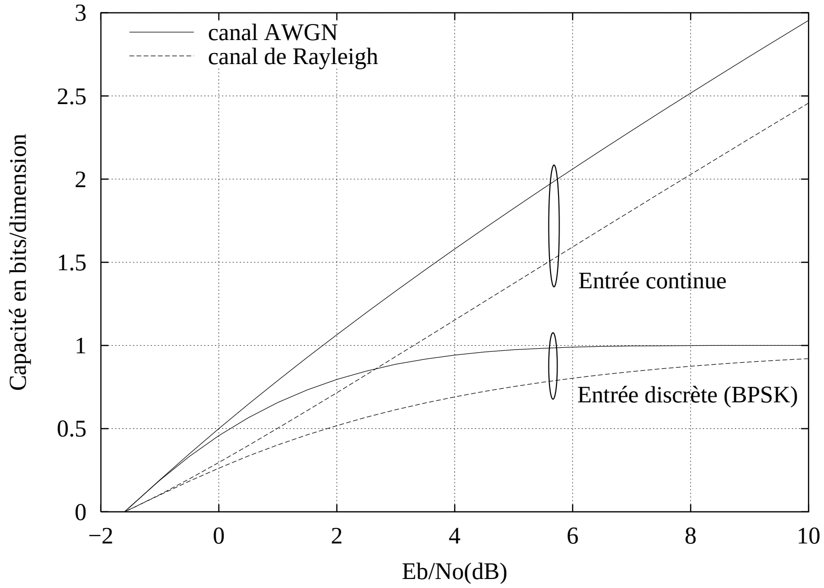


Fig. 1.3 – Capacité du canal de Rayleigh

1.2.5 Influence de la diversité sur la capacité du canal de Rayleigh

Considérons maintenant un système de communications apportant une diversité L. Ce système peut être modélisé par L canaux de diversité parallèles portant chacun le même signal d'information. Chaque canal est un canal de Rayleigh supposé non-sélectif en fréquence et variant lentement de sorte que l'évanouissement est considéré constant sur un temps symbole. Les évanouissements présents sur chaque canal sont supposés indépendants. Le signal est ensuite corrompu sur chaque canal par un bruit blanc additif gaussien de variance N_0.

Plaçons-nous dans le cas idéal où le rapport signal-à-bruit instantané total γ en réception est la somme des rapports signal-à-bruit instantanés γ_ℓ, $\ell = 1, \ldots, L$, sur chaque canal (*i.e.*, récepteur optimal) :

$$\gamma = \sum_{\ell=1}^{L} \gamma_\ell \ .$$

Pour évaluer la distribution du rapport signal-à-bruit γ du signal combiné, nous considérons sa fonction caractéristique. Comme γ et γ_ℓ, $\ell = 1, \ldots, L$, sont réels et positifs, la fonction de Laplace F peut être utilisée. Elle est définie par :

$$F(s) = E[e^{-s\gamma}] = \int_0^{+\infty} \exp(-s\gamma) p_\gamma(\gamma) d\gamma \ ,$$

où $p_\gamma(\gamma)$ désigne la densité de probabilité de γ et s est une variable complexe. Les variables aléatoires γ_ℓ sont indépendantes, $F(s)$ devient alors :

$$F(s) = E[e^{-s\sum_{\ell=1}^{L}\gamma_\ell}] = \prod_{\ell=1}^{L} E[e^{-s\gamma_\ell}] \ .$$

Par ailleurs, les rapports signal-à-bruit γ_ℓ sont distribués selon une loi de Rayleigh, donc :

$$\begin{aligned}
E[e^{-s\gamma_\ell}] &= \int_0^{+\infty} \exp(-s\gamma_\ell)\frac{1}{\Gamma_\ell}\exp\left(-\frac{\gamma_\ell}{\Gamma_\ell}\right)d\gamma_\ell \\
&= \frac{1}{\Gamma_\ell}\int_0^{+\infty}\exp\left(-\left(s+\frac{1}{\Gamma_\ell}\right)\gamma_\ell\right)d\gamma_\ell \\
&= \frac{1}{\Gamma_\ell}\left[-\frac{1}{s+\frac{1}{\Gamma_\ell}}\exp\left(-\left(s+\frac{1}{\Gamma_\ell}\right)\gamma_\ell\right)\right]_0^{+\infty} \\
&= \frac{1}{1+s\Gamma_\ell}\ ,
\end{aligned}$$

où Γ_ℓ désigne la valeur moyenne de γ_ℓ. Par conséquent :

$$F(s) = \prod_{\ell=1}^{L}\frac{1}{1+s\Gamma_\ell}\ .$$

Les L pôles de F se trouvent en $s = -\frac{1}{\Gamma_\ell}$, $\ell = 1, \ldots, L$. En considérant la transformée inverse de F, nous obtenons pour la distribution de γ :

$$p_\gamma(\gamma) = \frac{1}{2j\pi}\int_{c-j\infty}^{c+j\infty}\frac{\exp(s\gamma)}{\prod_{\ell=1}^{L}(1+s\Gamma_\ell)}ds \qquad c \geq 0\ .$$

Appliquons maintenant la méthode des résidus. En supposant que les valeurs moyennes Γ_ℓ sont toutes égales à $\bar{\Gamma}$, $-\frac{1}{\Gamma_\ell}$ est un pôle multiple d'ordre L. La distribution de γ s'écrit alors :

$$\begin{aligned}
p_\gamma(\gamma) &= \frac{1}{(L-1)!}\left[\frac{d^{L-1}}{ds^{L-1}}\left(\frac{1}{\bar{\Gamma}^L}\left(s+\frac{1}{\bar{\Gamma}}\right)^L\frac{\exp(s\gamma)}{(s+\frac{1}{\bar{\Gamma}})^L}\right)\right]_{s=-\frac{1}{\bar{\Gamma}}} \\
&= \frac{1}{(L-1)!}\frac{1}{\bar{\Gamma}^L}\left[\frac{d^{L-1}}{ds^{L-1}}(\exp(s\gamma))\right]_{s=-\frac{1}{\bar{\Gamma}}} \\
&= \frac{1}{(L-1)!}\frac{1}{\bar{\Gamma}^L}\left[\gamma^{L-1}\exp(s\gamma)\right]_{s=-\frac{1}{\bar{\Gamma}}} \\
&= \frac{1}{(L-1)!}\frac{\gamma^{L-1}}{\bar{\Gamma}^L}\exp\left(-\frac{\gamma}{\bar{\Gamma}}\right)\ .
\end{aligned}$$

Pour revenir à la distribution des évanouissements α, il suffit de poser $\gamma = \alpha^2$ (rapport signal-à-bruit normalisé), ce qui implique :

$$\bar{\Gamma} = \frac{E[\alpha^2]}{L}\ .$$

La densité de probabilité p_α des évanouissements vérifie ainsi

$$\begin{aligned}
p_\alpha(\alpha) &= p_\gamma(\gamma)\left|\frac{d\gamma}{d\alpha}\right| = 2\alpha\ p_\gamma(\gamma) \\
&= 2\alpha\frac{\alpha^{2(L-1)}}{(L-1)!}\frac{L^L}{E[\alpha^2]^L}\exp\left(-\frac{L\alpha^2}{E[\alpha^2]}\right) \\
&= \frac{2}{\Gamma(L)}\left(\frac{L}{E[\alpha^2]}\right)^L\alpha^{2L-1}\exp\left(-L\frac{\alpha^2}{E[\alpha^2]}\right)\ . \qquad (1.12)
\end{aligned}$$

où $\Gamma(x)$ désigne la fonction d'Euler définie par $\Gamma(x) = \int_0^{+\infty} e^{-t}t^{x-1}dt$. La loi de l'évanouissement α est donc une distribution de Nakagami d'ordre L.

La capacité du canal de Rayleigh avec une diversité L peut maintenant être évaluée à partir de cette distribution. La connaissance de l'évanouissement par le récepteur est toujours supposée parfaite. Les équations (1.10) et (1.11) restent valides, mais la distribution de Rayleigh est maintenant remplacée par une distribution de Nakagami d'ordre L.

La figure 1.4 met en évidence l'impact de la diversité sur la capacité du canal pour $L = 4$. Lorsque la diversité augmente, la capacité du canal de Rayleigh se rapproche de celle du canal AWGN. En effet d'après l'équation (1.12) lorsque L tend vers l'infini, la loi de Nakagami tend vers une loi gaussienne. Inversement, pour $L = 1$, la loi de Nakagami est équivalente à une distribution de Rayleigh. Ainsi par la suite, nous ne considérons que le cas du canal AWGN et celui du canal de Rayleigh, la capacité des canaux à diversité se situant entre celles de ces deux canaux.

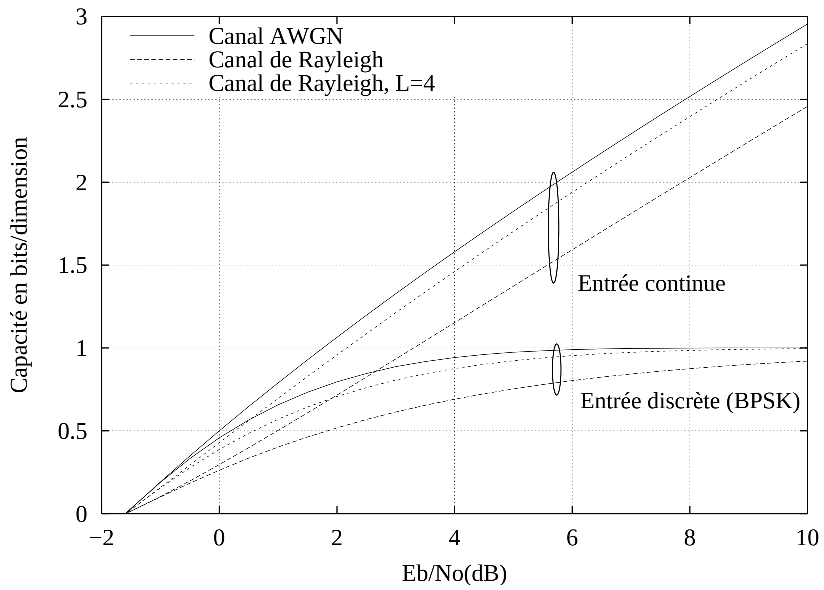


Fig. 1.4 – Capacité du canal de Rayleigh avec une diversité $L = 4$

Pour une étude plus complète de la capacité du canal de Rayleigh, avec ou sans estimation de canal (parfaite ou non), le lecteur est invité à se référer à [12] [17] [86].

La limite fixée par la capacité d'un canal donne une borne asymptotique des performances d'un système codé. Il s'agit de limites absolues, que seul un code de longueur infinie pourrait atteindre. En pratique, des codes de longueurs beaucoup plus petites sont utilisés. Les limites imposées par la théorie de l'information ne permettent alors pas de juger si, pour une longueur donnée, les codes choisis sont optimaux. Il est donc nécessaire de considérer d'autres bornes évaluées pour une longueur finie des codes. Une telle approche a été proposée par Shannon en

1959 [73].

1.3 Performances des codes optimaux de longueur finie

Le théorème de codage et sa réciproque établissent qu'il existe un code[2] de rendement R dont la probabilité d'erreur par mot tend vers 0 si et seulement si R est inférieur à la capacité C du canal et si la longueur N du code tend vers l'infini. Ce théorème s'appuie sur l'expression de la probabilité d'erreur par mot qui est majorée par $2^{-NE(R)}$, où $E(R)$ est l'exposant de Gallager [43]. Cette borne peut être utilisée pour évaluer les performances moyennes sur l'ensemble des codes aléatoires de rendement R et de longueur N fixés, mais l'évaluation de l'exposant de Gallager ne donne qu'une estimation peu précise de ces performances lorsque N est fini [28].

Plutôt que d'utiliser un tel raisonnement, Shannon propose une approche géométrique pour calculer la probabilité d'erreur optimale sur canal gaussien d'un code sphérique (*i.e.*, tous les mots de ce code ont exactement la même puissance) de longueur N fixée [73]. Cette approche lui permet de déduire une borne inférieure et une borne supérieure de la probabilité d'erreur optimale qui sont relativement proches pour de nombreuses valeurs de rendements R et de longueurs N.

Après une brève présentation des hypothèses et des notations nécessaires à la compréhension de cette étude, nous rappelons l'approche géométrique présentée par Shannon pour le canal gaussien. Les bornes obtenues sont mises en forme et utilisées pour donner une estimation de la dimension du code nécessaire pour atteindre des performances données lorsque le rendement du code est fixé. Par ailleurs, ces performances sont comparées aux performances optimales du calcul de la capacité présenté au paragraphe 1.2 lorsque la longueur du code tend vers l'infini. Ceci permet de mesurer l'influence de la dimension finie du code sur ses performances. Dans le cadre de cette étude, nous nous intéressons également aux performances des meilleurs codes connus en pratique (tels les turbo-codes [11]) et observons leur comportement par rapport à celui des codes optimaux de même longueur et de même rendement. Une analyse similaire a été conduite indépendamment par [28]. Nous proposons alors d'étudier l'évolution de ces performances optimales sur le canal de Rayleigh lorsqu'une modulation BPSK est utilisée.

1.3.1 Hypothèses et notations

Soit $\mathcal{C}(N, K)$ un code en blocs de dimension K et de longueur N. Dans l'approche de Shannon, le code constitue l'interface entre l'information et le canal, c'est-à-dire qu'il comprend à la fois le code correcteur d'erreurs et la modulation. La sortie du codeur $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ est par conséquent l'entrée même du canal. Le rendement du code R est égal à $\frac{K}{N}$ bits par dimension et peut ainsi être supérieur à 1. Si les alphabets en entrée et en sortie du code (de Shannon) sont de même tailles, alors le rendement est compris entre 0 et 1 (par exemple un code correcteur d'erreurs binaire et une entrée du canal binaire). Les mots de code sont de même énergie $2NE_c$, où E_c est l'énergie moyenne par bit codé. L'énergie moyenne par bit d'information est donnée par $E_c = RE_b$. Les symboles $c_i,\ i = 1, \ldots, N$ sont émis sur un canal sans mémoire dont la sortie est le vecteur réel $\mathbf{y} = (y_1, \ldots, y_N)$.

[2]le terme code inclut ici le code correcteur d'erreurs et la modulation

Deux types de canaux sont étudiés :

- le canal gaussien, pour lequel la sortie $\mathbf{y}$ est donnée par :

$$y_i = c_i + z_i \quad \forall\ i = 1, \dots, N,$$

  z_i représentant un bruit blanc gaussien réel de moyenne nulle et de variance N_0.
- le canal à évanouissements de Rayleigh, pour lequel la sortie $\mathbf{y}$ est donnée par :

$$y_i = \alpha_i\ c_i + z_i \quad \forall\ i = 1, \dots, N,$$

  où $\alpha_i \in [0, +\infty[$ est distribué selon une loi de Rayleigh :

$$p(\alpha_i) = 2\alpha_i e^{-\alpha_i} \ .$$

  Notons que $E[\alpha_i^2]$ est égal à 1 de sorte que la valeur de l'énergie moyenne par bit codé reste inchangée à la sortie du canal. Le cas où $\alpha_i = 1$ est équivalent à celui du canal gaussien.

En réception, le décodage est supposé à vraisemblance maximale, comme lors du calcul de la capacité du canal présenté au paragraphe 1.2.

Considérons maintenant la représentation géométrique du code $\mathcal{C}$ dans l'espace euclidien à N dimensions $\mathbb{R}^N$. Chaque composante d'un mot de code est représentée sur une dimension différente de $\mathbb{R}^N$. Les mots de code $\mathbf{c}$ sont supposés équiprobables et choisis de manière aléatoire à la surface de la sphère N-dimensionnelle, notée $\mathrm{Sph}(O, N)$, centrée sur l'origine O et de rayon $\sqrt{E} = \sqrt{2NE_c}$.

La zone de décision ou région de Voronoï d'un mot de code $\mathbf{c}$ pour un décodage à maximum de vraisemblance est un polyèdre à N dimensions. Il s'agit plus exactement d'une pyramide multi-dimensionnelle, notée $V(\mathbf{c})$, dont le sommet est confondu avec l'origine O ; cette pyramide est décrite par un angle solide $\Omega_\mathbf{c}$. Si Ω_N désigne l'angle solide d'une sphère de dimension N, alors

$$\sum_{\mathbf{c} \in \mathcal{C}} \Omega_\mathbf{c} = \Omega_N \ .$$

L'effet du bruit consiste à déplacer le mot de code $\mathbf{c}$ en dehors de sa région de Voronoï $V(\mathbf{c})$. Soit $Pe_\mathbf{c}$ la probabilité que $\mathbf{c}$ soit porté hors de la pyramide $V(\mathbf{c})$ par un bruit de distribution gaussienne sphérique de variance N_0 par dimension. La probabilité d'erreur par mot Pe_w pour le code $\mathcal{C}$ s'écrit alors :

$$Pe_\mathrm{w} = \frac{1}{M} \sum_{\mathbf{c} \in \mathcal{C}} Pe_\mathbf{c} \ , \tag{1.13}$$

où $M = 2^{NR}$ est le nombre de mots de code de $\mathcal{C}$.

La région de Voronoï $V(\mathbf{c})$ du mot de code $\mathbf{c}$ est approchée par un cône circulaire $\Lambda(\mathbf{c})$ délimité par une sphère de dimension $N-1$, dont le rayon $\rho_\mathbf{c}$ a été choisi de sorte que l'angle solide $\Omega_\mathbf{c}$ de la pyramide $V(\mathbf{c})$ est égal à celui du cône $\Lambda(\mathbf{c})$. Le demi-angle au sommet du cône $\Lambda(\mathbf{c})$ est noté $\theta_\mathbf{c}$. L'angle solide $\Omega_\mathbf{c}$ est ainsi une fonction de $\theta_\mathbf{c}$, et sera dorénavant noté $\Omega(\theta_\mathbf{c})$. La sphère de dimension $N-1$ et de rayon $\rho_\mathbf{c}$, notée $\mathrm{Sph}(\mathbf{a}, N-1)$, est l'intersection de la sphère

Sph(O, N) et d'un hyperplan $\mathcal{H}$. Le centre $\mathbf{a}$ de la sphère Sph$(\mathbf{a}, N-1)$ est en fait la projection du mot de code $\mathbf{c}$ sur l'hyperplan $\mathcal{H}$. Les modules des vecteurs $\mathbf{a}$ et $\mathbf{c}$ vérifient donc la relation suivante :

$$\|\mathbf{a}\| = \|\mathbf{c}\| \cos(\theta_{\mathbf{c}}) = \sqrt{E}\cos(\theta_{\mathbf{c}}) \ .$$

La figure 1.5 permet de visualiser les différents éléments que nous venons de décrire.

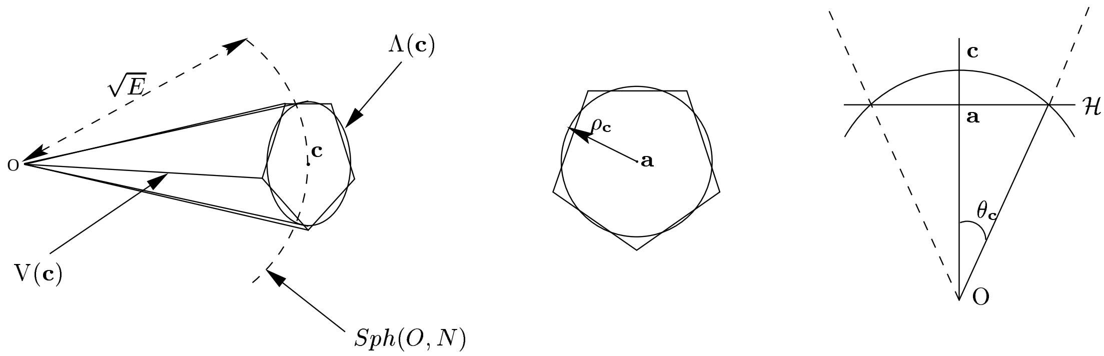


FIG. 1.5 – Représentation géométrique de l'angle solide associé à un mot de code.

1.3.2 Estimation de la probabilité d'erreur Pe_{w} pour le canal gaussien

Borne inférieure de la probabilité d'erreur

Quelle que soit la dimension N de l'espace euclidien, la densité de probabilité du bruit gaussien $\mathbf{z}$ est une fonction décroissante de la distance euclidienne d entre le mot reçu et le mot de code $\mathbf{c}$:

$$\frac{\partial p_{\mathbf{z}}}{\partial V} = \frac{1}{(2\pi N_0)^{\frac{N}{2}}} \exp\left(-\frac{d^2}{2N_0}\right) \ . \tag{1.14}$$

En moyenne, les éléments de la pyramide $V(\mathbf{c})$ sont plus éloignés de l'origine que les éléments du cône $\Lambda(\mathbf{c})$; il peut être aisément montré que :

$$\int_{\Lambda(\mathbf{c})} \frac{\partial p_{\mathbf{z}}}{\partial V} dV \geq \int_{V(\mathbf{c})} \frac{\partial p_{\mathbf{z}}}{\partial V} dV \ . \tag{1.15}$$

La probabilité pour un mot de code $\mathbf{c}$ d'être déplacé par le bruit à l'intérieur du cône $\Lambda(\mathbf{c})$ est ainsi supérieure à la probabilité que ce mot de code soit déplacé à l'intérieur de sa région de Voronoï $V(\mathbf{c})$. La borne de la probabilité d'erreur $Pe_{\mathbf{c}}$ calculée à partir de l'espace délimité par $\Lambda(\mathbf{c})$ est donc une borne inférieure.

Soit $Q(\Omega)$ la probabilité qu'un mot de code situé sur l'axe d'un cône d'angle solide $\Omega = \Omega(\theta)$ soit déplacé à l'extérieur de ce cône. Pour le mot de code $\mathbf{c}$, cette probabilité vérifie :

$$Q(\Omega_{\mathbf{c}}) = 1 - \int_{\Lambda(\mathbf{c})} \frac{\partial p_z}{\partial V} dV \ .$$

Par conséquent en utilisant l'égalité (1.13) et la définition ci-dessus de $Q(\Omega_{\mathbf{c}})$, la borne inférieure suivante est obtenue pour la probabilité d'erreur $P\mathrm{e}_{\mathrm{w}}$ [73] :

$$\frac{1}{M}\sum_{\mathbf{c}\in\mathcal{C}} Q(\Omega_{\mathbf{c}}) \leq P\mathrm{e}_{\mathrm{w}} \ . \tag{1.16}$$

La densité du bruit est une fonction décroissante de la distance, par conséquent la fonction $Q(\Omega)$ est une fonction convexe. Nous avons donc en appliquant l'inégalité de Jensen :

$$Q\left(\frac{\Omega_N}{M}\right) \leq P\mathrm{e}_{\mathrm{w}} \ .$$

De manière équivalente, nous pouvons écrire

$$Q(\theta_0) \leq P\mathrm{e}_{\mathrm{w}} \ , \tag{1.17}$$

où θ_0 désigne le demi-angle au sommet du cône d'angle solide $\Omega(\theta_0) = \frac{\Omega_N}{M}$.

L'équation (1.17) montre que la probabilité d'erreur optimale d'un code sur le canal gaussien est minorée par la probabilité d'erreur d'un code qui serait construit en répartissant de manière uniforme les M mots de code à la surface de la sphère N-dimensionnelle, de telle sorte que les cônes entourant chacun de ces mots de code soient identiques et de même angle solide $\frac{\Omega_N}{M}$.

Borne supérieure de la probabilité d'erreur

Shannon a également évalué une borne supérieure de la probabilité d'erreur optimale. Cette borne est fondée sur le codage aléatoire. On considère l'ensemble des codes $\mathcal{C}$ obtenus en plaçant M points aléatoirement sur la sphère de dimension N de rayon $\sqrt{E}$. Les M points sont placés indépendamment les uns des autres et sont équiprobables. Par symétrie, la probabilité d'erreur moyenne de l'ensemble ainsi défini est égale à M fois la probabilité d'erreur moyenne évaluée en un point particulier noté $\mathbf{c}$.

La probabilité que le mot de code $\mathbf{c}$ soit déplacé par le bruit en un point situé dans la région $\mathrm{Rég}(\theta, \theta + d\theta)$ délimitée par les cônes de demi-angles au sommet respectifs θ et $\theta + d\theta$ est égale à $-dQ(\theta)$. Considérons le cône de demi-angle au sommet θ autour d'un point reçu ainsi placé. Si ce cône ne contient aucun mot de code, le signal reçu est correctement décodé en $\mathbf{c}$. Sinon, cela signifie que d'autres points sont plus proches du mot reçu que le mot de code $\mathbf{c}$ et par conséquent le processus de décodage donnera un résultat erroné. La probabilité d'avoir deux ou plusieurs mots à la même distance exactement est supposée nulle, et peut ainsi être ignorée.

La probabilité qu'un mot de code soit dans le cône de demi-angle au sommet θ autour du point reçu est égale à $\frac{\Omega(\theta)}{\Omega_N}$, où Ω_N désigne l'angle solide de l'espace total ($\Omega_N = \Omega(\pi)$). Ainsi, la probabilité qu'un mot de code soit en dehors de ce cône est $1 - \frac{\Omega(\theta)}{\Omega_N}$. Comme chaque mot de code est placé indépendamment des autres, la probabilité que $M-1$ mots de code (différents du message $\mathbf{c}$) soient en dehors du cône est $\left(1 - \frac{\Omega(\theta)}{\Omega_N}\right)^{M-1}$; il s'agit de la probabilité de décoder correctement le message $\mathbf{c}$ pour ce signal reçu. Or, la probabilité de recevoir un tel signal est

$-dQ(\theta)$, par conséquent la probabilité d'erreur conditionnelle sachant que le mot de code $\mathbf{c}$ a été émis, et que le message reçu se situe dans la région $\text{Rég}(\theta, \theta + d\theta)$ est égale à

$$-\frac{1}{M}\left(1-\left(1-\frac{\Omega(\theta)}{\Omega_N}\right)^{M-1}\right)dQ(\theta)\ .$$

En moyennant sur l'ensemble des mots de code et sur tous les déplacements du bruit, la probabilité d'erreur moyenne Pe'_{w} s'écrit :

$$Pe'_{\text{w}} = -\int_0^{\pi}\left(1-\left(1-\frac{\Omega(\theta)}{\Omega_N}\right)^{M-1}\right)dQ(\theta)\ . \tag{1.18}$$

Il s'agit d'une formule exacte de la probabilité d'erreur moyenne sur l'ensemble des codes aléatoires. Ainsi, il existe un code particulier $\mathcal{C}_0$ tel que sa probabilité d'erreur par mot soit inférieure à cette probabilité d'erreur moyenne. Nous notons Pe_w la probabilité d'erreur du code optimal.

Afin de faciliter le calcul de la borne (1.18), Shannon propose deux approximations [73] :

$$\left(1-\frac{\Omega(\theta)}{\Omega_N}\right)^{M-1} \leq 1$$

et

$$1-\left(1-\frac{\Omega(\theta)}{\Omega_N}\right)^{M-1} \leq (M-1)\frac{\Omega(\theta)}{\Omega_N}\ ,$$

car $(1-x)^n \geq 1-nx$ pour des valeurs positives de x et n. Ainsi pour un angle θ_1 tel que $0 \leq \theta_1 \leq \pi$, l'équation (1.18) devient :

$$Pe_{\text{w}} \leq -\int_0^{\theta_1} M\frac{\Omega(\theta)}{\Omega_N}dQ(\theta) - \int_{\theta_1}^{\pi} dQ(\theta)\ .$$

Si l'on choisit pour θ_1 la même valeur que pour la borne inférieure, c'est-à-dire $\theta_1 = \theta_0$ tel que $\frac{\Omega(\theta_0)}{\Omega_N} = \frac{1}{M}$, alors la probabilité d'erreur optimale Pe_w est bornée par :

$$Pe_{\text{w}} \leq Q(\theta_0) - \frac{M}{\Omega_N}\int_0^{\theta_0} \Omega(\theta)dQ(\theta)\ . \tag{1.19}$$

avec $Q(\pi) = 0$.

Notons que Battail *et al.* [4] ont développé des expressions littérales plus élaborées de la probabilité d'erreur exacte Pe'_{w} donnée par l'équation (1.18). Ils ont pu ainsi évaluer les performances des codes optimaux sur canal AWGN sans avoir recours aux approximations proposées par Shannon, mais en effectuant directement le calcul numérique des expressions exactes de la probabilité d'erreur.

Expression de $Q(\theta)$

L'expression de la borne supérieure comme celle de la borne inférieure de la probabilité d'erreur Pe_{w} font toutes deux intervenir la fonction $Q(\theta)$. Il est donc nécessaire d'expliciter plus précisément cette fonction.

Considérons le schéma de la figure 1.6 représentant une section du cône de demi-angle au sommet θ dans l'espace euclidien à N dimensions. Dans l'équation (1.14), dV représente le volume

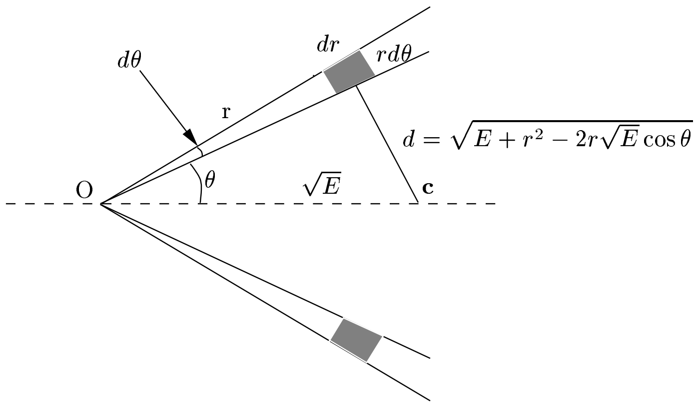


Fig. 1.6 – Plan du cône de demi-angle au sommet θ.

élémentaire d'un anneau d'épaisseur dr, de largeur $rd\theta$ et placé à une distance d du mot de code $\mathbf{c}$ telle que $d^2 = E + r^2 - 2r\sqrt{E}\cos\theta$. L'expression du volume dV est :

$$dV = \frac{(N-1)\pi^{\frac{N-1}{2}}(r\sin\theta)^{N-2}}{\Gamma(\frac{N+1}{2})}\ r\ dr\ d\theta\ .$$

Par conséquent, la densité de la probabilité de déplacer $\mathbf{c}$ dans cet élément de cône à l'aide d'un bruit de distribution gaussienne est :

$$-d^2Q(r,\theta) = \frac{1}{(2\pi N_0)^{N/2}} \exp\left(-\frac{E + r^2 - 2r\sqrt{E}\cos\theta}{2N_0}\right) \frac{(N-1)\pi^{\frac{N-1}{2}}(r\sin\theta)^{N-2}}{\Gamma\left(\frac{N+1}{2}\right)} r dr d\theta\ ,$$

d'où en intégrant selon la variable r entre 0 et $+\infty$,

$$-\frac{dQ(\theta)}{d\theta} = \frac{(N-1)\exp\left(-\frac{E(\sin\theta)^2}{2N_0}\right)(\sin\theta)^{N-2}}{(2N_0)^{N/2}\sqrt{\pi}\Gamma\left(\frac{N+1}{2}\right)} \int_0^{+\infty} \exp\left(-\frac{(r-\sqrt{E}\cos\theta)^2}{2N_0}\right) r^{N-1} dr\ . \quad (1.20)$$

En utilisant le lemme de David et Kruskal dont l'énoncé est le suivant :

$$\int_0^{+\infty} z^{\nu} \exp\left(-\frac{1}{2}z^2 + z\sqrt{\nu+1}w\right) dz \sim T\sqrt{2\pi}\left(\frac{\bar{z}}{e}\right)^{\nu} \exp\left(\frac{1}{2}\bar{z}^2\right) \qquad \text{lorsque } \nu \to +\infty,$$

pour w fixé et

$$T = \left[1 + \frac{1}{4}\left(\sqrt{w^2+4} - w\right)^2\right]^{-\frac{1}{2}}, \qquad \bar{z} = \frac{1}{2}\ w\ \sqrt{\nu+1} + \sqrt{\frac{1}{4}(\nu+1)w^2 + \nu},$$

et en posant

$$\left\{\begin{array}{lcl} z & = & r, \\ w & = & \sqrt{\frac{E}{N_0}}\cos\theta = \sqrt{\frac{2NE_c}{N_0}}\cos\theta, \\ \nu & = & N-1, \end{array}\right.$$

une approximation de l'équation (1.20) est :

$$-dQ(\theta) \sim \frac{(N-1)}{\sqrt{N\pi}\sqrt{1+G^2\sin^2\theta}}\left[G\sin\theta\exp\left(-\frac{E_c}{N_0}+\frac{1}{2}\sqrt{\frac{2E_c}{N_0}}G\cos\theta\right)\right]^N d\theta, \tag{1.21}$$

avec

$$G = \frac{1}{2}\left[\sqrt{\frac{2E_c}{N_0}}\cos\theta + \sqrt{\frac{2E_c}{N_0}\cos^2\theta + 4}\right] .$$

La fonction $Q(\theta)$ peut alors être évaluée à partir de l'approximation (1.21) et de l'équation suivante :

$$Q(\theta) = \int_\theta^{\frac{\pi}{2}} -dQ(\phi) + Q\left(\frac{\pi}{2}\right), \tag{1.22}$$

avec $Q\left(\frac{\pi}{2}\right) = \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E}{2N_0}}\right)$, où erfc désigne la fonction d'erreur complémentaire.

En combinant les relations (1.21) et (1.22), la fonction $Q(\theta)$ évaluée au point θ_0 s'écrit :

$$Q(\theta_0) \sim \frac{1}{\sqrt{N\pi}}\frac{1}{\sqrt{1+G^2\sin\theta_0}}\frac{\left[G\sin\theta_0\exp\left(-\frac{E_c}{N_0}+\frac{1}{2}\sqrt{\frac{2E_c}{N_0}}G\cos\theta_0\right)\right]^N}{\sqrt{\frac{2E_c}{N_0}}G\sin^2\theta_0 - \cos\theta_0}, \tag{1.23}$$

avec $E_c = RE_b$.

Calcul de l'angle solide $\Omega(\theta)$ et de θ_0

Pour évaluer la borne supérieure de Pe_w, il est nécessaire de disposer d'une expression littérale de l'angle solide $\Omega(\theta)$ en fonction de θ. A partir de cette expression, il sera alors possible de déduire l'angle θ_0 grâce à l'égalité :

$$2^{NR} = \frac{\Omega_N}{\Omega(\theta_0)}, \tag{1.24}$$

où R et N sont fixés. Cet angle θ_0 est ensuite utilisé pour le calcul de la borne supérieure et de la borne inférieure de la probabilité d'erreur $P\mathrm{e}_{\mathrm{w}}$.

L'angle solide Ω_N est, par définition, égal à la surface de la sphère de dimension N et de rayon unité. La surface d'une sphère de rayon r et de dimension N est donnée par :

$$\mathrm{S}(r,N) = N\ \mathrm{Vol}(1,N)\ r^{N-1},$$

où $\mathrm{Vol}(1,N)$ est le volume d'une sphère N-dimensionnelle de rayon 1. Ainsi,

$$\Omega_N = N\ \mathrm{Vol}(1,N)\ .$$

Le volume d'une sphère N-dimensionnelle de rayon r est égal à [23] :

$$\mathrm{Vol}(r,N) = \frac{\pi^{N/2}\ r^N}{\Gamma\left(\frac{N}{2}+1\right)},$$

où $\Gamma(x)$ est la fonction d'Euler définie par $\Gamma(x) = \int_0^{+\infty} e^{-t} t^{x-1} dt$. Pour de grandes valeurs de N, en appliquant l'approximation de Stirling [1], ce volume devient :

$$\mathrm{Vol}(r, N) \sim \frac{\left(\frac{2e\pi}{N}\right)^{N/2} r^N}{\sqrt{N\pi}} \; .$$

L'angle solide $\Omega(\theta)$ peut être calculé en sommant les surfaces d'anneaux élémentaires $d\Omega(\phi)$ définies par :

$$d\Omega(\phi) = \mathrm{S}(\sin\phi, N-1)d\phi = \Omega_{N-1}(\sin\phi)^{N-2}d\phi,$$

d'où

$$\Omega(\theta) = \frac{(N-1)\pi^{\frac{N-1}{2}}}{\Gamma\left(\frac{N+1}{2}\right)} \int_0^{\theta} (\sin\phi)^{N-2} d\phi \; . \tag{1.25}$$

En combinant les équations (1.24) et (1.25), le rendement du code s'écrit en fonction du demi-angle au sommet à l'aide de la relation :

$$R = \frac{1}{N} \log_2 \left(\frac{\sqrt{\pi}\,\Gamma\left(\frac{N-1}{2}\right)}{\Gamma\left(\frac{N}{2}\right) \int_0^{\theta_0} (\sin\phi)^{N-2} d\phi} \right) \; .$$

Asymptotiquement, lorsque $N \gg 1$, les expressions de l'angle solide et du rendement deviennent :

$$\Omega(\theta) \sim \frac{1}{\pi\sqrt{2}} \left(\frac{2e\pi}{N}\right)^{N/2} \frac{(\sin\theta)^{N-1}}{\cos\theta}, \tag{1.26}$$

$$2^{NR} \sim \frac{\sqrt{2\pi N}\sin\theta_0\cos\theta_0}{(\sin\theta_0)^N} \; . \tag{1.27}$$

Lazic *et al.* [53] ont développé une expression plus précise de l'angle solide $\Omega(\theta)$. Cependant les résultats obtenus à l'aide de cette expression sont très proches de ceux présentés par Shannon. Nous utilisons donc par la suite les expressions asymptotiques proposées par Shannon et données par les équations (1.26) et (1.27).

En résolvant l'équation (1.27) pour une longueur N et un rendement R du code donnés, nous obtenons la valeur de θ_0 à utiliser dans la formule (1.23) pour évaluer une borne inférieure de la probabilité d'erreur optimale par mot Pe_{w}. La figure 1.7 représente l'évolution du rendement du code R en fonction de l'angle θ_0, lorsque $N \gg 1$. Cette évolution est indépendante de la longueur du code N ; asymptotiquement en effet le rendement du code est relié à l'angle θ_0 par l'expression :

$$R \sim -\log_2(\sin\theta_0) \tag{1.28}$$

Notons également que pour R fixé, l'approximation (1.26) indique que l'angle solide $\Omega(\theta)$ tend exponentiellement vers 0 lorsque la longueur du code N tend vers l'infini. Ceci signifie que les mots de code sont de plus en plus proches lorsque la dimension de l'espace augmente.

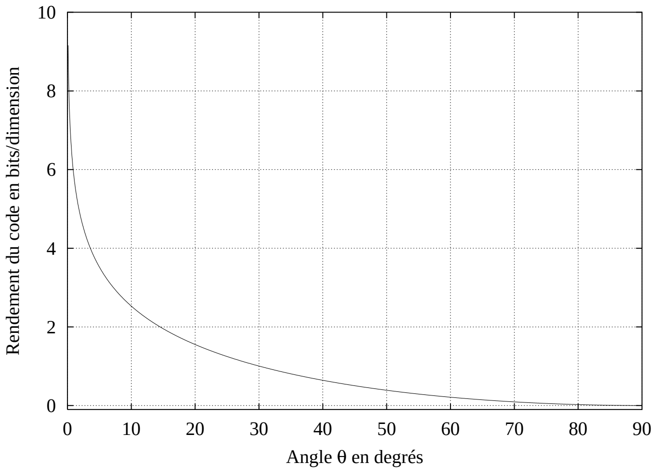


FIG. 1.7 – Evolution du rendement R du code en fonction de l'angle θ_0

Comparaison des bornes (1.17) et (1.19)

La borne inférieure (1.17) et la borne supérieure (1.19) de la probabilité d'erreur optimale évaluées par Shannon sont identiques à un facteur multiplicatif près lorsque le rendement R se situe entre un rendement R_c, appelé rendement critique [73][85], et la capacité C du canal. Ce facteur multiplicatif, noté $f(A, \theta_0)$, est indépendant de la longueur du code N :

$$f(A, \theta_0) = 1 - \frac{\cos\theta_0 - AG\sin^2\theta_0}{2\cos\theta_0 - AG\sin^2\theta_0} \ ,$$

où

$$A = \sqrt{2R\frac{E_b}{N_0}} \quad \text{et} \quad G = \frac{1}{2}\left[A\cos\theta + \sqrt{A^2\cos^2\theta + 4}\right] \ .$$

Lorsque l'angle θ_0 tend vers l'angle θ_C correspondant à la capacité du canal (évalué en utilisant l'approximation (1.28)), le facteur $f(A, \theta_0)$ tend vers 1, les deux bornes sont confondues. Inversement, lorsque θ_0 tend vers l'angle θ_{R_c} correspondant au rendement critique, le facteur multiplicatif $f(A, \theta_0)$ tend vers l'infini, et les deux bornes divergent. L'angle θ_{R_c} est en effet solution de l'équation :

$$2\cos\theta - AG\sin^2\theta = 0.$$

Lorsque $A \to 0$, le rapport entre la capacité C et le rendement critique R_c tend vers 4. Lorsque $A \to +\infty$, l'écart entre R_c et C est de l'ordre de 0.5 bit/dimension [73].

Le rendement critique R_c est, comme la capacité, une fonction du rapport signal-à-bruit $\frac{E_b}{N_0}$. Considérons un rapport signal-à-bruit pour lequel le rendement R et la capacité C sont proches l'un de l'autre. D'après ce qui précède, pour un tel rapport signal-à-bruit, les bornes inférieure et supérieure de Shannon sont confondues et donnent précisément la valeur de la probabilité d'erreur optimale. Si l'on augmente le rapport signal-à-bruit, la capacité du canal s'éloigne peu à peu du rendement R qui se rapproche, quant à lui, du rendement critique R_c. Pour des contraintes raisonnables en termes de probabilité d'erreur dans le domaine du codage (par exemple $Pe_w > 10^{-10}$) et pour des longueurs de code $N > 100$, les rapports signal-à-bruit considérés permettent au rendement R de rester proche de la capacité du canal [74][70] et d'utiliser ainsi indifféremment la borne supérieure ou la borne inférieure de Shannon comme estimation de la probabilité d'erreur optimale. Dans la section suivante, nous ne considérerons donc que la fonction $Q(\theta)$ pour évaluer la probabilité d'erreur optimale.

Notons que pour $R_c < R \leq C$, les bornes supérieures et inférieures de la probabilité d'erreur Pe_w peuvent s'écrire sous la forme $e^{-NE(R)+o(N)}$ où $E(R)$ est déterminé de manière exacte par ces bornes et représente une mesure de la rapidité avec laquelle la probabilité d'erreur Pe_w tend vers 0 lorsque N tend vers l'infini. Le terme $o(N)$ désigne un terme négligeable devant N. L'exposant $E(R)$ est positif pour $R < C$ et s'annule en $R = C$. Il peut, par ailleurs, être montré que la probabilité d'erreur optimale Pe_w tend vers 1 lorsque le rendement du code R est supérieur à la capacité du canal. Le code dont la probabilité d'erreur est Pe_w atteint donc la capacité du canal gaussien.

1.3.3 Résultats numériques pour le canal AWGN

Les bornes proposées par Shannon ont été évaluées pour un code sphérique aléatoire sur un canal AWGN à entrée continue. En pratique, nous nous intéressons à la pertinence de ces bornes lorsque l'entrée du canal est contrainte, par exemple par une modulation binaire ou plus généralement par une modulation à amplitude constante. Battail *et al.* [4] ont montré que la distribution des distances euclidiennes obtenues avec un codage aléatoire et une modulation à amplitude constante est asymptotiquement identique à la distribution des distances euclidiennes obtenues par le codage aléatoire à énergie constante proposé par Shannon. Ainsi, les bornes évaluées par Shannon donnent-elles une bonne prévision des performances optimales accessibles lorsqu'une modulation à amplitude constante est considérée.

Nous nous intéressons dans ce paragraphe aux rendements R du code compris entre 0 et 1. La figure 1.8 représente le rapport signal-à-bruit $\frac{E_b}{N_0}$ nécessaire pour atteindre une probabilité d'erreur par mot égale à 10^{-5} en fonction de la dimension $K = R\ N$ du code. Plusieurs rendements sont représentés : $R = \frac{1}{2},\ \frac{1}{3},\ \frac{1}{4}$. Les limites correspondantes données par la capacité du canal sont également mentionnées. Nous constatons que lorsque la dimension K du code diminue, les écarts entre les performances des codes optimaux et les limites données par la capacité du canal AWGN augmentent rapidement et sont de l'ordre de 5 dB (respectivement, 2.5 dB) pour les différents rendements considérés lorsque $K = 10$ (respectivement, $K = 100$). La capacité du canal donne alors une estimation beaucoup trop optimiste des performances optimales auxquelles on peut s'attendre avec un code de cette dimension. Inversement, lorsque la dimension du code est supérieure à 10^6, la capacité du canal donne une estimation très précise des limites du code envisagé, et l'écart est inférieur à 1/20-ème de dB.

La figure 1.8 peut se lire de différentes manières. Elle permet par exemple d'évaluer l'apport d'un code de rendement inférieur pour une dimension K du code donnée ou encore pour un code de rendement R donné, de mesurer le gain obtenu en augmentant la dimension du code. Ainsi,

- pour obtenir une probabilité d'erreur de 10^{-5} à $\frac{E_b}{N_0} = 0.5$ dB, il faut considérer un code de rendement $R = \frac{1}{4}$ et de dimension $K = 500$ ou un code de rendement $R = \frac{1}{2}$ mais de dimension $K = 5000$.

– pour une dimension du code $K = 1000$ fixée, il faut un rapport signal-à-bruit égal à 0.80 dB pour le code de rendement $R = \frac{1}{2}$ afin d'atteindre une probabilité d'erreur égale à 10^{-5}, tandis que le code de rendement $R = \frac{1}{4}$ n'a besoin que de $\frac{E_b}{N_0} = 0.30$ dB pour obtenir la même probabilité d'erreur.

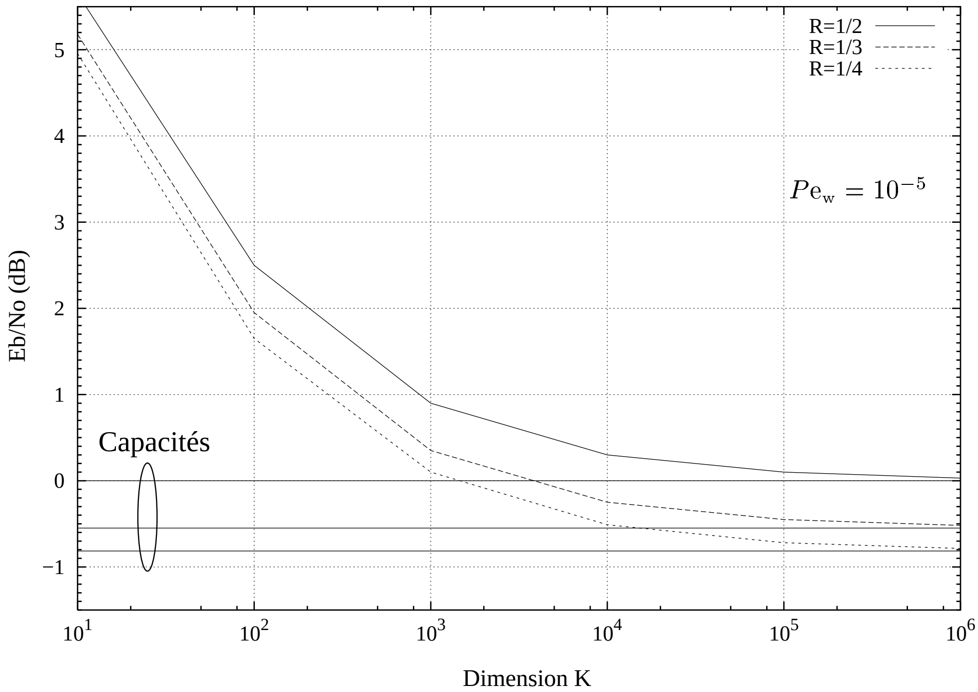


FIG. 1.8 – Evolution du rapport $\frac{E_b}{N_0}$ en fonction de la dimension du code K pour le canal AWGN

La figure 1.9 représente l'évolution de la probabilité d'erreur par mot en fonction de la dimension K pour un code de rendement $R = \frac{1}{3}$ et pour différents rapports signal-à-bruit : $\frac{E_b}{N_0} = -0.40,\ -0.20,\ 0.0,\ 0.20,\ 0.40,\ 0.60,\ 0.80,\ 1.00$ dB. Par exemple, lorsque le rapport signal-à-bruit est égal à -0.20 dB, la probabilité d'erreur est divisée par un facteur 10^5 lorsque la dimension est multipliée par 10. Nous retrouvons également sur cette figure les résultats donnés par la figure 1.8, à savoir, le rapport signal-à-bruit nécessaire pour atteindre une probabilité d'erreur donnée, la dimension du code K étant fixée.

Les performances présentées sur les figures 1.8 et 1.9 donnent une estimation de la probabilité d'erreur *par mot* optimale en fonction du rapport $\frac{E_b}{N_0}$ et de la dimension du code K. Nous nous intéressons maintenant à une estimation de la probabilité d'erreur *par bit* optimale. Nous considérons une modulation BPSK et un code correcteur d'erreurs binaire. Le rendement R désigne alors le rendement du code correcteur d'erreurs.

Pour un code de distance de Hamming minimale $d_{\mathrm{H\,min}}$, une approximation de la probabilité d'erreur par bit est donnée par :

$$Pe_{\mathrm{b}} \approx \frac{d_{\mathrm{H\,min}}}{N} Pe_{\mathrm{w}} \ .$$

Gallager [42] a montré que lorsque N est suffisamment grand, les codes binaires aléatoires ap-

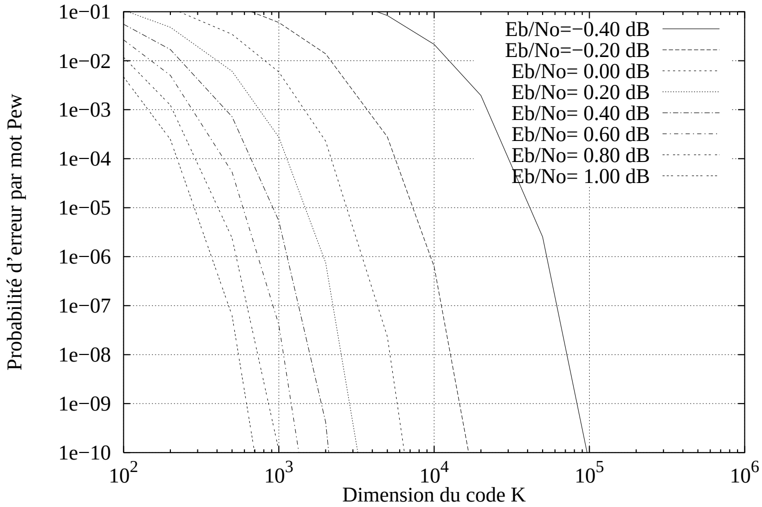


FIG. 1.9 – Evolution de la probabilité d'erreur Pe_w en fonction de la dimension K, $R = \frac{1}{3}$

prochent la borne de Gilbert-Varshamov [60], *i.e.*,

$$H_2\left(\delta = \frac{d_{\mathrm{H\,min}}}{N}\right) = 1 - R,$$

où H_2 désigne l'entropie binaire et $0 < \delta \leq \frac{1}{2}$ est la distance minimale normalisée du code. La probabilité d'erreur par bit optimale, pour un code aléatoire binaire, peut alors s'écrire :

$$P\mathrm{e}_{\mathrm{b}} \approx H^{-1}_{2|[0,1/2]}(1 - R) \times P\mathrm{e}_{\mathrm{w}} \ . \tag{1.29}$$

Quelques valeurs de δ sont données dans la table 1.1 pour des rendements R usuels.

R	$\delta = H^{-1}_{2\|[0,1/2]}(1 - R)$
1/7	0.2812
1/4	0.2145
1/3	0.1739
1/2	0.1100
2/3	0.0615

TAB. 1.1 – Borne de Gilbert-Varshamov sur la distance minimale normalisée

La figure 1.10 présente les probabilités d'erreur par bit obtenues en considérant l'approximation (1.29) pour un code aléatoire binaire. Trois dimensions de code sont considérées, $K = 128,\ 1024,\ 65536$, et le rendement est $R = \frac{1}{2}$. Sur cette figure se trouvent également, à titre de comparaison, les performances atteintes par un turbo-code de mêmes dimensions et de même rendement. Ces derniers ont été introduits par Berrou *et al.* [11] en 1993 et présentent des performances très proches des limites données par la capacité du canal lorsque leur dimension tend vers l'infini. La structure et les principales propriétés des turbo-codes sont abordées au chapitre 2. Le turbo-code représenté sur la figure 1.10 est une concaténation parallèle de deux codes convolutifs récursifs systématiques identiques, de rendement $\frac{1}{2}$ et de polynômes générateurs $(37, 21)$. L'entrelaceur est un entrelaceur pseudo-aléatoire.

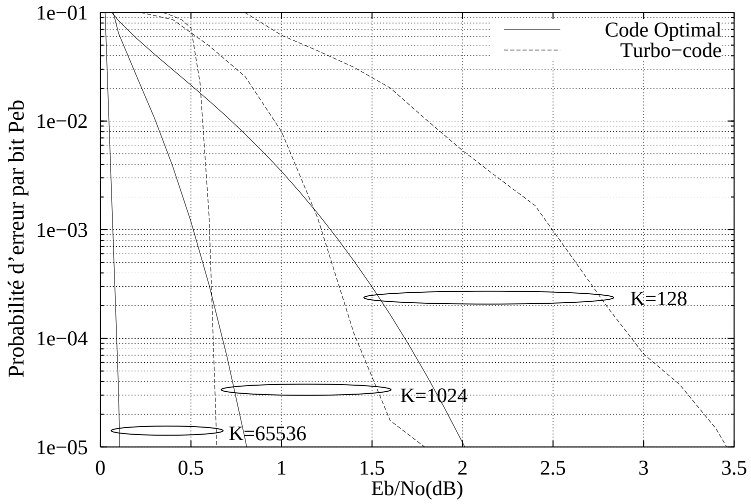


Fig. 1.10 – Comparaison des performances des turbo-codes avec les performances des codes optimaux pour $R = \frac{1}{2}$

Pour les dimensions du turbo-code considérées, l'écart avec les performances optimales reste constant pour les rapports signal-à-bruit représentés. Pour une probabilité d'erreur par bit $Pe_b = 10^{-4}$, cet écart est de 0.5 dB pour la dimension $K = 65536$, 0.7 dB pour $K = 1024$ et enfin 1.3 dB pour $K = 128$. L'évolution des performances des turbo-codes en fonction de la dimension s'explique par le gain d'entrelacement qu'ils présentent. En effet, nous verrons au chapitre 2 que la probabilité d'erreur par bit des turbo-codes parallèle, pour un rapport signal-à-bruit fixé, se comporte en $\frac{1}{N_\pi}$, où N_π est la taille de l'entrelaceur, c'est-à-dire la dimension K du code pour un turbo-code parallèle. Au-delà de la probabilité d'erreur 10^{-5}, l'écart entre les performances optimales et les performances des turbo-codes se creuse, du fait du changement de pente observé sur la courbe de probabilité d'erreur par bit de ces derniers et qui provient de leur distance minimale.

Nous proposons maintenant d'analyser l'évolution de ces performances optimales en présence d'évanouissements de Rayleigh. Nous avons vu que les codes ayant de telles performances sont des codes atteignant la capacité de Shannon lorsque leur longueur tend vers l'infini, c'est-à-dire que leur probabilité d'erreur tend vers 0 lorsque leur rendement est inférieur à la capacité du canal. Sur le canal de Rayleigh, bien que nous ne puissions affirmer que les performances de ces codes restent optimales, ils continuent cependant à atteindre la capacité du canal [71][12] : l'évaluation de leurs performances donne donc une référence quant aux probabilités d'erreur pouvant être obtenues avec de tels codes.

1.3.4 Extension au canal de Rayleigh

Nous supposons dans cette partie que la modulation et le code correcteur d'erreurs considérés sont binaires. Dans ce cas, le rendement du code $\mathcal{C}$ de Shannon (qui regroupe le code correcteur d'erreurs et la modulation) est compris entre 0 et 1 et désigne également le rendement du code correcteur d'erreurs. Les mots de code de $\mathcal{C}$ sont sélectionnés parmi les sommets du cube défini par l'ensemble $\{\pm\sqrt{2E_c}\}^N$ de telle sorte qu'ils soient équiprobables et répartis le plus uniformément possible à la surface de la sphère $\mathrm{Sph}(O,N)$. Nous avons vu dans la section précédente qu'une estimation de la probabilité d'erreur d'un tel code est donnée par la fonction $Q(\theta_0)$, où θ_0 est le demi-angle au sommet du cône d'angle solide $\Omega(\theta_0) = \frac{\Omega_N}{M}$ avec $M = 2^{NR}$

Un mot de code $\mathbf{c} = (c_1, \ldots, c_N)$ du code sphérique $\mathcal{C}$ est affecté par un évanouissement de Rayleigh, noté $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$, qui agit selon les axes du cube $\{\pm\sqrt{2E_c}\}^N$. Par symétrie du problème, la probabilité d'erreur moyenne par mot $P\mathrm{e}_{\mathrm{w}}$ est égale à la probabilité d'erreur conditionnelle sachant que le mot de code $\mathbf{c}$ a été émis ; nous prenons par exemple le mot $\mathbf{c} = (+\sqrt{2E_c}, +\sqrt{2E_c}, \ldots, +\sqrt{2E_c})$. Ce mot de code est transformé par l'évanouissement $\boldsymbol{\alpha}$ en $(\alpha_1 c_1, \ldots, \alpha_N c_N)$, noté $\boldsymbol{\alpha} \odot \mathbf{c}$. La transformation $\odot$ désigne par la suite le produit composante par composante de deux vecteurs. Si $\mathcal{A}$ est un ensemble de vecteurs, alors $\boldsymbol{\alpha} \odot \mathcal{A}$ représente l'ensemble des vecteurs de $\mathcal{A}$ multipliés composante par composante au vecteur $\boldsymbol{\alpha}$.

Nous cherchons, comme pour le canal gaussien, à évaluer une borne inférieure de la probabilité d'erreur par mot $P\mathrm{e}_{\mathrm{w}}$ du code sphérique $\mathcal{C}$. Pour cela, nous proposons de suivre les étapes ci-dessous :

1. Pour un évanouissement donné $\boldsymbol{\alpha}$, évaluer l'angle solide de la région de Voronoï du mot de code $\boldsymbol{\alpha} \odot \mathbf{c}$
2. Trouver l'angle $\theta_1(\boldsymbol{\alpha})$ équivalent correspondant à un cône circulaire de même angle solide centré sur le point $\boldsymbol{\alpha} \odot \mathbf{c}$
3. Évaluer la probabilité d'erreur conditionnelle $P\mathrm{e}_{\mathrm{w}|\boldsymbol{\alpha}} = P\mathrm{e}_{\mathrm{w}}(\theta_1)$ à l'aide de la fonction Q
4. Répéter les étapes 1, 2, 3, pour différents évanouissements de Rayleigh afin de calculer par simulation Monte-Carlo la probabilité d'erreur $P\mathrm{e}_{\mathrm{w}}$ donnée par :

$$P\mathrm{e}_{\mathrm{w}} = \int_{\boldsymbol{\alpha}} P\mathrm{e}_{\mathrm{w}|\boldsymbol{\alpha}}(\boldsymbol{\alpha}) p_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) d\boldsymbol{\alpha} \ .$$

Région de Voronoï de $\boldsymbol{\alpha} \odot \mathbf{c}$

La région de Voronoï du mot de code $\mathbf{c}$ après évanouissement est notée $V(\boldsymbol{\alpha} \odot \mathbf{c})$. Pour une modulation BPSK, les équivalences suivantes peuvent être écrites en notant $\mathbf{y}$ le point reçu :

$$\begin{aligned}
\forall\, \mathbf{y} \in \mathbb{R}^N,\ \mathbf{y} \in V(\boldsymbol{\alpha} \odot \mathbf{c}) \quad &\Leftrightarrow \quad \forall\, \mathbf{c}' \in \mathcal{C}\ \|\mathbf{y} - \boldsymbol{\alpha} \odot \mathbf{c}\|^2 \le \|\mathbf{y} - \boldsymbol{\alpha} \odot \mathbf{c}'\|^2 \\
&\Leftrightarrow \quad \forall\, \mathbf{c}' \in \mathcal{C}\ \sum_{i=1}^{N} (y_i - \alpha_i c_i)^2 \le \sum_{i=1}^{N} (y_i - \alpha_i c'_i)^2 \\
&\Leftrightarrow \quad \forall\, \mathbf{c}' \in \mathcal{C}\ \sum_{i=1}^{N} \alpha_i y_i c_i \ge \sum_{i=1}^{N} \alpha_i y_i c'_i \\
&\Leftrightarrow \quad \forall\, \mathbf{c}' \in \mathcal{C},\ \forall\, \mathbf{z} \in \mathbb{R}^N \text{tel que } \mathbf{z} = \boldsymbol{\alpha} \odot \mathbf{y},\ \|\mathbf{z} - \mathbf{c}\|^2 \le \|\mathbf{z} - \mathbf{c}'\|^2 \\
&\Leftrightarrow \quad \forall\, \mathbf{z} \in \mathbb{R}^N \text{tel que } \mathbf{z} = \boldsymbol{\alpha} \odot \mathbf{y},\ \mathbf{z} \in V(\mathbf{c}) \\
&\Leftrightarrow \quad \boldsymbol{\alpha} \odot \mathbf{y} \in V(\mathbf{c}) \ . \qquad (1.30)
\end{aligned}$$

La région de Voronoï du mot $\boldsymbol{\alpha} \odot \mathbf{c}$ est donc :

$$V(\boldsymbol{\alpha} \odot \mathbf{c}) = \boldsymbol{\alpha}^{-1} \odot V(\mathbf{c}) \ , \qquad (1.31)$$

où $\boldsymbol{\alpha}^{-1}$ désigne le vecteur $\left(\frac{1}{\alpha_1}, \frac{1}{\alpha_2}, \ldots, \frac{1}{\alpha_N}\right)$.

Angle solide de la région $V(\boldsymbol{\alpha} \odot \mathbf{c})$

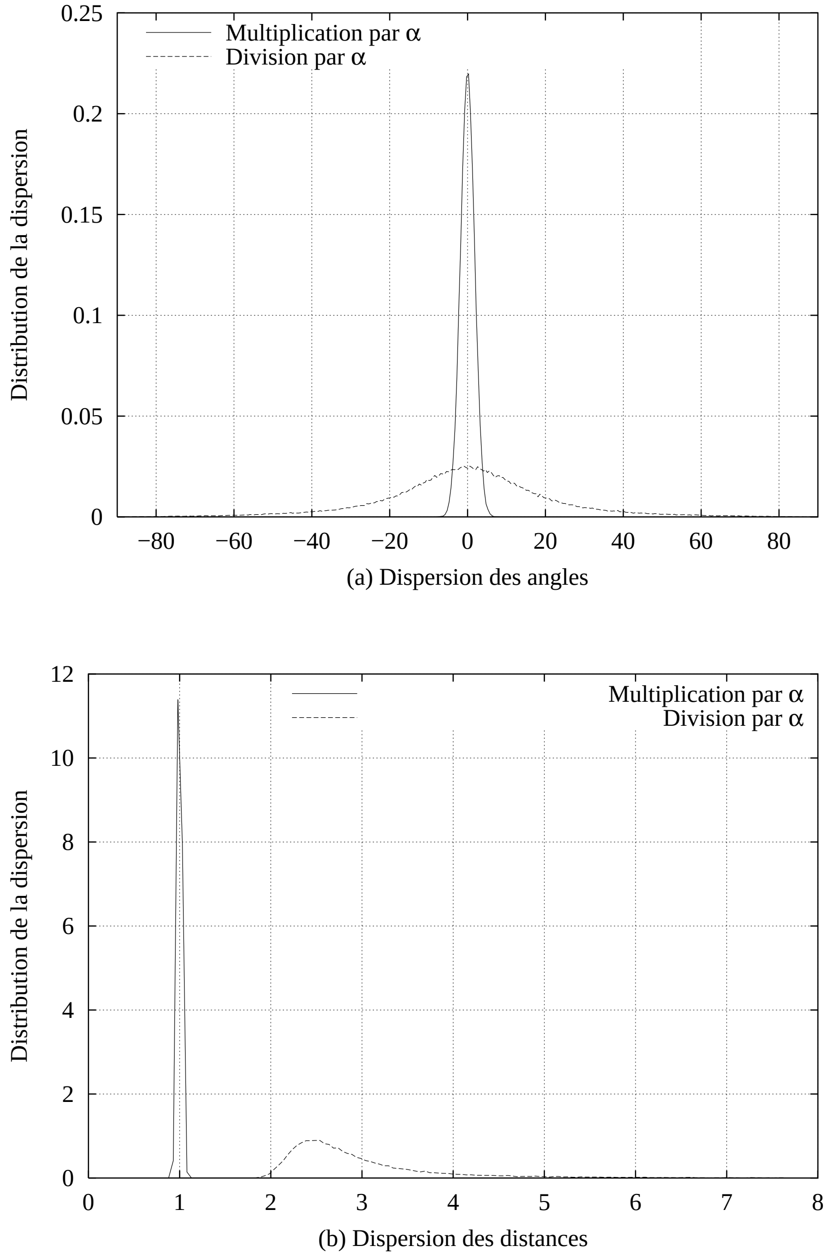
Pour évaluer l'angle solide $\Omega(\theta)$ de la région de Voronoï $V(\mathbf{c})$ dans le cas du canal gaussien, Shannon a utilisé le cône $\Lambda(\mathbf{c})$ de même angle solide $\Omega(\theta)$. Ce cône est, comme nous l'avons vu au paragraphe 1.3.1, délimité par la sphère $\mathrm{Sph}(\mathbf{a}, N-1)$ située à l'intersection de l'hyperplan $\mathcal{H}$ et de la sphère $\mathrm{Sph}(O, N)$. La sphère $\mathrm{Sph}(\mathbf{a}, N-1)$ prend la forme de la région de Voronoï dans l'hyperplan $\mathcal{H}$ ce qui permet, en calculant l'angle solide correspondant, d'obtenir une borne très précise de la probabilité d'erreur.

Un tel raisonnement ne peut cependant être appliqué au canal de Rayleigh. Il n'est pas possible d'approcher la région de Voronoï $V(\boldsymbol{\alpha} \odot \mathbf{c})$ dans l'hyperplan $\boldsymbol{\alpha}^{-1} \odot \mathcal{H}$ par l'ellipsoïde $\boldsymbol{\alpha}^{-1} \odot \mathrm{Sph}(\mathbf{a}, N-1)$ pour déterminer de manière précise la probabilité d'erreur. Les principales raisons à cela sont les suivantes :

- L'ellipsoïde $\boldsymbol{\alpha}^{-1} \odot \mathrm{Sph}(\mathbf{a}, N-1)$ est centré sur le point $\boldsymbol{\alpha}^{-1} \odot \mathbf{a}$, qui est radialement aligné avec le point $\boldsymbol{\alpha}^{-1} \odot \mathbf{c}$. Ce point est très éloigné du point évanoui $\boldsymbol{\alpha} \odot \mathbf{c}$. Or, le mot que nous cherchons après évanouissement est le mot $\boldsymbol{\alpha} \odot \mathbf{c}$, car la distance à minimiser selon le critère ML est $\|\mathbf{y} - \boldsymbol{\alpha} \odot \mathbf{c}\|^2$.
- Si nous considérons l'autre distance donnée par le système d'équivalence (1.30), *i.e.*, $\|\boldsymbol{\alpha} \odot \mathbf{y} - \mathbf{c}\|^2$, le point "reçu" utilisé est alors $\boldsymbol{\alpha} \odot \mathbf{y}$. Cette variable n'a plus la même loi de probabilité que $\mathbf{y}$, les composantes de ce point sont en effet des variables aléatoires gaussiennes de variances distinctes. Or, le raisonnement de Shannon fait l'hypothèse que le bruit est symétrique sur chaque composante. Nous ne pouvons pas utiliser cette distance pour évaluer la probabilité d'erreur.
- L'ellipsoïde $\boldsymbol{\alpha}^{-1} \odot \mathrm{Sph}(\mathbf{a}, N-1)$ n'épouse pas bien la forme de la région de Voronoï $\boldsymbol{\alpha}^{-1} \odot V(\mathbf{c})$: la division par $\boldsymbol{\alpha}$ ne conserve ni les distances ni les angles entre deux points (si α est une variable de Rayleigh, $E\left[\frac{1}{\alpha^2}\right]$ diverge). Soient deux mots de code $\mathbf{c_1}$ et $\mathbf{c_2}$, séparés avant évanouissement par l'angle $\theta_i = 90°$ (le demi-angle au sommet du cône est $45°$; ceci correspond à un code de rendement $R = \frac{1}{2}$ d'après la figure 1.7). La figure 1.11(a) représente la distribution de la dispersion $\theta_i - \theta_f$, si θ_f est l'angle entre les mots de code après division par $\boldsymbol{\alpha}$. De même, la figure 1.11(b) donne la distribution du rapport des distances entre $\mathbf{c_1}$ et $\mathbf{c_2}$ avant et après division par $\boldsymbol{\alpha}$. La longueur du code représenté est $N = 1000$.

Au lieu de considérer l'ellipsoïde $\boldsymbol{\alpha}^{-1} \odot \mathrm{Sph}(\mathbf{a}, N-1)$, nous proposons de considérer l'ellipsoïde $\mathcal{E} = \boldsymbol{\alpha} \odot \mathrm{Sph}(\mathbf{a}, N-1)$ comme approximation de la région de Voronoï $\boldsymbol{\alpha}^{-1} \odot V(\mathbf{c})$ dans l'hyperplan $\boldsymbol{\alpha} \odot \mathcal{H}$. Cet ellipsoïde est centré sur $\boldsymbol{\alpha} \odot \mathbf{a}$, qui se trouve sur le même rayon que $\boldsymbol{\alpha} \odot \mathbf{c}$. L'angle solide calculé à partir de cet ellipsoïde est ainsi centré sur le mot $\boldsymbol{\alpha} \odot \mathbf{c}$.

Soient $\mathbf{c_1}$ un mot de code voisin du mot de code $\mathbf{c}$, et $\mathcal{F}_1$ la facette de $V(\mathbf{c})$ séparant $\mathbf{c}$ de $\mathbf{c_1}$. La facette $\mathcal{F}_1$ est un hyperplan orthogonal à la droite $(\mathbf{c}\ \mathbf{c_1})$. Le milieu du segment $[\mathbf{c}\ \mathbf{c_1}]$ est appelé M_1 et appartient à la facette $\mathcal{F}_1$. La droite $(\mathbf{c}M_1)$ est orthogonale à cette facette. Le



(a) Dispersion des angles

(b) Dispersion des distances

FIG. 1.11 – Dispersion des angles et des distances après multiplication ou division par $\boldsymbol{\alpha}$

point M_1 est donc la projection orthogonale du point $\mathbf{c}$ sur la facette $\mathcal{F}_1$; il représente à ce titre un point critique pour la probabilité d'erreur en présence de bruit blanc additif gaussien. Après évanouissement, la nouvelle facette entre $\boldsymbol{\alpha} \odot \mathbf{c}$ et $\boldsymbol{\alpha} \odot \mathbf{c_1}$ est, d'après l'égalité (1.31), égale à l'hyperplan $\boldsymbol{\alpha}^{-1} \odot \mathcal{F}_1$. Or, le milieu du segment $[(\boldsymbol{\alpha} \odot \mathbf{c})\ (\boldsymbol{\alpha} \odot \mathbf{c_1})]$ est le point $\boldsymbol{\alpha} \odot M_1$ et la droite $((\boldsymbol{\alpha} \odot \mathbf{c})\ (\boldsymbol{\alpha} \odot M_1))$ est orthogonale à la facette $\boldsymbol{\alpha}^{-1} \odot \mathcal{F}_1$. Par conséquent, le point $\boldsymbol{\alpha} \odot M_1$ appartient à la facette $\boldsymbol{\alpha}^{-1} \odot \mathcal{F}_1$ et est la projection orthogonale de $\boldsymbol{\alpha} \odot \mathbf{c}$ sur cette facette.

Considérons maintenant un second mot de code $\mathbf{c_2}$ voisin du mot de code $\mathbf{c}$, tel que les facettes relatives aux mots $\mathbf{c_1}$ et $\mathbf{c_2}$ forment un coin de la région de Voronoï, comme le montre la figure 1.12. Nous définissons comme pour le mot de code $\mathbf{c_1}$, le point milieu M_2 et la facette $\mathcal{F}_2$ tels que $M_2 \in \mathcal{F}_2$. L'intersection des facettes $\mathcal{F}_1$ et $\mathcal{F}_2$ forme le coin de $V(\mathbf{c})$ situé entre $\mathbf{c_1}$ et $\mathbf{c_2}$; il s'agit par définition d'un espace de dimension $N-2$. L'intersection de cet espace avec le plan $\mathcal{P}$ contenant les vecteurs $\mathbf{c}$, $\mathbf{c_1}$ et $\mathbf{c_2}$ est un point noté A. Le point A est en fait l'intersection de la médiatrice de $[\mathbf{c}\ \mathbf{c_1}]$ qui passe par M_1 et de la médiatrice de $[\mathbf{c}\ \mathbf{c_2}]$ qui passe par M_2. Nous appellons par la suite ce point A "pointe" de la région $V(\mathbf{c})$.

La projection de la sphère $\mathrm{Sph}(\mathbf{a}, N-1)$ sur le plan $\mathcal{P}$ donne une sphère passant entre le point A et les deux points M_1 et M_2, par définition du cône $\Lambda(\mathbf{c})$. C'est pour cela que nous disons que $\mathrm{Sph}(\mathbf{a}, N-1)$ prend la forme de la région $V(\mathbf{c})$. Ceci est d'autant plus vrai que le nombre de facettes de la région de Voronoï $V(\mathbf{c})$ est grand lorsque la dimension de l'espace N est grand ; en effet la forme hexagonale de la région $V(\mathbf{c})$ ressemble alors à une forme circulaire. La probabilité d'erreur évaluée à partir de l'angle solide de $\mathrm{Sph}(\mathbf{a}, N-1)$ donne alors une valeur très proche de la véritable probabilité d'erreur.

Après évanouissement :

$$\begin{aligned}
\mathbf{c} &\quad \text{devient} \quad \boldsymbol{\alpha} \odot \mathbf{c}, \\
\mathbf{c_1} &\quad \text{devient} \quad \boldsymbol{\alpha} \odot \mathbf{c_1}, \\
\mathbf{c_2} &\quad \text{devient} \quad \boldsymbol{\alpha} \odot \mathbf{c_2}, \\
M_1 &\quad \text{devient} \quad \boldsymbol{\alpha} \odot M_1, \\
M_2 &\quad \text{devient} \quad \boldsymbol{\alpha} \odot M_2, \\
\mathcal{F}_1 &\quad \text{devient} \quad \boldsymbol{\alpha} \odot \mathcal{F}_1, \\
\mathcal{F}_2 &\quad \text{devient} \quad \boldsymbol{\alpha} \odot \mathcal{F}_2,
\end{aligned}$$

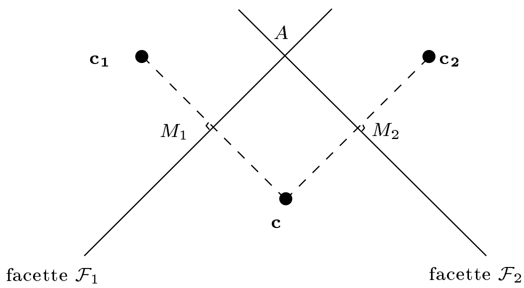


FIG. 1.12 – Disposition des mots de code $\mathbf{c}$, $\mathbf{c_1}$ et $\mathbf{c_2}$

Comme nous l'avons vu précédemment, $\boldsymbol{\alpha} \odot M_1$ et $\boldsymbol{\alpha} \odot M_2$ se trouvent respectivement sur les facettes $\boldsymbol{\alpha} \odot \mathcal{F}_1$ et $\boldsymbol{\alpha} \odot \mathcal{F}_2$ de la région de Voronoï de $\boldsymbol{\alpha} \odot \mathbf{c}$. Par ailleurs les figures 1.11(a) et 1.11(b) montrent que les angles et les distances ne varient que très peu lors de la multiplication par $\boldsymbol{\alpha}$. Le point $\boldsymbol{\alpha} \odot A$ est par conséquent positionné à l'intersection des facettes $\boldsymbol{\alpha} \odot \mathcal{F}_1$, $\boldsymbol{\alpha} \odot \mathcal{F}_2$ et du plan $\boldsymbol{\alpha} \odot \mathcal{P}$ contenant les vecteurs $\boldsymbol{\alpha} \odot \mathbf{c}$, $\boldsymbol{\alpha} \odot \mathbf{c_1}$ et $\boldsymbol{\alpha} \odot \mathbf{c_2}$. La distance du point $\boldsymbol{\alpha} \odot A$ à l'ellipsoïde $\mathcal{E}$ reste ainsi quasi-constante (*i.e.*, similaire à la distance entre A et $\mathrm{Sph}(\mathbf{a}, N-1)$). Il en est de même pour la distance des points $\boldsymbol{\alpha} \odot M_1$ et $\boldsymbol{\alpha} \odot M_2$ à l'ellipsoïde $\mathcal{E}$. La projection de $\mathcal{E}$ sur le plan $\boldsymbol{\alpha} \odot \mathcal{P}$ est centrée sur le point $\boldsymbol{\alpha} \odot \mathbf{c}$ et passe comme pour le canal gaussien, entre la pointe $\boldsymbol{\alpha} \odot A$ et les deux milieux $\boldsymbol{\alpha} \odot M_1$ et $\boldsymbol{\alpha} \odot M_2$. Rappelons que ces deux derniers points sont des points critiques pour la probabilité d'erreur en tant que projection orthogonale du mot $\boldsymbol{\alpha} \odot \mathbf{c}$ sur les facettes de la région de Voronoï. L'angle solide calculé à partir de $\mathcal{E}$ représente donc une approximation correcte de l'angle solide de la région de Voronoï $V(\boldsymbol{\alpha} \odot \mathbf{c})$.

Calcul de l'angle θ_1

L'ellipsoïde $\mathcal{E}$ est défini comme l'intersection de l'hyperplan $\boldsymbol{\alpha} \odot \mathcal{H}$ et de l'ellipsoïde de dimension N centré à l'origine, $\boldsymbol{\alpha} \odot \mathrm{Sph}(O, N)$. Tous les points $\mathbf{x}$ appartenant à l'ellipsoïde $\mathcal{E}$ vérifient donc les deux équations suivantes :

$$\boldsymbol{\alpha} \odot \mathrm{Sph}(O, N) : \qquad \sum_{i=1}^{N} \frac{x_i^2}{\alpha_i^2} = E \tag{1.32}$$

et

$$\boldsymbol{\alpha} \odot \mathcal{H} : \qquad \sum_{i=1}^{N} \frac{a_i x_i}{\alpha_i} = \|\mathbf{a}\|^2 \ . \tag{1.33}$$

Le point $\boldsymbol{\alpha} \odot \mathbf{a}$ est le centre de l'ellipsoïde $\mathcal{E}$ et appartient à la droite $(O\ \boldsymbol{\alpha} \odot \mathbf{c})$, comme le montre la figure 1.13.

L'angle solide $\Omega(\theta_1)$ est égal à la surface de l'ellipsoïde $\mathcal{E}$, qui est en fait un volume de dimension $N-1$, projeté sur la sphère unité de dimension N. Une bonne approximation de la surface $\Omega(\theta_1)$ est :

$$\Omega(\theta_1) \sim \mathrm{Vol}(1, N-1) \prod_{j=1}^{N-1} d_j = \frac{\pi^{\frac{N-1}{2}}}{\Gamma\left(\frac{N+1}{2}\right)} \prod_{j=1}^{N-1} d_j, \tag{1.34}$$

où les d_j sont les projections des $N-1$ axes de l'ellipsoïde $\mathcal{E}$ sur la sphère unité. Chaque axe de l'ellipsoïde est défini par deux points opposés par rapport au centre $\boldsymbol{\alpha} \odot \mathbf{a}$. La distance $2d_j$ est la longueur de l'arc obtenu après projection des deux points opposés sur la sphère.

Pour trouver les $2N-2$ points $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ définissant les $N-1$ axes de l'ellipsoïde $\mathcal{E}$, nous devons optimiser la quantité $\|\mathbf{x} - \boldsymbol{\alpha} \odot \mathbf{a}\|^2$ en prenant en compte les deux contraintes données par les équations (1.32) et (1.33). En introduisant les multiplicateurs de Lagrange λ et γ associés respectivement aux contraintes (1.32) et (1.33), la nouvelle fonction à optimiser devient :

$$f(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\alpha} \odot \mathbf{a}\|^2 - \lambda \left(\sum_{i=1}^{N} \frac{x_i^2}{\alpha_i^2} - E \right) - \gamma \left(\sum_{i=1}^{N} \frac{a_i x_i}{\alpha_i} - \|\mathbf{a}\|^2 \right), \tag{1.35}$$

d'où, après dérivation de $f(\mathbf{x})$ par rapport à tous les $x_i,\ i = 1, \ldots, N$,

$$x_i = \frac{\frac{\gamma}{2} \alpha_i a_i + \alpha_i^3 a_i}{\alpha_i^2 - \lambda} \ . \tag{1.36}$$

car $\lambda \neq \alpha_i^2$. En effet, si $\lambda = \alpha_i^2$, alors la condition (1.33) mène à $\gamma = -2\alpha_i^2 = -2\lambda$. Dans ce cas, la contrainte (1.32) conduit à $x_i = \alpha_i a_i$, c'est-à-dire $\mathbf{x} = \boldsymbol{\alpha} \odot \mathbf{a}$, le centre de l'ellipsoïde.

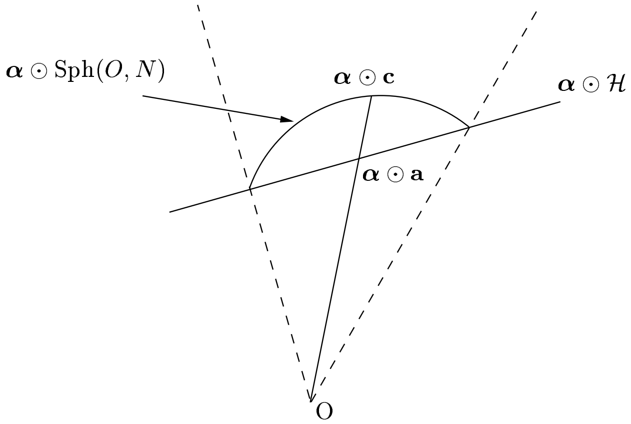


FIG. 1.13 – Après évanouissements

En remplaçant x_i par l'équation (1.36) dans la condition (1.33), nous obtenons :

$$\left(\frac{\gamma}{2} + \lambda\right) \sum_{i=1}^{N} \frac{a_i^2}{\alpha_i^2 - \lambda} = 0,$$

soit

$$\sum_{i=1}^{N} \frac{a_i^2}{\alpha_i^2 - \lambda} = 0 \ . \tag{1.37}$$

Or $a_i = c_i \cos\theta_0$ donc $a_i^2 = (\cos\theta_0)^2$ et par conséquent l'équation (1.37) devient :

$$\sum_{i=1}^{N} \frac{1}{\alpha_i^2 - \lambda} = 0 \ . \tag{1.38}$$

Cette équation définit un polynôme de degré $N-1$ dont les $N-1$ racines λ_j correspondent aux $N-1$ axes de l'ellipsoïde. Pour résoudre cette équation, nous classons les valeurs α_j, $j = 1, \ldots, N$, par ordre croissant. Nous remarquons alors qu'entre deux valeurs consécutives α_j et α_{j+1} se trouve une racine λ_j de l'équation (1.38). Nous évaluons cette racine par dichotomie sur l'intervalle $]\alpha_j, \alpha_{j+1}[$.

Pour une racine λ ainsi déterminée, l'équation (1.32) mène à :

$$\sum_{i=1}^{N} \frac{a_i^2 \left(\frac{\gamma}{2} + \alpha_i^2\right)^2}{(\alpha_i^2 - \lambda)^2} = E \ . \tag{1.39}$$

Posons

$$\begin{aligned}
V_1 &= \sum_{i=1}^{N} \frac{1}{(\alpha_i^2 - \lambda)^2}, \\
V_2 &= \sum_{i=1}^{N} \frac{\alpha_i^2}{(\alpha_i^2 - \lambda)^2}, \\
V_4 &= \sum_{i=1}^{N} \frac{\alpha_i^4}{(\alpha_i^2 - \lambda)^2} \ .
\end{aligned}$$

L'équation (1.39) devient :

$$V_1\gamma^2 + 4V_2\gamma + 4\left(V_4 - \frac{1}{\cos^2\theta_0}\right) = 0 \ . \tag{1.40}$$

Il s'agit d'une équation du second degré en γ, dont le déterminant est positif. Pour chaque valeur de λ il existe donc deux valeurs γ_A et γ_B de γ associées aux deux points opposés autour du point $\boldsymbol{\alpha} \odot \mathbf{a}$. Chaque triplet $(\lambda, \gamma_A, \gamma_B)$ définit un axe de l'ellipsoïde. Après résolution de l'équation (1.40), nous obtenons :

$$\gamma_{A,B} = \frac{-2V_2 \pm 2\sqrt{V_2^2 - V_1(V_4 - 1/\cos^2\theta_0)}}{V_1} \ . \tag{1.41}$$

Les distances d_j nécessaires pour évaluer l'angle solide $\Omega(\theta_1)$ sont alors définies par :

$$2d_j = \psi_A(\lambda_j) + \psi_B(\lambda_j) \qquad j = 1, \ldots, N-1,$$

où les deux angles ψ_A et ψ_B appartiennent à $[0, \frac{\pi}{2}]$, et sont exprimés en radians. Ces deux angles sont égaux à la longueur de l'arc se trouvant sur la sphère unité et liant la projection du centre de l'ellipsoïde $\boldsymbol{\alpha} \odot \mathbf{a}$ à la projection du point $\mathbf{x}$ déterminé par l'équation (1.36). Si

$$< \mathbf{x}, \mathbf{y} >= \sum_{i=1}^{N} x_i y_i$$

désigne le produit scalaire dans l'espace euclidien $\mathrm{I\!R}^N$, alors les angles ψ_A et ψ_B sont définis par les équations suivantes :

$$\cos(\psi_A) =< \frac{\mathbf{x}(\lambda_j, \gamma_{A,j})}{\|\mathbf{x}(\lambda_j, \gamma_{A,j})\|}, \frac{\boldsymbol{\alpha} \odot \mathbf{a}}{\|\boldsymbol{\alpha} \odot \mathbf{a}\|} > \qquad \cos(\psi_B) =< \frac{\mathbf{x}(\lambda_j, \gamma_{B,j})}{\|\mathbf{x}(\lambda_j, \gamma_{B,j})\|}, \frac{\boldsymbol{\alpha} \odot \mathbf{a}}{\|\boldsymbol{\alpha} \odot \mathbf{a}\|} > \ .$$

A partir des distances d_j que nous venons de calculer, nous pouvons maintenant déduire l'angle solide $\Omega(\theta_1)$, en utilisant l'équation (1.34). Puis, en résolvant l'équation (1.26), nous obtenons le demi-angle au sommet θ_1 d'un cône de même angle solide. Enfin, de l'équation (1.23), nous déduisons la probabilité d'erreur conditionnelle $Q(\theta_1)$. En moyennant ensuite sur les vecteurs $\boldsymbol{\alpha}$, nous obtenons une approximation de la borne inférieure de la probabilité d'erreur par mot du code sphérique sur le canal de Rayleigh.

1.3.5 Résultats numériques pour le canal de Rayleigh

La figure 1.14 présente l'évolution du rapport signal-à-bruit nécessaire pour atteindre une probabilité d'erreur par mot de 10^{-5} en fonction de la dimension du code binaire. Cette évolution est représentée pour les rendements $R = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$. Les limites lorsque la longueur du code tend vers l'infini sont évaluées à partir de l'expression de la capacité donnée par l'équation (1.11) et sont mentionnées sur la figure en guise de comparaison.

La figure 1.15 illustre la probabilité d'erreur par mot pour un code binaire de rendement $\frac{1}{3}$ sur canal AWGN et canal de Rayleigh. Différentes dimensions du code sont représentées, $K =$

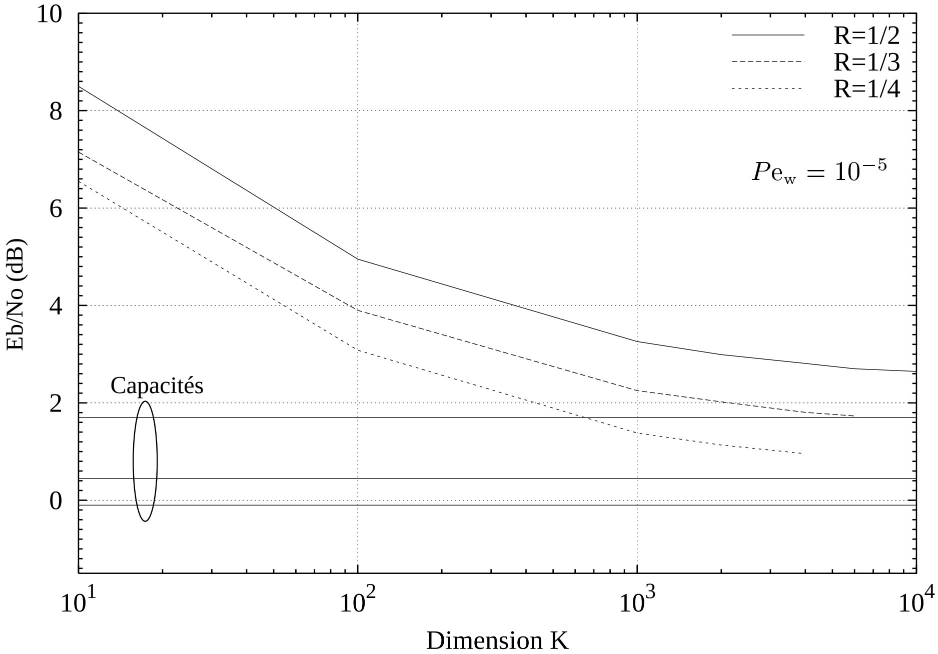


FIG. 1.14 – Évolution du rapport signal-à-bruit en fonction de la dimension du code pour le canal de Rayleigh

$100, 200, 400, 800$. Lorsque la dimension K du code augmente, les performances se rapprochent de la capacité du canal de Rayleigh qui se trouve à 0.45 dB pour un code de rendement $\frac{1}{3}$, lorsque la modulation employée est une BPSK. Pour une probabilité d'erreur par mot $Pe_w = 10^{-5}$, l'écart entre les performances sur canal AWGN et sur canal de Rayleigh, à peu près identique pour les dimensions considérées, et de l'ordre de 1.90 dB.

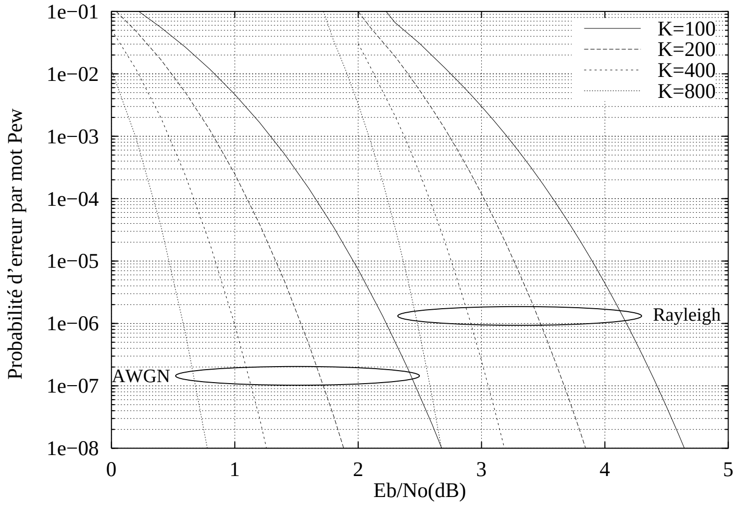


FIG. 1.15 – Performances sur canaux AWGN et de Rayleigh pour $R = \frac{1}{3}$, $K = 100, 200, 400, 800$

Les figures 1.16 et 1.17 comparent les probabilités d'erreur par mot obtenues sur canal de Rayleigh et canal AWGN pour des dimensions $K = 100, 200, 400, 800$ lorsque le rendement du code est respectivement $R = \frac{1}{2}$ et $R = \frac{1}{4}$.

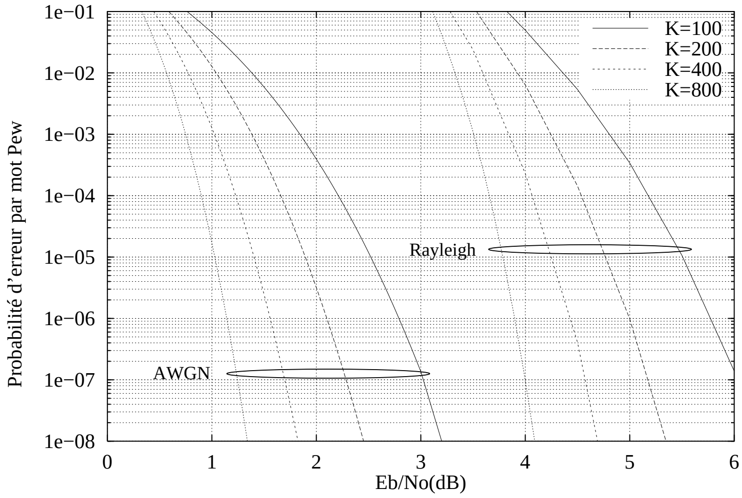


Fig. 1.16 – Performances sur canaux AWGN et de Rayleigh pour $R = \frac{1}{2}$, $K = 100, 200, 400, 800$

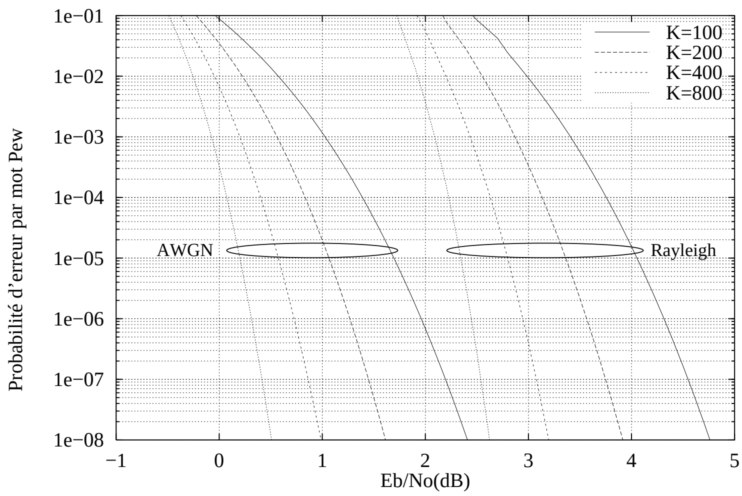


Fig. 1.17 – Performances sur canaux AWGN et de Rayleigh pour $R = \frac{1}{4}$, $K = 100, 200, 400, 800$

La figure 1.18 présente les performances d'un turbo-code de rendement $R = \frac{1}{2}$ construit à partir de deux codes convolutifs récursifs systématiques de rendement $\frac{1}{2}$ et de polynômes générateurs $(37, 21)$. La probabilité d'erreur par bit atteinte par ce turbo-code sur le canal de Rayleigh est comparée à la probabilité d'erreur par bit des codes optimaux. Cette dernière est obtenue en utilisant l'approximation (1.29). Lorsque la probabilité d'erreur est $Pe_b = 10^{-4}$, l'écart entre les deux courbes est de 0.5 dB pour $K = 1024$ et 1.4 dB pour $K = 128$; ces écarts sont à peu près identiques aux écarts relevés pour le canal gaussien (la même remarque peut être faite pour les probabilités d'erreur par mot). Quand le taux d'erreur binaire diminue, les performances des turbo-codes ont tendance à s'éloigner des performances des codes optimaux en raison de l'apparition du phénomène de palier ("*error-floor*") dû à leur faible distance minimale.

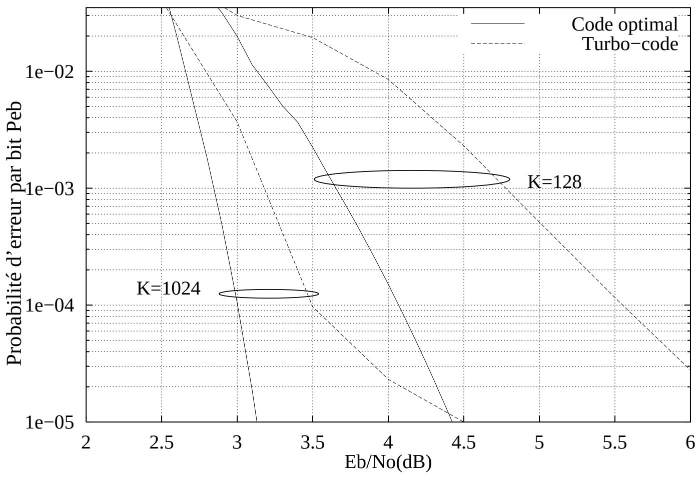


FIG. 1.18 – Comparaison des turbo-codes avec les codes optimaux sur canal de Rayleigh, $R = \frac{1}{2}$

1.4 Conclusions

Nous nous sommes intéressés dans ce chapitre aux limites des systèmes de codage lorsque le critère du maximum de vraisemblance est retenu pour le décodage. Ces limites ont été étudiées tout d'abord pour des codes de longueur infinie ; elles sont dans ce cas déterminées par la capacité du canal considéré. Les expressions littérales de la capacité ont été écrites en fonction du rapport signal-à-bruit $\frac{E_b}{N_0}$ du système pour les canaux AWGN et de Rayleigh. L'influence de la diversité sur la capacité du canal de Rayleigh a également été étudiée.

Dans un second temps, nous avons présenté l'approche géométrique proposée par Shannon pour évaluer les performances des codes optimaux de longueur finie et de rendement fixé, sur le canal AWGN. Cette approche permet de déterminer le rapport signal-à-bruit nécessaire pour atteindre une probabilité d'erreur par mot donnée, en fonction de la dimension K du code. Nous avons également utilisé cette étude pour analyser l'évolution de la probabilité d'erreur par mot en fonction du rapport signal-à-bruit pour une dimension K et un rendement R du code fixés. Afin de déduire de ces performances la probabilité d'erreur par bit optimale, une approximation de la distance minimale des codes optimaux a été introduite en se basant sur la borne de Gilbert-Varshamov pour les codes binaires aléatoires. Nous avons alors pu comparer ces taux d'erreurs binaires aux performances obtenues par simulation d'un turbo-code de rendement $R = \frac{1}{2}$. Il est apparu que l'écart entre les performances augmente lorsque la taille de l'entrelaceur du turbo-code diminue.

Enfin, nous avons étudié l'évolution de la probabilité d'erreur optimale en présence d'évanouissements de Rayleigh, lorsqu'une modulation BPSK est employée. Les performances du turbo-code considéré pour le canal gaussien ont été comparées à la nouvelle borne obtenue. L'écart avec cette borne est resté quasi-identique à celui observé sur le canal gaussien pour différentes dimensions du code.

Chapitre 2

Principes des codes composites

La recherche de bons codes longs imitant le codage aléatoire a toujours été un défi pour la communauté du codage. Il s'agit de trouver des codes permettant de se rapprocher des limites fixées par Shannon présentées au chapitre 1. Le codage et le décodage de ces codes doivent être simples, du moins réalisables en pratique. Shannon ne propose dans sa théorie aucune "recette" pour atteindre les performances optimales si ce n'est le codage aléatoire. Ce dernier est certes facile à mettre en œuvre, mais requiert un décodage exhaustif de complexité prohibitive lorsque la longueur du code tend vers l'infini.

La composition de codes est une méthode spécifique permettant de construire de longs codes à partir de codes plus simples. Elle inclut en particulier des techniques comme la concaténation [33], l'intersection [42][76], ou encore le produit [31] de codes. Ces codes suscitent un regain d'intérêt du fait de leurs excellentes performances, moyennant une faible complexité de décodage par rapport à un unique code de même longueur. En effet, leur structure favorise le décodage des différents codes constituants du code composite[1] de manière parallèle ou séquentielle. Par ailleurs, si l'on considère un décodage à entrée et à sortie souples pour chaque code composant, il est alors possible d'itérer simplement le processus de décodage, comme nous le verrons par la suite. Ces codes sont appelés *codes composites*.

Nous allons tout d'abord présenter les principales méthodes de combinaison de codes qui permettent de construire des codes composites. Puis nous nous intéressons à la représentation graphique de ces codes. Dans une troisième partie, nous décrivons leurs performances théoriques pour un décodage à maximum de vraisemblance. Un tel décodage étant irréalisable en pratique, nous expliquons les principes du décodage itératif en présentant les principaux algorithmes qui peuvent être utilisés. Nous insistons plus particulièrement sur deux algorithmes que nous avons utilisés tout au long de ce travail : l'algorithme aller-retour [2] et l'algorithme de Fossorier-Lin [37] pour le décodage souple des codes en blocs. Nous avons adapté ce dernier algorithme afin d'en extraire une sortie souple pour itérer le processus de décodage. Enfin, nous concluons sur les performances obtenues avec les différents codes présentés, qui sont comparées aux performances optimales évaluées dans le chapitre précédent.

[1] par abus de langage et dans un souci de simplification, nous utiliserons désormais la dénomination *code constituant* pour traduire le terme anglais *constituent code*

2.1 Différentes structures de codes composites

La combinaison de codes repose essentiellement sur trois méthodes : la concaténation, l'intersection et le produit de codes. Ces différentes techniques sont décrites ci-après et illustrées par quelques exemples couramment évoqués dans la littérature.

2.1.1 Concaténation de codes

Les codes concaténés furent introduits par Forney en 1966 [33]. Celui-ci proposa de construire de longs codes en blocs par concaténation série de codes beaucoup plus simples. Le schéma général initialement proposé par Forney est représenté sur la figure 2.1 : N_1 et N_2 sont les longueurs respectives du code extérieur C^1 et du code intérieur C^2, tandis que R_1 et R_2 sont leurs rendements respectifs. La longueur totale du code est $N = N_2$ et son rendement est $R = R_1\ R_2$. Le code intérieur prend pour bits d'information la sortie codée du code extérieur. Le décodage se déroule en deux phases : la sortie du canal est utilisée par le décodeur intérieur pour estimer un mot de code de C^2, puis cette estimation est transmise au décodeur extérieur pour décoder un mot de C^1.

Le code extérieur originalement proposé par Forney est un code de Reed Solomon, pour lequel il existe un schéma de décodage efficace fondé sur la structure algébrique du code, l'algorithme de Berlekamp-Massey [10]. Le code intérieur est un code BCH binaire, dont le décodage est amélioré à l'aide de l'algorithme GMD (*Generalized Minimum Distance*) [34], qui permet d'utiliser la sortie souple du canal. Par la suite, d'autres codes ont été considérés. Notons en particulier

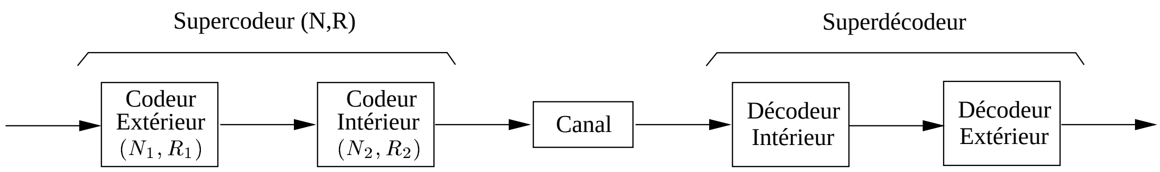


FIG. 2.1 – Structure générale d'un code concaténé de Forney.

l'introduction de codes convolutifs comme codes intérieurs dans des schémas de codage tels que celui utilisé pour la sonde Galiléo [18], ou encore comme codes intérieurs et extérieurs dans le schéma de codage des turbo-codes série [6][7]. L'un des principaux intérêts d'un schéma de codage concaténé construit à partir de deux codes convolutifs est la disponibilité d'un décodeur optimal maximisant la vraisemblance (algorithme de Viterbi) ou la probabilité *a posteriori* des symboles (algorithme aller-retour) en sortie du décodeur, ce qui permet d'une part de transmettre une information souple d'un décodeur à l'autre afin d'éviter toute perte d'information, mais également d'itérer le processus de décodage comme le décrivent Benedetto *et al.* [6][7].

Alors que la concaténation série consiste à coder le résultat d'un codage antérieur, il est également possible de considérer une concaténation parallèle, comme celle décrite notamment dans l'article original sur les turbo-codes parallèle[2] [11]. Le schéma de codage est alors celui

[2]nous appelons pour simplifier *turbo-code parallèle* (respectivement *turbo-code série*) un turbo-code construit à partir d'une concaténation parallèle (respectivement série) de codes élémentaires.

représenté sur la figure 2.2 pour deux codes constituants. Le schéma de décodage est toujours

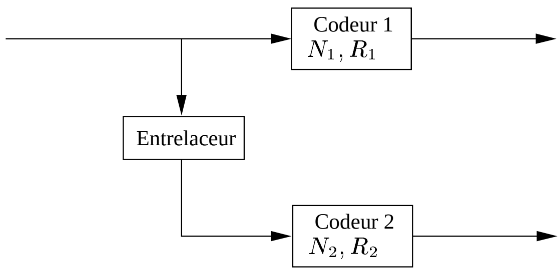


FIG. 2.2 – Structure générale d'un code concaténé parallèle.

fondé sur le même principe, *i.e.*, chaque code constituant est décodé seul, en prenant en compte le résultat du décodage du code constituant précédent ou du décodage global précédent. Alors que la structure initiale des turbo-codes parallèle utilise des codes constituants convolutifs, récursifs, systématiques et identiques, il est également possible de considérer comme codes constituants des codes en blocs [48], ou des codes constituants différents [75].

Le rendement global de la concaténation parallèle représentée sur la figure 2.2 ($J = 2$ codes constituants) est donné par :

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

Dans le cas des turbo-codes parallèle, les codes constituants étant systématiques, la sortie du codeur est perforée afin de ne garder qu'une unique version de la séquence d'information. Si l'on néglige la fermeture du treillis des codes convolutifs, le rendement du turbo-code est :

$$R = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2} \ ,$$

soit, si les deux codes sont identiques,

$$R = \frac{R_1}{2 - R_1} \ .$$

2.1.2 Intersection de codes

En 1963, Gallager a introduit une famille de codes correcteurs d'erreurs construits à partir de matrices de parité à faible densité, appelés *codes LDPC* (*Low Density Parity Check Codes*) [42]. Ces codes furent longtemps oubliés bien qu'exploitant des idées particulièrement pertinentes pour la construction de bons codes. Ainsi, Gallager utilise des permutations aléatoires reliant les codes de parité afin de construire un code efficace de faible complexité et imitant le codage aléatoire. Le décodage proposé est un décodage probabiliste, "ancêtre" en quelque sorte du décodage itératif. Depuis 1993 et l'introduction des turbo-codes, les codes LDPC sont largement redécouverts et exploités sous différentes formes [58][56][66][25].

Un code LDPC de paramètres (N, j, ℓ) est un code en blocs linéaire de longueur N tel que sa matrice de parité H a j "1" par colonne et ℓ "1" par ligne. Les nombres j et ℓ sont très petits

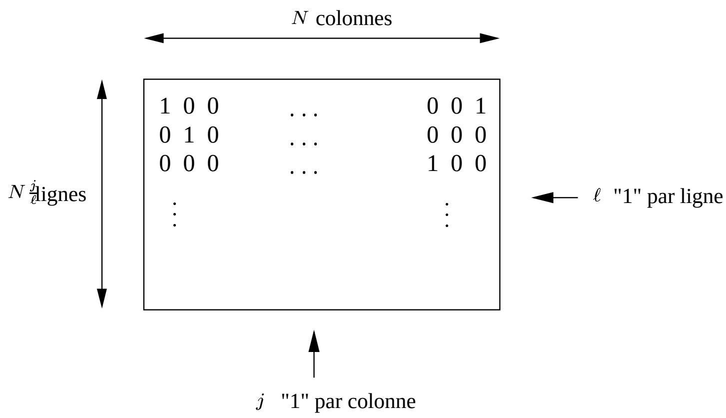


Fig. 2.3 – Matrice de Parité d'un code LDPC (N, j, ℓ).

par rapport à la longueur du code N afin d'obtenir une matrice de parité à faible densité. La figure 2.3 illustre la matrice de parité d'un code LDPC. Cette matrice H a par conséquent Nj/ℓ lignes, *i.e.*, le code est constitué de Nj/ℓ équations de parité (*parity check equations* notées pce). Chaque bit appartient à j pce et chaque pce implique ℓ bits. Le rendement du code vérifie donc :

$$R \geq 1 - \frac{j}{\ell} ,$$

l'inégalité provenant du fait que les lignes de la matrice H ne sont pas forcément indépendantes en raison des permutations aléatoires.

La matrice de parité H, telle que présentée par Gallager dans sa thèse [42], peut être divisée en j sous-matrices $H^1, \ldots, H^j$, chacune contenant un seul "1" par colonne. La première sous-matrice H^1 est une sorte de matrice identité dans laquelle chaque "1" serait remplacée par ℓ "1", et dont le nombre de colonnes serait par conséquent multiplié par un facteur ℓ. Les $j-1$ autres sous-matrices $H^2, \ldots, H^{j-1}$ s'obtiennent en appliquant $j-1$ permutations aléatoires $\pi_2, \ldots, \pi_j$ sur les colonnes de la sous-matrice H^1. Ainsi la matrice d'un code LDPC $(N = 20, j = 3, \ell = 4)$ particulier est donnée sur la figure 2.4. On remarque qu'en sommant l'ensemble des lignes de chaque sous-matrice on obtient une ligne ne contenant que des «1» ce qui signifie qu'il y a au moins $j-1 = 2$ lignes dépendantes dans cette matrice de parité et que le rendement est supérieur à $1 - \frac{3}{4} = 1/4$.

Ce code LDPC peut également être vu comme l'intersection de j super-codes $C^1, \ldots, C^j$ dont les matrices de parité respectives sont $H^1, \ldots, H^j$. Comme chaque sous-matrice est constituée de $\frac{N}{\ell}$ équations de parité mutuellement exclusives, chaque super-code est la somme directe de $\frac{N}{\ell}$ codes de parité notés $C_0(\ell, \ell-1)$. Nous avons par conséquent les deux relations suivantes :

$$\mathcal{C} = \bigcap_{i=1}^{j} C^i \qquad \text{et} \qquad C^i = \bigoplus_{m=1}^{\frac{N}{\ell}} \mathcal{C}_0 \ \ \forall\, i = 1, \ldots, j,$$

où le signe $\oplus$ désigne ici la somme directe de plusieurs codes. Notons qu'il peut être nécessaire de systématiser la matrice de parité du code LDPC pour calculer le rendement du code ou encore pour faciliter le codage, qui reste l'inconvénient majeur des codes LDPC malgré plusieurs tentatives de simplification [67]. Après systématisation de la matrice H, celle-ci n'est plus à faible

$$
H_{20,3,4} = \begin{array}{cc}
\left[\begin{array}{c}
1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1 \\
\hdashline
1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\
0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0 \\
0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0 \\
0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1 \\
\hdashline
1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0 \\
0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\
0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0 \\
0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0 \\
0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1
\end{array}\right]
&
\begin{array}{c}
H^1 \\ \pi_2 \\ H^2 \\ \pi_3 \\ H^3
\end{array}
\end{array}
$$

FIG. 2.4 – Matrice de Parité d'un code LDPC $(20, 3, 4)$.

densité. Or, le décodage probabiliste proposé par Gallager utilise cette propriété particulière de la matrice de parité ; il est donc nécessaire de considérer la matrice systématique pour le codage et la matrice de parité à faible densité pour le décodage. Ces matrices sont identiques à une permutation près.

La construction de codes à faible densité introduite par Gallager peut être généralisée en remplaçant les codes de parité par des codes en blocs linéaires plus complexes. Ainsi en 1981, Tanner [76] a introduit des codes construits à partir de graphes bipartites utilisant un entrelaceur déterministe. Plus tard, Boutros *et al.* [14], et Lentmaïer *et al.* [55] ont proposé indépendamment des codes à faible densité construits à partir de codes de Hamming et de permutations aléatoires. Ces codes sont appelés codes à faible densité généralisés construits à partir de codes en blocs, notés GLD blocs (*Generalized Low Density Codes*). Ce sont des codes de Tanner fondés sur des graphes bipartites aléatoires.

La matrice de parité d'un code GLD de longueur N construit à partir d'un code en blocs $C_0(n_0, k_0)$ est obtenue en remplaçant chaque ligne de la matrice de parité du code LDPC par $n_0 - k_0$ lignes incluant une copie de la matrice de parité H_0 du code linéaire en blocs C_0. Cette opération est représentée sur la figure 2.5, lorsque la matrice de parité est constituée de $J = 2$ sous-matrices[3]. Par définition, les codes C_0 sont disjoints. La première sous-matrice H^1 est donc la somme directe de $\frac{N}{n_0}$ codes identiques C_0. La matrice du code GLD est formée de J sous-matrices qui sont des versions entrelacées de la sous-matrice H^1. Par conséquent, comme pour les LDPC, nous avons les relations suivantes :

$$
\mathcal{C} = \bigcap_{i=1}^{J} C^i \qquad \text{avec} \quad C^i = \pi_i(C^\star) \quad \forall\, i = 1, \ldots, J \quad \text{et} \quad C^\star = \bigoplus_{m=1}^{\frac{N}{n_0}} C_0(n_0, k_0) \ .
$$

Si C_0 est de rendement r_0, et si les permutations $\pi_j,\ j = 1, \ldots, J$ sont aléatoires (π_1 n'est autre que l'identité, $C^1 = \pi_1(C^1)$) alors le rendement du code GLD correspondant vérifie l'inégalité :

$$
R \geq 1 - J(1 - r_0) \ .
$$

[3] J ne désigne plus maintenant le nombre de "1" par colonne, mais le nombre de sous-matrices. Il peut y avoir plus d'un "1" par colonne dans chaque sous-matrice.

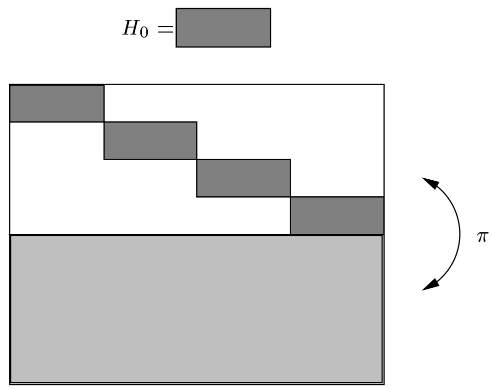


FIG. 2.5 – Matrice de Parité d'un code GLD avec $J = 2$ sous-matrices.

2.1.3 Codes Produits

L'idée de combiner des codes simples pour obtenir un code plus complexe n'est pas récente : 1966 pour les codes concaténés, 1963 pour les intersections de codes. Une autre solution pour construire un code composite est de considérer des codes produits, introduits en 1954 par Elias [31]. Le principe d'un code produit est d'écrire les mots d'un code selon les lignes d'une matrice puis de recoder ces mots à l'aide d'un second code appliqué sur les colonnes. En généralisant ce processus à un tableau à J dimensions et en faisant tendre J vers l'infini, Elias a montré que la probabilité d'erreur d'un tel code tend vers 0.

Si l'on considère J codes linéaires binaires $C^1(N_1, K_1, d_1)$, $C^2(N_2, K_2, d_2)$, $\ldots$, $C^J(N_J, K_J, d_J)$, le code produit $\mathcal{C} = C^1 \times C^2 \times \ldots \times C^J$ représente toutes les matrices pour lesquelles les différentes dimensions sont des mots de code des C^i, $i = 1, \ldots , J$. Dans le cas particulier où $J = 2$ et où les deux codes sont systématiques, une représentation du code produit est donnée sur la figure 2.6. Le code produit $\mathcal{C}$ a pour longueur $N = \prod_{i=1}^{J} N_i$ et pour rendement $R = \prod_{i=1}^{J} R_i$.

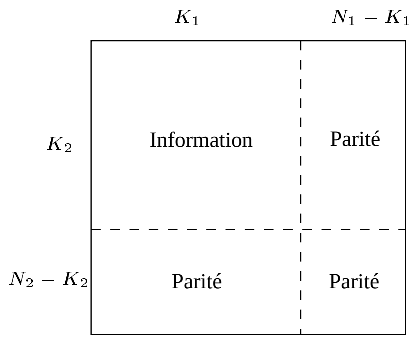


FIG. 2.6 – Structure d'un mot de code systématique du code produit pour $J = 2$.

De même la distance minimale du code produit $\mathcal{C}$ est égale au produit des distances minimales de ses codes constituants, *i.e.*, $d = \prod_{i=1}^{J} d_i$. Notons que, du fait de la linéarité du code, la parité qui se trouve dans la partie en bas à droite du code produit sur la figure 2.6, est à la fois parité pour le code C^1 et parité pour le code C^2.

Les codes produits initialement proposés par Elias ont été construits à partir de codes en blocs et plus particulièrement d'équations de parité. Pour augmenter la longueur du code produit tout en gardant son rendement constant, il faut augmenter soit la longueur des codes constituants, soit le nombre de dimensions du code produit, en contrôlant le rendement de chaque code constituant. Dans le premier cas, la complexité du décodage de chaque code constituant augmente, tandis que dans le second cas, le code produit souffre d'une perte d'efficacité. Pour remédier à ce problème, nous proposons d'utiliser des codes convolutifs comme codes constituants. Pour construire le code produit, il est alors nécessaire de les transformer en codes en blocs. Différentes méthodes sont disponibles à cet effet, dont la troncature du treillis du code convolutif ou l'ajout de bits de fermeture. L'inconvénient de telles techniques est que les derniers bits émis ne sont pas correctement protégés avec la première méthode, et que le rendement du code final est sacrifié avec la seconde méthode. Or, dans le cas des codes produits, pour $J = 2$ par exemple, la longueur du code ligne, comme celle du code colonne, est trop petite pour tolérer des bits de fermeture qui engendreraient des pertes en rendement trop importantes. Par conséquent, nous proposons d'utiliser des codes convolutifs à treillis circulaire (*tail-biting codes*, notés codes TB), qui permettent de convertir de manière efficace des codes convolutifs en codes en blocs quasi-cycliques sans diminuer le rendement du code. Ceci se fait en forçant l'état initial et l'état final du codeur à être identiques, grâce à la manipulation illustrée dans l'exemple suivant.

Afin de garder la structure du code produit représentée sur la figure 2.6, c'est-à-dire la séparation des bits d'information et des bits de parité, nous considérons un code convolutif systématique récursif (*code RSC*). Dans le cas d'un code non récursif, l'état final du codeur correspond aux derniers bits d'information transmis ; par conséquent, il est aisé de commencer et de finir dans le même état. Pour un code RSC, le problème est légèrement différent. Prenons l'exemple d'un code de rendement $R = 1/2$, de polynômes générateurs $g_1 = (5)_8 = [g_{11} \;\; g_{12} \;\; g_{13}] = [1 \;\; 0 \;\; 1]$ et $g_2 = (7)_8 = [g_{21} \;\; g_{22} \;\; g_{23}] = [1 \;\; 1 \;\; 1]$. Le polynôme g_2 est le polynôme de rétroaction. Une représentation du codeur canonique correspondant est donnée sur la figure 2.7.

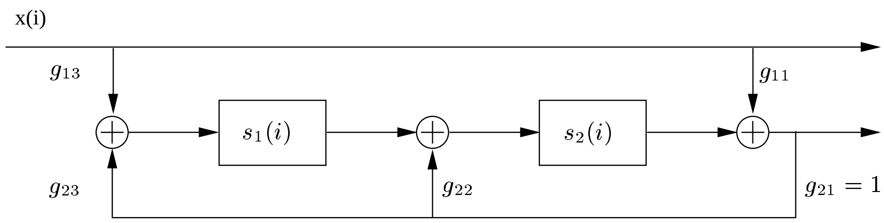


Fig. 2.7 – Représentation du codeur canonique du code RSC $(1, 5/7)$.

Soient $S(i) = \begin{bmatrix} s_1(i) \\ s_2(i) \end{bmatrix}$ l'état du codeur à l'instant i, et G et $X(i)$ les matrices définies par :

$$G = \begin{bmatrix} 0 & g_{23} \\ 1 & g_{22} \end{bmatrix} \qquad X(i) = \begin{bmatrix} (g_{13} + g_{11} \; g_{23})x(i) \\ (g_{11} \; g_{22})x(i) \end{bmatrix}.$$

$x(i)$ désigne l'entrée du codeur à l'instant i. L'état du codeur à l'instant $i + 1$ s'écrit alors :

$$S(i+1) = G \;\; S(i) \;\; + \;\; X(i),$$

soit :

$$S(i) = G^i S(0) \quad + \quad \sum_{p=1}^{i} G^{i-p} \ X(p) \ , \tag{2.1}$$

où $S(0)$ désigne l'état du codeur à l'instant initial.

Soient W la longueur de la séquence d'information et $S(W)$ l'état dans lequel se trouve le codeur après codage de cette séquence si l'état initial est $S(0) = 0$. L'état $S(W)$ vérifie :

$$S(W) = \sum_{p=1}^{W} G^{W-p} X(p) \ . \tag{2.2}$$

Soit S_c l'état circulant du codeur, c'est-à-dire l'état du codeur tel que son état final et son état initial soient égaux à S_c après encodage d'une séquence de longueur W. D'après l'équation (2.1), cet état vérifie :

$$S_c = G^W \ S_c + \sum_{p=1}^{W} G^{W-p} X(p) \ ,$$

soit, si la matrice $(I + G^W)$ est inversible (I est la matrice identité),

$$S_c = (I + G^W)^{-1} \sum_{p=1}^{W} G^{W-p} \ X(p) \ . \tag{2.3}$$

Par conséquent, en combinant les équations (2.2) et (2.3), nous obtenons la formule suivante, donnant l'état initial et l'état final du codeur convolutif à treillis circulaire :

$$S_c = (I + G^W)^{-1} S(W) \ . \tag{2.4}$$

Le code produit construit à partir d'un code TB présente les mêmes propriétés qu'un code produit construit à partir d'un code en blocs. Chaque code constituant est en effet un code en blocs linéaire binaire obtenu à partir d'un code RSC.

Nous avons présenté dans ce paragraphe différentes méthodes permettant de construire un code composite. Les codes résultant de ces trois méthodes de construction sont très similaires et peuvent être réunis dans la famille des codes de Tanner, c'est-à-dire des intersections de codes. En effet, un code produit n'est qu'un cas particulier des codes GLD blocs, pour lequel un entrelaceur ligne-colonne est considéré au lieu d'un entrelaceur aléatoire [62]. De même, nous montrerons au chapitre 3 que les concaténations série et parallèle, et notamment les turbo-codes, peuvent être vues comme des cas particuliers de codes GLD. Nous nous intéressons dans la partie suivante aux principaux résultats existant dans la littérature sur les performances théoriques de ces différents codes.

2.2 Performances analytiques avec décodage ML

Soit $\mathcal{C}(N,K)$ un code composite construit à partir de J codes constituants $C^1, \ldots, C^J$. Pour simplifier l'étude des performances analytiques de $\mathcal{C}(N,K)$, les codes constituants sont supposés identiques[4] à une permutation près, et représentent

- la somme directe d'équations de parité, pour les codes LDPC,
- la somme directe de codes BCH $C_0(n,k)$, pour les codes GLD blocs et les codes produits,
- des codes convolutifs, RSC ou non, pour les turbo-codes série et parallèle.

Nous supposons dans ce paragraphe que le code composite $\mathcal{C}$ est décodé de manière optimale selon le critère du maximum de vraisemblance. En pratique, un tel décodage n'est pas envisagé pour les codes composites du fait de sa trop grande complexité, mais les performances estimées sous cette hypothèse nous permettent de poser les limites de ces schémas de codage avec un décodage optimal, et de les comparer entre eux. Ces performances sont évaluées pour un ensemble de codes, réalisé en considérant tous les entrelaceurs aléatoires possibles.

2.2.1 Codes à faible densité LDPC

Gallager [42] a montré que lorsque $j \geq 3$, les codes LDPC sont asymptotiquement bons, *i.e.*, leur distance minimale croît de manière linéaire avec leur longueur N. Le nombre moyen de mots de code $\overline{A(w)}$ de poids w de l'ensemble des codes LDPC de paramètres (N,j,ℓ) vérifie en effet, lorsque N est suffisamment grand :

$$\overline{A(w)} \leq f(\lambda,N)\exp\left(-NB_{j\ell}(\lambda)\right) \tag{2.5}$$

où $\lambda = \frac{w}{N}$ est le poids normalisé des mots de code. Les fonctions $f(\lambda,N)$ et $B_{j\ell}(\lambda)$ sont définies par :

$$B_{j\ell}(\lambda) = (j-1)H(\lambda) - \frac{j}{\ell}\left[\mu(s) + (\ell-1)\ln 2\right] + js\lambda \ ,$$

$$f(\lambda,N) = \left[2\pi N\lambda(1-\lambda)\right]^{\frac{j-1}{2}} \exp\left(\frac{j-1}{12N\lambda(1-\lambda)}\right) \ .$$

H désigne l'entropie naturelle (écrite à l'aide du logarithme népérien). $\mu(s)$ est une fonction dépendant uniquement de ℓ et du paramètre s à optimiser, dont la valeur est donnée par :

$$\mu(s) = \ln\left(2^{-\ell}\left[(1+e^s)^\ell + (1-e^s)^\ell\right]\right) \ .$$

L'équation (2.5) montre que le comportement du nombre de mots de code moyen dépend du signe de $B_{j\ell}(\lambda)$. Si $B_{j\ell}(\lambda) > 0$ alors $\overline{A(w)}$ tend vers 0 lorsque N tend vers l'infini. Par conséquent, la plus grande valeur de λ telle que $B_{j\ell}(\lambda)$ soit strictement positif nous donne une borne inférieure de la distance de Hamming minimale moyenne $\delta_{j\ell}$ de l'ensemble des codes LDPC de paramètres (N,j,ℓ). Cette valeur est strictement positive si $j \geq 3$, et l'on observe que $\delta_{j\ell}$ s'approche de la borne de Gilbert Varshamov, donnée par :

$$\delta = H_2^{-1}(1-R) \ ,$$

lorsque j augmente. Pour de plus amples détails, le lecteur pourra se référer à [42].

[4] excepté pour les turbo-codes série

2.2.2 Codes GLD blocs

Un raisonnement similaire a été mené par [14] et [62] pour déterminer les performances ML des codes GLD blocs. Il a été montré en particulier que le nombre moyen de mots de code $\overline{A(w)}$ de poids w d'un code GLD construit à partir d'un code en blocs $C_0(n_0, k_0)$ et contenant J sous-matrices, vérifie :

$$\overline{A(w)} \leq f(\lambda, N) \exp\left[-NB(\lambda)\right] \ , \tag{2.6}$$

où les fonctions $f(\lambda, N)$ et $B(\lambda)$ sont définies par :

$$B(\lambda) = (J-1)H(\lambda) - \frac{J}{n_0}\left[\mu(s) + k_0 \ln 2\right] + Js\lambda \ ,$$

$$f(\lambda, N) = \left[2\pi N\lambda(1-\lambda)\right]^{\frac{J-1}{2}} \exp\left(\frac{J-1}{12N\lambda(1-\lambda)}\right) \ .$$

La fonction $\mu(s)$ est égale à $\ln(g(s))$, où $g(s)$ désigne le moment générateur de C_0, c'est-à-dire le polynôme exponentiel dont les coefficients g_i de degré i sont les nombres normalisés de mots de code de poids i de C_0. Par exemple, pour le code de Hamming $C_0(n_0 = 7, k_0 = 4)$, le moment générateur est donné par :

$$g(s) = \frac{1 + 7e^{3s} + 7e^{4s} + e^{7s}}{16} \ .$$

De même que pour les codes LDPC, le comportement du nombre moyen de mots de code de poids w est lié au signe de $B(\lambda)$. Le paramètre s doit être optimisé afin de maximiser la valeur de $B(\lambda)$, $\forall\ \lambda$. Nous obtenons alors une borne inférieure pour la distance minimale normalisée du code GLD, correspondant à la plus grande valeur de λ telle que $B(\lambda)$ soit positive. Notons que pour $J = 2$ les valeurs trouvées pour cette distance minimale normalisée sont toutes strictement positives [62], ce qui signifie que les codes GLD sont asymptotiquement bons à partir de $J = 2$ sous-matrices seulement. Lorsque le nombre de sous-matrices augmentent, les GLD blocs s'approchent de la borne de Gilbert Varshamov.

Par ailleurs, une formule exacte du nombre de mots de code moyen est donnée par :

$$\overline{A(w)} = \frac{2^{\frac{Jk_0N}{n_0}} R(w)^J}{\left(C_N^w\right)^{J-1}} \ , \tag{2.7}$$

où $R(w)$ est la probabilité qu'un mot de code de poids w soit un mot du code C^j, et C_N^w désigne le coefficient binômial. Comme C^j est la somme directe de $\frac{N}{n_0}$ sous-codes C_0, $R(w)$ vérifie la relation suivante :

$$\left[g(s)\right]^{\frac{N}{n_0}} = \sum_w R(w)e^{ws} \ .$$

A partir de l'équation (2.7), nous pouvons obtenir une borne supérieure des probabilités d'erreur moyennes par mot et par bit, en utilisant la borne de l'union :

$$\overline{Pe_{\text{w}}} \leq \sum_{w=1}^{N} \overline{A(w)}\ Q\left(\sqrt{2Rw\frac{E_b}{N_0}}\right) \ ,$$

$$\overline{Pe_b} \leq \sum_{w=1}^{N} \frac{w}{N} \overline{A(w)} \left(\sqrt{2Rw \frac{E_b}{N_0}} \right) .$$

La fonction Q est la fonction d'erreur définie par :

$$Q(x) = \frac{1}{2\pi} \int_{x}^{+\infty} e^{-\frac{t^2}{2}} dt .$$

2.2.3 Turbo-codes

Les performances ML des turbo-codes série et parallèle ont été étudiées en détail dans [8][9]. Nous n'avons repris ici que les principaux résultats de cette étude.

Turbo-codes parallèle

Afin de simplifier l'étude des performances ML des turbo-codes parallèle et les notations employées, nous nous restreignons par la suite à un turbo-code construit à partir de deux codes RSC identiques, $C^1 = C^2$, de rendement $R_1 = 1/2$. Le rendement du turbo-code résultant est égal à $R = 1/3$. La terminaison des treillis est négligée, et l'on note N_π la taille de l'entrelaceur.

Le polynôme énumérateur de poids du code C^1 s'écrit :

$$A^{C^1}(I, Z) = \sum_{i=i_{\min}}^{N_\pi} \sum_{z=z_{\min}}^{N_\pi} A_{i,z}^{C^1} I^i Z^z ,$$

où $A_{i,z}^{C^1}$ est le nombre de mots de code de C^1 correspondant à un poids i de l'information et un poids z de la parité. Le polynôme $A^{C^1}(I, Z)$ n'inclut pas le mot $0, \ldots, 0$. Pour un code RSC dont le treillis est fermé, le poids minimal en entrée est $i_{\min} = 2$. Le poids minimal en sortie $z_{\min}$ ne correspond pas forcément au poids minimal en entrée ; si on note $z_{\min}^{(2)}$ le poids de la parité qui correspond à un poids 2 en entrée, alors en général $z_{\min} \leq z_{\min}^{(2)}$.

Le polynôme $A^{C^1}(I, Z)$ s'écrit également sous la forme suivante :

$$A^{C^1}(I, Z) = \sum_{i=i_{\min}}^{N_\pi} I^i A^{C^1}(i, Z) \qquad \text{avec} \qquad A^{C^1}(i, Z) = \sum_{z=z_{\min}}^{N_\pi} A_{i,z}^{C^1} Z^z .$$

$A^{C^1}(i, Z)$ est appelé polynôme énumérateur de poids conditionnel, puisque le poids i de l'entrée est fixé. Une méthode pour calculer ce polynôme est présentée dans [8] et [27].

La distribution de poids du turbo-code $\mathcal{C}$ considéré est calculée en moyennant sur tous les entrelaceurs possibles. L'entrelaceur du turbo-code étant aléatoire, les entrées des deux codeurs sont indépendantes. Par conséquent il est aisément vérifié que :

$$A^{\mathcal{C}}(i, Z) = \frac{\left[A^{C^1}(i, Z) \right]^2}{C_{N_\pi}^i} .$$

En considérant alors la borne de l'union, une borne supérieure de la probabilité d'erreur par bit est donnée par :

$$Pe_{\mathrm{b}} \ \lessapprox \ \frac{1}{2} \sum_{i=i_{\min}}^{N_\pi} \frac{i}{N_\pi} I^i A^C(i,Z) \quad \text{avec} \ \ I = Z = e^{-R\frac{E_b}{N_0}} \ .$$

L'inconvénient majeur de cette borne est qu'elle diverge pour de faibles rapports signal-à-bruit. Afin de préciser l'influence de l'entrelaceur et des codes constituants dans cette zone, Benedetto *et al.* ont décomposé les mots de code en événements d'erreurs simples [8], ce qui leur a permis d'obtenir la formule suivante pour un turbo-code construit à partir de deux codes RSC :

$$Pe_{\mathrm{b}} \ \lessapprox \ \frac{1}{2} \sum_{\substack{i=1 \\ i \ pair}}^{N_\pi} \frac{i}{N_\pi} \frac{\left(H^{2+2z_{\min}^{(2)}}\right)^{\frac{i}{2}}}{\left(1 - H^{z_{\min}^{(2)}-2}\right)^i} \quad \text{avec} \quad H = e^{-R\frac{E_b}{N_0}} \ . \tag{2.8}$$

En observant la probabilité d'erreur par bit Pe_{b}, on remarque que les turbo-codes construits à partir de codes RSC présentent un gain d'entrelacement[5], *i.e.*, la probabilité d'erreur par bit décroît en $\frac{1}{N_\pi}$.

A fort rapport signal-à-bruit, le facteur déterminant dans les performances ML d'un code est sa distance minimale. La borne (2.8) montre qu'une approximation de la distance minimale asymptotique des turbo-codes construits à partir de deux codes RSC est $d_{\text{eff}} \ = \ 2 + 2z_{\min}^{(2)}$, aussi appelée *distance effective* du turbo-code. L'emploi de polynômes de rétroaction primitifs permet d'augmenter $z_{\min}^{(2)}$ et donc la distance effective d_{eff}.

Plus généralement, Kahale et Urbanke [50] ont montré que la distance minimale d'un turbo-code parallèle construit à partir de J codes constituants RSC varie en $(N_\pi)^{\frac{J-2}{J}}$. Par conséquent, si le nombre de codes constituants est égal à 2, comme envisagé ci-dessus, la distance minimale du turbo-code résultant n'est plus fonction de la taille de l'entrelaceur, ni de la longueur du code N. Cette faible distance minimale est à l'origine du phénomène de palier (ou *error floor*) rencontré dans les performances des turbo-codes parallèle pour une probabilité d'erreur faible. Il est possible d'agir contre ce phénomène en choisissant soigneusement l'entrelaceur utilisé pour la construction du turbo-code [29]. De plus, la variation de la distance minimale étant une fonction croissante de J, augmenter le nombre de codes constituants du turbo-code permet asymptotiquement d'obtenir de meilleures performances à fort rapport signal-à-bruit avec un décodage à maximum de vraisemblance.

Turbo-codes série

Dans le cas des turbo-codes série, le calcul des performances ML est similaire.

Soient C^1 le code extérieur de dimension K, de longueur N_π, de distance minimale d_1, et C^2 le code intérieur de dimension N_π, de longueur N, et de distance minimale d_2. La longueur N_π est également la taille de l'entrelaceur positionné entre le code extérieur et le code intérieur. Le turbo-code série $\mathcal{C}$ est de dimension K et de longueur N.

[5]ce n'est pas le cas si les codes constituants ne sont pas RSC

Le polynôme énumérateur de poids du turbo-code série s'exprime comme la convolution normalisée des polynômes énumérateurs de poids des deux codes constituants :

$$A^{\mathcal{C}}(I,W) = \sum_{m=d_1}^{N_\pi} \frac{A^{C^1}(I,m)A^{C^2}(m,W)}{C_{N_\pi}^m} \ ,$$

où la variable muette W désigne cette fois-ci le poids total en sortie du turbo-code série (et non plus le poids de la parité uniquement comme le considérait la variable Z). Par conséquent, en introduisant le polynôme énumérateur de poids conditionnel $A^{\mathcal{C}}(i,W)$ tel que :

$$A^{\mathcal{C}}(I,W) = \sum_{i=i_{\min}}^{K} I^i A^{\mathcal{C}}(i,W) \ ,$$

nous obtenons comme borne supérieure de la probabilité d'erreur par bit :

$$Pe_{\mathrm{b}} \ \tilde{\leq} \ \frac{1}{2} \sum_{i=i_{\min}}^{K} \frac{i}{K} I^i A^{\mathcal{C}}(i,W) \quad \text{avec} \ \ I = W = e^{-R\frac{E_b}{N_0}} \ .$$

Un raisonnement fondé sur l'étude des événements d'erreurs simples des codes intérieurs et extérieurs permet de montrer que pour obtenir un gain d'entrelacement, le code intérieur C^2 doit être récursif et systématique. Le gain obtenu est alors en $\frac{1}{(N_\pi)^{\frac{d_1+1}{2}}}$, *i.e.*, le gain d'entrelacement obtenu pour un turbo-code série est supérieur à celui d'un turbo-code parallèle. En général, on choisit un code extérieur non récursif non systématique afin d'augmenter la distance d_1 et par conséquent de favoriser le gain d'entrelacement.

Ce choix permet également d'augmenter la distance minimale du turbo-code série. En effet, Kahale et Urbanke [50] ont montré que la distance minimale d'un turbo-code série varie en $(N_\pi)^{\frac{d_1-2}{d_1}}$. En choisissant le code extérieur de manière appropriée pour optimiser sa distance minimale d_1, il est possible de rendre l'évolution de la distance minimale du turbo-code série quasi-linéaire.

2.2.4 Codes produits

La distance minimale d'un code produit est uniquement fonction de la distance minimale de ses codes constituants. En effet, nous avons vu que pour $J = 2$, la distance minimale est $d_{\min} = d_1 \ d_2$, si d_1 et d_2 sont les distances minimales respectives des codes C^1 et C^2. Par conséquent, cette distance minimale n'est pas du tout fonction de la longueur du code produit.

En utilisant la borne de l'union comme précédemment, nous obtenons une borne supérieure pour la probabilité d'erreur par mot et pour la probabilité d'erreur par bit :

$$Pe_{\mathrm{w}} \leq \sum_{w=1}^{N} A(w) \ Q\left(\sqrt{2Rw\frac{E_b}{N_0}}\right) \ , \tag{2.9}$$

$$Pe_{\mathrm{b}} \leq \sum_{w=1}^{N} \frac{w}{N} A(w) \ Q\left(\sqrt{2Rw\frac{E_b}{N_0}}\right) \ ,$$

où $A(w)$ est le nombre de mots de code de $\mathcal{C}$ de poids w. La quantité $A(w)$ ne peut être déterminé entièrement à partir des polynômes énumérateurs de poids des codes constituants. Seuls les poids faibles peuvent être ainsi évalués. Il a notamment été montré dans [79] que pour des codes binaires, et pour $1 \leq w < d_1\ d_2 + d_2$:

$$A(w) = \begin{cases} 0 & \text{si } w \text{ non divisible par } d_1, \\ A^{C^2}\left(\frac{w}{d_1}\right) A^{C^1}(d_1) & \text{si } w \text{ divisible par } d_1, \end{cases}$$

lorsque $A^{C^1}(w)$ et $A^{C^2}(w)$ représentent le nombre de mots de code de poids w de C^1 et C^2 respectivement. Si $(d_1 d_2 < w < d_1 d_2 + \min(d_1, d_2))$, le code $\mathcal{C}$ n'a pas de mot de code de poids w. Cette distribution, même partielle, permet d'obtenir une approximation de la probabilité d'erreur par bit d'un code produit à fort rapport signal-à-bruit, en ne prenant en compte que le premier terme de la somme (terme dominant) dans l'inégalité (2.9) :

$$Pe_b \underset{\sim}{\leq} \frac{d_1 d_2}{N} A(d_1 d_2)\ Q\left(\sqrt{2Rd_1 d_2 \frac{E_b}{N_0}}\right) .$$

Deux types de comportements ont été observés à travers ce rappel des performances analytiques des codes composites :

- un comportement à «faible» rapport signal-à-bruit : nous avons vu que les turbo-codes, aussi bien parallèle que série, présentent un gain d'entrelacement si les codes constituants sont habilement choisis (codes RSC). Un tel comportement n'a pu être mis en évidence pour les codes GLD ou pour les codes LDPC.
- un comportement à «fort» rapport signal-à-bruit : l'évolution de la distance minimale en fonction de la longueur du code ou de la taille de l'entrelacement a été étudiée. Les codes LDPC et les codes GLD présentent ainsi une distance minimale qui croît linéairement avec la longueur du code N, tandis que pour les turbo-codes ou les codes produits, la croissance est logarithmique, voire indépendante de la longueur N.

Ces performances ont été évaluées en supposant que le décodage employé est à maximum de vraisemblance. En pratique un décodage ML est difficile à mettre en œuvre pour de tels codes, du fait d'une complexité prohibitive. On utilise un décodage itératif permettant, à fort rapport signal-à-bruit, d'obtenir des performances similaires au décodage ML et de se rapprocher de ces performances à faibles rapports signal-à-bruit selon le nombre d'itérations considérées. Il est alors particulièrement intéressant de représenter les codes sous forme graphique afin de mieux comprendre la propagation de l'information au cours des itérations de décodage.

2.3 Représentation graphique

Récemment, de nombreux chercheurs se sont intéressés à la représentation graphique des codes ; citons notamment Wiberg, Loeliger et Kötter [83][84], ou Kschichang et Frey [52]. Ces représentations graphiques sont très utiles pour décrire et étudier les techniques de décodage itératif et la propagation de l'information lors de ce décodage. Nous présentons dans cette partie les deux représentations graphiques les plus usuelles, la représentation bayésienne et la représentation graphique de Tanner, et nous donnons quelques exemples choisis parmi les codes composites étudiés précédemment.

2.3.1 Représentation bayésienne

Un réseau bayésien est un graphe sans cycle et orienté, autrement dit les directions des arêtes sont représentées sur le graphe[6]. La direction des arêtes est particulièrement utile pour modéliser de manière explicite la causalité ; en observant cette direction, il est possible par exemple de comprendre quelles sont les variables qui influencent les autres directement. Ainsi, en codage, les relations entre les variables caractérisant les symboles d'information $\mathbf{u}$, l'état du codeur $\mathbf{s}$, la sortie du codeur $\mathbf{x}$ et la sortie du canal $\mathbf{y}$, peuvent être aisément représentées sur un graphe bayésien, comme le montre la figure 2.8. Ce graphe montre que la sortie du codeur $\mathbf{x}$ est influencée par les

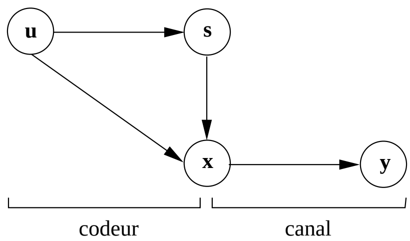


Fig. 2.8 – Graphe bayésien d'un schéma de codage de canal

symboles d'information, mais également par l'état du codeur (notamment dans le cas d'un code convolutif, où l'effet de mémoire du code est représenté par l'arête reliant $\mathbf{s}$ à $\mathbf{x}$). Les relations entre ces variables peuvent être exprimées à l'aide de probabilités. En particulier, en observant le réseau représenté sur la figure 2.8, on peut écrire que la distribution conjointe de $(\mathbf{u}, \mathbf{s}, \mathbf{x}, \mathbf{y})$ vérifie :

$$\Pr(\mathbf{u}, \mathbf{s}, \mathbf{x}, \mathbf{y}) = \Pr(\mathbf{u}) \ \Pr(\mathbf{s}|\mathbf{u}) \ \Pr(\mathbf{x}|\mathbf{u}, \mathbf{s}) \ \Pr(\mathbf{y}|\mathbf{x}) \ .$$

Les distributions $\Pr(\mathbf{x}|\mathbf{u}, \mathbf{s})$ et $\Pr(\mathbf{s}|\mathbf{u})$ sont en général déterministes, et $\Pr(\mathbf{y}|\mathbf{x})$ caractérise l'observation du canal. Pour un code de longueur N émis sur un canal AWGN sans mémoire, dont le bruit est de moyenne nulle et de variance N_0, cette observation est donnée par :

$$\Pr(\mathbf{y}|\mathbf{x}) = \left(\frac{1}{\sqrt{2\pi N_0}} \right)^N \exp\left(-\sum_{i=1}^{N} \frac{(y_i - x_i)^2}{2N_0} \right) \ .$$

Si nous appliquons cette représentation bayésienne à un code convolutif systématique de rendement $\frac{1}{2}$ sur un canal sans mémoire, nous obtenons alors le schéma de la figure 2.9. Cette représentation peut être simplifiée en réunissant les sommets désignant les variables $\mathbf{u}$, $\mathbf{s}$, $\mathbf{x}$, et $\mathbf{y}$ en un *super-nœud* ou *super-sommet* de degré n. Le graphe du code convolutif se réduit alors à une simple chaîne orientée. La représentation bayésienne d'un turbo-code parallèle est ainsi très

[6]Le graphe obtenu si l'on ignore les directions peut cependant contenir des cycles.

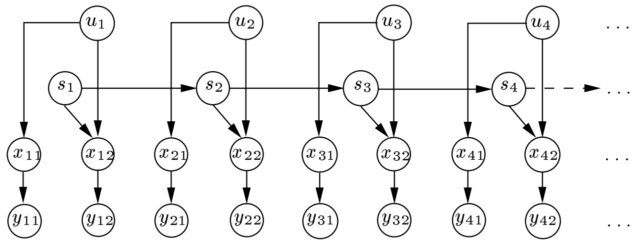


FIG. 2.9 – Graphe bayésien d'un code convolutif systématique

simple. Considérons par exemple que ce turbo-code est constitué de deux codes RSC, C^1 et C^2, de rendement $R_1 = R_2 = \frac{1}{2}$ et d'un entrelaceur de longueur N_π. Chaque code constituant est représenté par une chaîne orientée ayant N_π super-sommets et $N_\pi - 1$ arêtes. Le réseau bayésien du turbo-code est obtenu en liant les deux chaînes orientées à l'aide d'arêtes choisies de manière aléatoire, ce qui correspond à l'entrelaceur du turbo-code. L'entrelaceur n'est pas orienté car il n'introduit pas de causalité particulière.

La figure 2.10 donne le réseau bayésien d'un turbo-code dont la taille de l'entrelaceur est égale à 10, et pour des codes constituants de rendement $R = 1/2$.

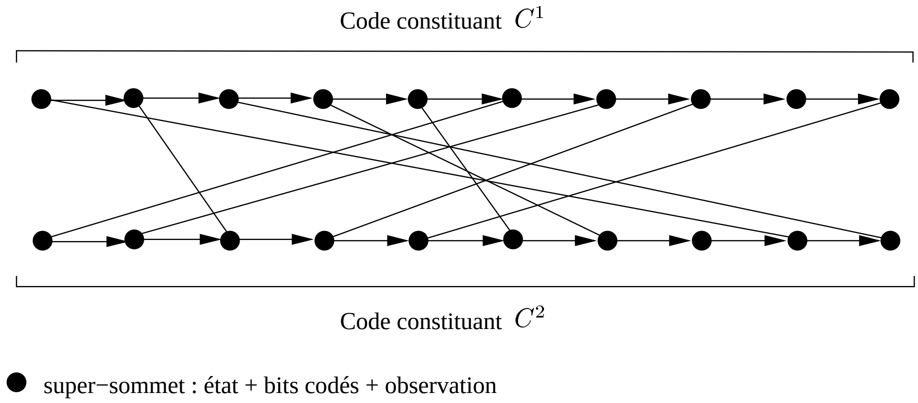


FIG. 2.10 – Réseau bayésien d'un turbo-code parallèle

2.3.2 Représentation de Tanner

Les graphes de Tanner furent introduits dans [76] afin de construire de longs codes correcteurs d'erreurs à partir de petits codes, en utilisant des entrelaceurs déterministes. Un graphe de Tanner est un graphe bipartite qui possède deux types de sommets caractérisant d'une part les bits codés et d'autre part les contraintes sur ces bits. Aucune arête ne relie deux sommets de même type. Une arête existe entre un sommet bit et un sommet contrainte si le bit respecte la contrainte.

Ainsi, si l'on considère le code LDPC (N, j, ℓ) dont la matrice de parité est représentée sur la figure 2.3, on obtient un graphe possédant N sommets bits et un nombre N_{nc} de sommets contraintes. Chaque sommet bit est relié à j sommets contraintes, tandis que chaque sommet contrainte est relié à ℓ sommets bits. Le graphe est régulier : le nombre d'arêtes partant d'un sommet d'un sous-ensemble est le même pour tous les sommets de ce sous-ensemble ; nous appelons degré d'un sommet le nombre d'arêtes partant de ce sommet. Le nombre total d'arêtes partant des sommets bits est le même que le nombre total d'arêtes partant des sommets contraintes d'où $N_{nc} = N\frac{j}{\ell}$. La figure 2.11 représente le graphe de Tanner d'un code LDPC (N, j, ℓ). Ce graphe est un graphe aléatoire, puisque les arêtes sont placées de manière aléatoire entre les bits et les contraintes, les permutations π_i, $i = 2, \ldots, j$, étant aléatoires.

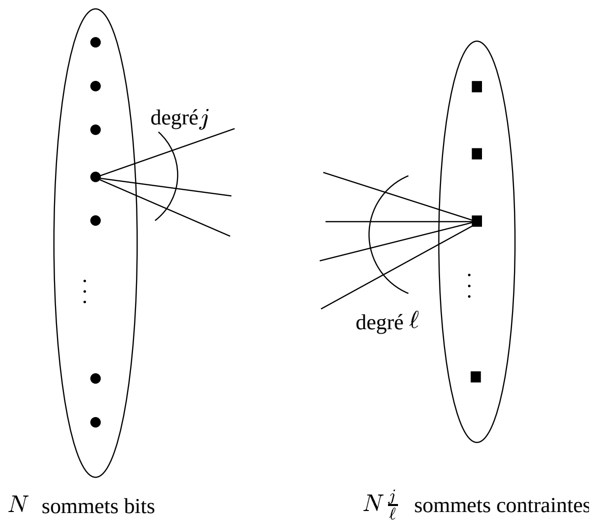


Fig. 2.11 – Graphe de Tanner d'un code LDPC (N, j, ℓ)

Dans le cas d'un code GLD blocs, les sommets contraintes symbolisent les codes en blocs binaires constituant le code GLD. Pour un turbo-code parallèle ou pour un turbo-code série, chaque sommet contrainte regroupe un grand nombre d'équations de parité sur les bits : chaque sommet contrainte correspond en fait à un code convolutif entier ou, de manière équivalente, à son treillis.

Comme nous le verrons dans la section suivante et au chapitre 4, la représentation bayésienne et la représentation de Tanner permettent de visualiser le cheminement de l'information lors du décodage itératif des codes composites. Notons qu'il est possible de passer d'une représentation à l'autre simplement en orientant ou non les arêtes vers les sommets contraintes.

2.4 Décodage itératif

Le décodage à maximum de vraisemblance des codes composites se révèle trop complexe ; cependant, il est possible de se rapprocher de ses performances en utilisant un processus de décodage itératif, fondé sur le décodage successif ou simultané de chaque code élémentaire constituant le code composite.

Afin de minimiser la perte d'information à chaque décodage et d'améliorer ainsi les performances, il est nécessaire de considérer des décodeurs à entrées souples, appelés *décodeurs SI* (décodeurs *Soft Input*). L'information souple représente une valeur de fiabilité sur les bits codés (ou sur les bits d'information). Nous présentons dans ce paragraphe deux algorithmes de décodage souple : l'algorithme aller-retour de Bahl *et al.* [2] (aussi connu sous le nom d'algorithme *forward-backward*) et l'algorithme de Fossorier-Lin [37]. Le premier algorithme est optimal pour le critère du maximum *a posteriori*, et peut être utilisé pour les codes convolutifs comme pour les codes en blocs, moyennant un treillis de taille raisonnable. Le second algorithme, sous-optimal, peut être appliqué à n'importe quel code en blocs linéaire puisqu'il ne nécessite pas de décodeur algébrique.

Pour itérer le processus de décodage, chaque décodeur élémentaire doit générer une sortie souple. Il s'agit alors d'un décodeur *SISO* (décodeur *Soft Input Soft Output*), c'est-à-dire à entrée souple et à sortie souple. Cette sortie peut être obtenue en adaptant les algorithmes précédemment cités. L'algorithme aller-retour fournit déjà une valeur de fiabilité sur les bits codés, qui doit être légèrement modifiée pour permettre le processus itératif. A l'opposé, l'algorithme de Fossorier-Lin ne délivre aucune information souple sur les bits codés dans sa forme initiale. Après avoir expliqué les principes du décodage itératif, nous présentons les modifications apportées à ce dernier algorithme afin qu'il génère en sortie des informations de fiabilité sur les bits.

Pour simplifier la description des algorithmes, nous considérons par la suite que les symboles codés sont modulés par une modulation BPSK, puis émis sur un canal AWGN discret sans mémoire. Le bruit gaussien est de moyenne nulle et de variance $\sigma^2 = N_0$.

2.4.1 Principe du décodage itératif

Le principe du décodage itératif repose sur la propagation d'informations souples au cours des itérations. Deux types d'informations souples sont générés à la sortie de chaque décodeur SISO : une probabilité *a posteriori* et une information extrinsèque sur les bits codés (ou uniquement sur les bits d'information selon les codes composites envisagés).

Soit $C(N, K)$ un code en blocs linéaire[7] et $\mathbf{c} = (c_1, \ldots, c_N)$ un mot de code de C. La probabilité *a posteriori* (ou APP pour *a posteriori probability*) d'un bit codé c_i, $i = 1, \ldots, N$, de $\mathbf{c}$ est définie par :

$$APP(c_i) = \Pr(c_i|\mathbf{y}) = \sum_{\mathbf{c}\in C} \Pr(c_i, \mathbf{c}|\mathbf{y}) = \sum_{\mathbf{c}\in C} \frac{\Pr(c_i, \mathbf{c})\Pr(\mathbf{y}|c_i, \mathbf{c})}{\Pr(\mathbf{y})} \,, \tag{2.10}$$

où $\mathbf{y}$ est la séquence des symboles reçus. L'observation $\Pr(\mathbf{y}|c_i, \mathbf{c})$ et la probabilité *a priori* $\Pr(c_i, \mathbf{c})$ sont nulles si la j-ième composante de $\mathbf{c}$ n'est pas égale à c_i. Par ailleurs, si le canal est sans mémoire et que les probabilités *a priori* des bits codés sont supposées indépendantes, l'équation (2.10) devient :

$$APP(c_i) \propto \sum_{\mathbf{c}\in C|c_i} \Pr(\mathbf{y}|\mathbf{c})\Pr(\mathbf{c}) = \sum_{\mathbf{c}\in C|c_i} \prod_{\ell=1}^{N} \Pr(y_\ell|c_\ell)P(c_\ell) \,, \tag{2.11}$$

[7] ce code peut être obtenu par fermeture d'un code convolutif $C_c(n, k)$

où $P(c_\ell)$ désigne la probabilité *a priori* du bit c_ℓ. Lorsqu'une probabilité *a priori* est disponible pour chaque bit codé à l'entrée du décodeur, l'APP du bit codé c_i s'écrit sous la forme :

$$APP(c_i) \propto (\text{observation}) \times (\text{probabilité } a\ priori) \times (\text{information extrinsèque})\ .$$

Son information extrinsèque est définie par :

$$Ext(c_i) \propto \sum_{\mathbf{c}\in C|c_i} \prod_{\substack{\ell=1\\ \ell\neq i}}^{N} \Pr(y_\ell|c_\ell)P(c_\ell)\ . \tag{2.12}$$

L'information extrinsèque ainsi générée à la sortie du décodeur est alors entrelacée (ou désentrelacée suivant le cas) puis transmise au décodeur suivant qui s'en sert comme nouvelle information *a priori* sur les bits codés ou sur les bits d'information. Ceci est représenté sur la figure 2.12 pour $J = 2$ codes constituants. A la sortie de chaque décodeur, les informations extrinsèques ne sont pas indépendantes, mais la présence de l'entrelaceur permet d'avoir en entrée du décodeur suivant des probabilités *a priori* indépendantes sur les bits.

La probabilité *a posteriori* du bit c_i est utilisée pour prendre une décision $\hat{c}_i$ sur ce bit et pour connaître la fiabilité de cette décision.

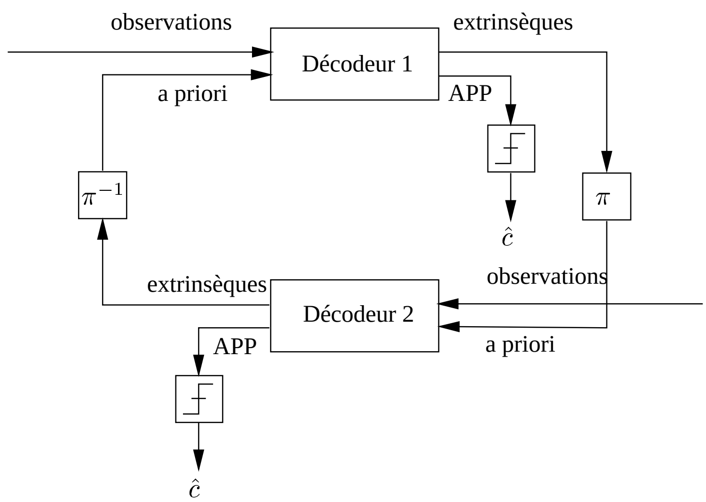


Fig. 2.12 – Décodage itératif d'un code composite construit à partir de deux codes constituants

Une itération consiste donc à décoder une fois chaque code constituant (sur la figure 2.12, on décode le premier code constituant puis le second). Dans le cas des codes GLD blocs ou des codes produits, chaque code constituant est formé de plusieurs codes en blocs élémentaires. Ces codes élémentaires peuvent être décodés en parallèle, puisqu'ils sont indépendants et agissent sur des bits différents. Ceci permet de réduire le délai introduit par le décodage.

Nous allons maintenant présenter deux algorithmes de décodage souple, l'algorithme aller-retour et l'algorithme de Fossorier-Lin, et préciser comment ces algorithmes sont adaptés afin d'être utilisés dans un processus de décodage itératif.

2.4.2 Algorithme aller-retour

L'algorithme aller-retour est un algorithme de décodage optimal, maximisant la probabilité *a posteriori* des bits. Tandis que l'algorithme de Viterbi [82][35] minimise la probabilité d'erreur par mot, l'algorithme aller-retour minimise, quant à lui, la probabilité d'erreur par symbole, ou comme dans notre cas, la probabilité d'erreur par bit. Il s'applique directement sur le treillis du code considéré. Par conséquent, il peut être utilisé pour le décodage des codes convolutifs comme pour celui des codes en blocs linéaires, moyennant une complexité du treillis raisonnable.

Décodage des codes convolutifs

Soit $C_c(n,k)$ un code convolutif de longueur de contrainte L prenant k bits d'information en entrée et délivrant n bits codés en sortie. Supposons sans perdre en généralités que ce code est récursif et systématique. Le treillis correspondant à ce code convolutif possède $M = 2^{k(L-1)}$ états, et est supposé fermé, i.e, le codeur commence et finit dans l'état 0. Pour cela, $\lceil \frac{L-1}{k} \rceil$ branches sont nécessaires dans le treillis. Le code convolutif résultant est alors équivalent à un code en blocs de longueur N et de dimension K ; son treillis contient $W = \frac{N}{n}$ branches (*cf.* figure 2.13).

Les notations suivantes sont utilisées par la suite :
- t désigne le temps, $t = 0, \dots, W$
- $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_W)$ est l'entrée du canal discret sans mémoire (*Discrete Memoryless Channel*, DMC). $\mathbf{x}_t,\ t = 1, \dots, W$ est un vecteur de n bits codés correspondant à la sortie du codeur modulée par une BPSK, à l'instant t.
  $\mathbf{x}_t$ s'écrit sous la forme $(x_{t,1}, \dots, x_{t,k}, x_{t,k+1}, \dots, x_{t,n})$ où $(x_{t,1}, \dots, x_{t,k})$ sont les bits d'information et $(x_{t,k+1}, \dots, x_{t,n})$ sont les bits de parité.
- $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_W)$ est la sortie du canal correspondant à $\mathbf{x}$. Comme le canal est DMC, $\mathbf{y_{t,i}}$ ne dépend que de $\mathbf{x}_{t,i}$ à l'instant t, pour $i = 1, \dots, n$.
- $\mathbf{x}_t^{t'}$ (respectivement $\mathbf{y}_t^{t'}$) désigne la séquence en entrée (respectivement en sortie) du canal entre les instants t et t'.
- S_t désigne l'état du treillis considéré : S_t peut prendre les valeurs $0, \dots, M-1$.

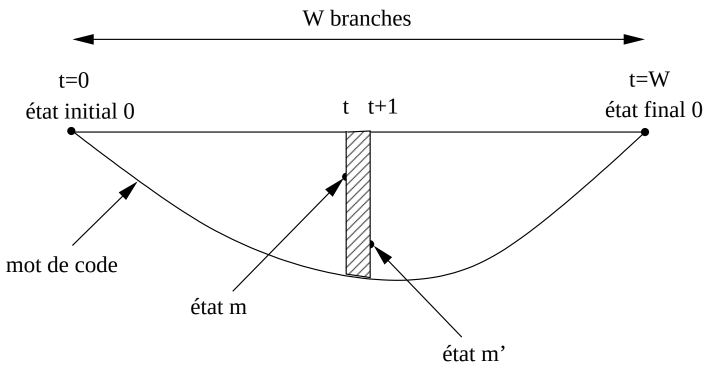


FIG. 2.13 – Treillis d'un code convolutif

L'algorithme aller-retour maximise la probabilité *a posteriori* d'un symbole $x_{t,i}$, $t = 1, \ldots, W$, $i = 1, \ldots, n$ définie comme la probabilité d'avoir ce symbole $x_{t,i}$ sachant toute l'observation du canal $\mathbf{y}_1^W$:

$$APP(x_{t,i}) = \Pr(x_{t,i}|\mathbf{y}_1^W) \ .$$

Cette APP peut s'écrire :

$$APP(x_{t,i}) = \sum_{m,m'|x_{t,i}} \sigma_t(m,m') \ ,$$

où $\sigma_t(m,m')$ désigne la probabilité *a posteriori* de la transition de l'état m vers l'état m' à l'instant t, soit :

$$\sigma_t(m,m') = \Pr(S_{t-1} = m, S_t = m', \mathbf{y}_1^W) \ .$$

En appliquant la règle de Bayes :

$$\Pr(A_1, A_2, ..., A_W) = \Pr(A_1)\Pr(A_2|A_1)\Pr(A_3|A_2, A_1)...\Pr(A_W|A_1, A_2, ..., A_{W-1}) \ ,$$

nous obtenons :

$$\sigma_t(m,m') = \Pr(S_{t-1} = m, \mathbf{y}_1^{t-1})\Pr(S_t = m', \mathbf{y}_t|S_{t-1} = m, \mathbf{y}_1^{t-1})\Pr(\mathbf{y}_{t+1}^W|S_{t-1} = m, \mathbf{y}_1^{t-1}, S_t = m', \mathbf{y}_t) \ .$$

Les séquences $\mathbf{y}_1^{t-1}$, $\mathbf{y}_t$ et $\mathbf{y}_{t+1}^W$ représentent respectivement le passé, le présent et le futur de $\mathbf{y}$.

Considérons les variables suivantes :

– la métrique de l'algorithme :

$$\gamma_t(m,m') = \Pr(S_t = m', \mathbf{y}_t|S_{t-1} = m) \ , \tag{2.13}$$

– la densité de probabilité conjointe de l'état m et de l'observation entre les instants 1 et t :

$$\alpha_t(m) = \Pr(S_t = m, \mathbf{y}_1^t) \ , \tag{2.14}$$

– la densité de probabilité de l'observation du canal entre les instants $t+1$ et W conditionnellement à l'état m :

$$\beta_t(m) = \Pr(\mathbf{y}_{t+1}^W|S_t = m) \ . \tag{2.15}$$

Le canal étant sans mémoire, la probabilité *a posteriori* de la transition s'écrit :

$$\begin{aligned} \sigma_t(m,m') &= \Pr(S_{t-1} = m, \mathbf{y}_1^{t-1})\Pr(S_t = m', \mathbf{y}_t|S_{t-1} = m)\Pr(\mathbf{y}_{t+1}^W|S_t = m') \\ &= \alpha_{t-1}(m) \times \gamma_t(m,m') \times \beta_t(m') \ . \end{aligned}$$

Intéressons nous maintenant à la métrique γ de l'algorithme :

$$\begin{aligned} \gamma_t(m,m') &= \Pr(S_t = m', \mathbf{y}_t|S_{t-1} = m) \\ &= \sum_{\mathbf{x}_t} \Pr(\mathbf{x}_t, S_t = m', \mathbf{y}_t|S_{t-1} = m) \\ &= \sum_{\mathbf{x}_t} p_t(m'|m)q_t(m,m')\Pr(\mathbf{y}_t|\mathbf{x}_t) \ , \end{aligned}$$

où

- $q_t(m,m') = \Pr(\mathbf{x}_t|S_{t-1} = m, S_t = m')$ est une fonction de Kronecker, égale à 1 si la transition entre m et m' étiquetée par $\mathbf{x}_t$ existe et 0 sinon. Cette valeur provient directement du treillis du code convolutif considéré.
- $p_t(m'|m) = \Pr(S_t = m'|S_{t-1} = m)$ est la probabilité *a priori* de la transition $m \to m'$,
- $\Pr(\mathbf{y}_t|\mathbf{x}_t)$ est la probabilité de transition du canal. D'après les hypothèses faites sur le canal,

$$\Pr(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{n} \Pr(y_{t,i}|x_{t,i}) = \left(\frac{1}{\sqrt{2\pi N_0}}\right)^n \exp\left(-\sum_{i=1}^{n} \frac{(y_{t,i} - x_{t,i})^2}{2N_0}\right) .$$

Dans l'expression de $\gamma_t(m,m')$, la somme $\sum_{\mathbf{x}_t}$ est égale à 0 sauf pour la valeur de $\mathbf{x}_t$ correspondant à la transition $m \to m'$. Par conséquent $\gamma_t(m,m')$ s'écrit :

$$\gamma_t(m,m') = q_t(m,m')p_t(m'|m)\Pr(\mathbf{y}_t|\mathbf{x}_t) .$$

L'algorithme se résume donc au diagramme de la figure 2.14. Les procédures aller (*forward*) et

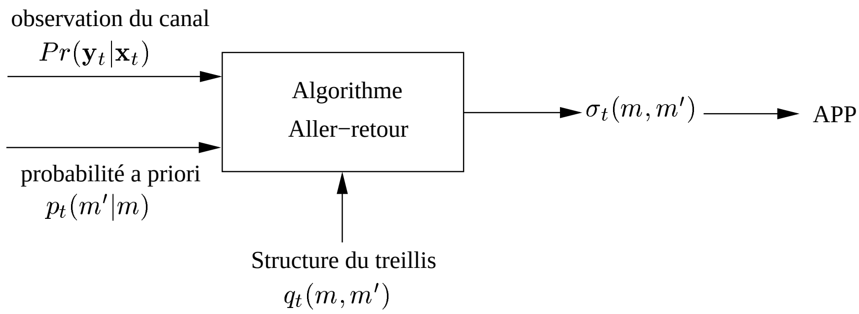


FIG. 2.14 – Représentation du décodeur MAP.

retour (*backward*) consistent respectivement à calculer $\alpha_t(m)$ and $\beta_t(m)$ à chaque instant t et pour chaque état m, à l'aide des formules de récurrence suivantes [2] :

$$\begin{aligned} \alpha_t(m') &= \sum_m \alpha_{t-1}(m)\gamma_t(m,m') , \\ \beta_t(m) &= \sum_{m'} \gamma_{t+1}(m,m')\beta_{t+1}(m') . \end{aligned}$$

L'algorithme aller-retour se déroule donc en quatre étapes :

1. **Initialisation :**

$$\alpha_0(0) = 1 \quad et \quad \alpha_0(m) = 0 \quad \forall\, m \neq 0 ,$$

$$\beta_W(0) = 1 \quad et \quad \beta_W(m) = 0 \quad \forall\, m \neq 0 .$$

2. **Procédure Aller :**
A chaque instant et pour chaque état, la densité de probabilité $\alpha_t(m)$ et la métrique $\gamma_t(m,m')$ sont calculées en utilisant les expressions suivantes :

$$\alpha_t(m') = \sum_m \gamma_t(m,m')\alpha_{t-1}(m) \quad \text{pour} \quad t = 1, \ldots, W,$$

$$\gamma_t(m,m') = q_t(m,m')\; p_t(m'|m)\; \Pr(\mathbf{y}_t|\mathbf{x}_t) .$$

3. **Procédure Retour :**
   Les densités de probabilité $\beta_t(m)$ sont calculées à partir de l'expression suivante :
   $$\beta_t(m) = \sum_{m'} \gamma_{t+1}(m,m')\beta_{t+1}(m') \quad \textit{pour} \quad t = W-1,\dots,0 \; .$$
4. **Calcul de l'APP :**
   Les densités de probabilité $\beta_t(m)$ sont associées aux densités de probabilité $\alpha_t(m)$ et aux métriques $\gamma_t(m,m')$ pour évaluer $\sigma_t(m,m')$, puis l'APP associée à chaque symbole par :
   $$APP(x_{t,i}) = \sum_{m,m'|x_{t,i}} \alpha_{t-1}(m)\; \gamma_t(m,m')\; \beta_t(m') \; . \tag{2.16}$$

Cet algorithme fournit une sortie souple, l'APP du bit[8] $c_{t,i}$, qui peut être utilisée directement pour prendre une décision sur le bit $c_{t,i}$ ou légèrement modifiée pour donner une information extrinsèque et être intégrée dans un schéma de décodage itératif.

L'information extrinsèque se déduit très facilement des équations (2.16) et (2.12) par :

$$Ext(x_{t,j}) = \frac{APP(x_{t,j})}{\Pr(y_{t,j}|x_{t,j})p_t(x_{t,j})} = \sum_{m,m'|x_{t,j}} \alpha_{t-1}(m) \left(\prod_{\substack{i=1\\ i\neq j}}^{n} p_t(x_{t,j})\Pr(y_{t,i}|x_{t,i}) \right) \beta_t(m') \; .$$

Ceci revient à calculer une seconde métrique $\gamma'_t(m,m')$ telle que :

$$\gamma'_t(m,m') = q_t(m,m') \prod_{\substack{i=1\\ i\neq j}}^{n} p_t(x_{t,i})\Pr(y_{t,i}|x_{t,i}) \; .$$

Nous obtenons alors pour l'information extrinsèque :

$$Ext(x_{t,j}) = \sum_{m,m'|x_{t,j}} \alpha_{t-1}(m)\gamma'_t(m,m')\beta_t(m') \; .$$

Notons que l'algorithme aller-retour calcule les valeurs exactes de l'APP et de l'information extrinsèque ; c'est un algorithme optimal pour le critère du maximum *a posteriori* (*critère MAP*).

Application aux codes convolutifs à treillis circulaire

L'algorithme aller-retour doit être légèrement modifié lorsqu'il est utilisé pour le décodage des codes convolutifs à treillis circulaire. Les principales étapes de l'algorithme restent identiques, exceptée la phase d'initialisation qui devient :

$$\alpha_0(m) = \beta_W(m) = 1/M \quad \forall\, m = 0,\dots,M-1 \; .$$

En pratique, le décodage est précédé d'un préambule de plusieurs bits (égal à un petit nombre de fois la mémoire du code convolutif) pour guider le processus de décodage vers un état qui semble être une bonne estimation de l'état circulant. Ce préambule est nécessaire pour le calcul des α et des β. Pour l'évaluation des α, la fin de la fenêtre des observations du canal et des probabilités *a priori* est considérée comme préambule, alors que pour le calcul des β c'est le début de la fenêtre qui est utilisée comme préambule. Ceci est illustré sur la figure 2.15.

[8] $APP(x_{t,i}) = APP(c_{t,i})$ avec $x_{t,i} = 2c_{t,i} - 1$

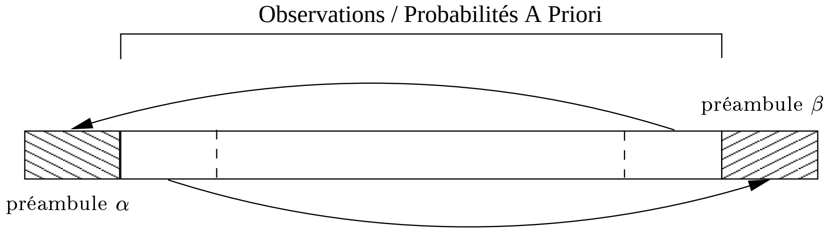


FIG. 2.15 – Préambule pour le décodage des codes convolutifs à treillis circulaire

Application aux codes en blocs linéaires

Enfin, dans le cas des codes produits ou des codes GLD blocs, nous pouvons utiliser l'algorithme aller-retour pour le décodage optimal des codes en blocs constituants. Pour pouvoir appliquer l'algorithme aller-retour, il est nécessaire d'avoir une représentation en treillis du code en blocs considéré. Plusieurs méthodes existent pour construire le treillis de ces codes [81], nous ne présentons que celle proposée par Bahl, Cocke, Jelinek & Raviv, appelée *méthode BCJR* [2]. Trois autres techniques sont détaillées en annexe B. Toutes ces méthodes mènent au treillis minimal[9] d'un code donné, ou à un treillis isomorphe.

Soit C un code en blocs linéaire binaire de longueur n et de dimension k. Son treillis minimal est noté $T = (V, E, A)$ où :

- V est l'ensemble des sommets de T : $V = V_0 \cup V_1 \cup ... \cup V_n$, où V_t est l'ensemble des sommets (ou états) à l'instant t,
- E est l'ensemble des branches du treillis : $E = E_0 \cup E_1 \cup ... \cup E_n$,
- A est l'alphabet des étiquettes sur les branches : $A = A_0 \cup A_1 \cup ... \cup A_n$ avec $A_t = GF(2)$, $\forall\, t$.

Soient $H = (h_1, h_2, ..., h_n)$ la matrice de parité de C et h_t une de ses colonnes. Le treillis BCJR du code C est construit en identifiant ses états aux syndromes partiels obtenus par rapport aux t premières colonnes de H :

$$V_t \stackrel{\triangle}{=} \{c_1 h_1 + c_2 h_2 + ... + c_t h_t : (c_1, c_2, ..., c_n) \in \mathcal{C} \text{ pour } c_{t+1}, .., c_n \in GF(2)\} \ ,$$

avec $V_0 = \{0\}$. Le syndrome d'un mot de code étant nul, nous avons également $V_n = \{0\}$. Pour calculer les syndromes V_t, il est possible de considérer la matrice G_t correspondant aux t premières colonnes de la matrice génératrice G du code C en écrivant :

$$V_t = \text{sous-espace engendré par } H_t G_t^{tr} \ ,$$

où G_t^{tr} désigne la transposée de la matrice G_t. Une branche e existe entre deux états $v \in V_{t-1}$ et $v' \in V_t$ si et seulement si un mot de code $(c_1, c_2, ..., c_n)$ de C existe tel que :

$$c_1 h_1 + ... + c_{t-1} h_{t-1} = v, \qquad c_1 h_1 + ... + c_{t-1} h_{t-1} + c_t h_t = v', \qquad a(e) = c_t \ ,$$

où $a(e)$ désigne l'étiquette de la branche e entre v et v'.

[9]Un treillis T est minimal si à chaque instant le nombre de sommets de T est inférieur au nombre de sommets de n'importe quel treillis du code. Pour les codes en blocs linéaires, le treillis minimal est unique et deux treillis minimaux d'un même code sont isomorphes.

Par exemple, considérons le code en blocs linéaire binaire $\mathcal{C}(7,3,3)$. Sa matrice de parité H et sa matrice génératrice G sont données par :

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \qquad G = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Les sommets du treillis sont alors

$$V_0 = V_7 = \{0\} \ ,$$

$$V_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \ , \quad V_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \ , \quad V_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \ ,$$

$$V_4 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \ , \quad V_5 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \ , \quad V_6 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \ .$$

Le treillis correspondant est dessiné sur la figure 2.16.

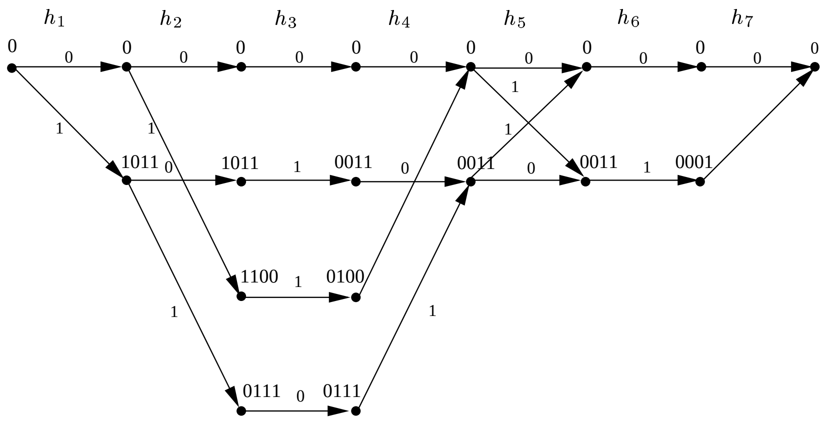


FIG. 2.16 – Treillis du code $(7,3,3)$ obtenu par la construction BCJR.

Il est maintenant possible d'appliquer au décodage des codes en blocs linéaires l'algorithme aller-retour précédemment introduit. La longueur du treillis est dans ce cas égale à la longueur du code constituant $W = n$ et il n'y a qu'un seul bit par branche, ce qui simplifie le calcul de la métrique γ. La procédure est alors identique à celle utilisée pour les codes convolutifs.

L'algorithme aller-retour est un algorithme optimal opérant sur le treillis du code en blocs. Sa complexité est donc directement liée à la complexité du treillis. Pour un code en blocs, le nombre d'états est de l'ordre de $\min(2^k, 2^{n-k})$. Par conséquent, la complexité du treillis augmente rapidement lorsque les valeurs de k ou de $n-k$ sont grandes. Il devient alors nécessaire de considérer d'autres algorithmes de décodage souple plus simples.

2.4.3 Algorithme de Fossorier-Lin pour le décodage souple des codes en blocs

L'algorithme de Fossorier-Lin [37] est un algorithme sous-optimal permettant le décodage souple des codes en blocs. Cet algorithme ne nécessite aucun décodeur algébrique contrairement à l'algorithme de Chase [19] et peut être utilisé pour le décodage souple de n'importe quel code en blocs linéaire. Chaque mot reçu est décodé selon trois étapes :
- classement des valeurs reçues par fiabilité décroissante,
- décodage à décisions dures,
- retraitement des symboles jusqu'à ce que les performances attendues soient atteintes.

Un algorithme s'appuyant sur un principe similaire a été proposé par Battail *et al.* [5].

Considérons un code en blocs linéaire binaire, $C(n,k)$, de matrice génératrice G, et un mot de code $\mathbf{c} = (c_1, c_2, ..., c_n)$ de C. Ce mot de code $\mathbf{c}$ est modulé par une BPSK en $\mathbf{x} = (x_1, x_2, ..., x_n)$ tel que $x_i = 2c_i - 1$, puis émis sur un canal gaussien caractérisé par un bruit de moyenne nulle et de variance N_0. La sortie du canal est alors notée $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. La fiabilité (ou rapport logarithmique de vraisemblance) LV_i du symbole reçu y_i est définie par :

$$LV_i = \log \frac{\Pr(y_i|x_i = 1)}{\Pr(y_i|x_i = -1)} = \frac{|y_i|}{2N_0} \; .$$

La valeur de N_0 étant constante, nous utilisons $|y_i|$ comme valeur de confiance du symbole y_i.

Détaillons maintenant les trois étapes de décodage de l'algorithme :

1. **Classement des valeurs de fiabilité :**

   La première étape du décodage consiste à classer les symboles de la séquence reçue $\mathbf{y}$ en fonction de leur fiabilité $|y_i|$, $i = 1, \ldots, n$ par ordre décroissant. La probabilité d'avoir $|y_i| = |y_j|$ pour $i \neq j$ est négligeable, du fait de l'indépendance des symboles reçus. La séquence résultante est notée $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ et est telle que $|z_1| > |z_2| > ... > |z_n|$. Le vecteur $\mathbf{z}$ est donc une version permutée de $\mathbf{y}$:

   $$\mathbf{z} = \lambda_1 \; \mathbf{y} \; .$$

   Ceci revient à appliquer la permutation λ_1 sur les colonnes de la matrice génératrice G pour construire la matrice G' telle que :

   $$G' = \lambda_1 G \; .$$

   Le code C' généré par la matrice G' est naturellement équivalent au code C.
   Soit g'_i la i-ème colonne de G' ; g'_i correspond à la i-ème composante de $\mathbf{z}$, par conséquent $|z_i|$ est la valeur de fiabilité associée à la colonne g'_i. En commençant par les premières colonnes de G', nous recherchons k colonnes indépendantes de la matrice, ayant les valeurs de confiance les plus grandes. Ces k colonnes sont utilisées pour former les k premières colonnes d'une nouvelle matrice génératrice G'', de dimension $k \times n$. Les $n - k$ dernières colonnes

sont classées par ordre décroissant de leur fiabilité. Ceci définit une seconde permutation λ_2, telle que :

$$G'' = \lambda_2 G' = \lambda_2[\lambda_1 G] \ .$$

Les composantes de $\mathbf{z}$ sont réordonnées[10] selon la permutation λ_2 et forment le vecteur $\mathbf{u} = (u_1, u_2, \ldots, u_k, u_{k+1}, \ldots, u_n)$ avec $|u_1| > |u_2| > ... > |u_k|$ et $|u_{k+1}| > |u_{k+2}| > ... > |u_n|$:

$$\mathbf{u} = \lambda_2 \mathbf{z} = \lambda_2[\lambda_1 \mathbf{y}] \ .$$

Les k premières composantes de $\mathbf{u}$ sont appelées symboles indépendants les plus fiables de $\mathbf{u}$ (notés symboles MRI pour *Most Reliable Independant*).
La matrice G'' est ensuite mise sous forme systématique $G_1 = [I_k|P]$, où I_k est la matrice identité de dimension k, et P une matrice de parité. La matrice G_1 est la matrice génératrice d'un code C_1 équivalent aux codes C' et C.

2. **Décodage à décisions fermes**

La seconde étape de l'algorithme consiste à prendre une décision sur les k symboles MRI de $\mathbf{u}$, pour obtenir les k premières composantes d'un vecteur $\mathbf{a}$ telles que :

$$a_i = \begin{cases} 0 & \text{si } u_i < 0 \\ 1 & \text{sinon} \end{cases} \ .$$

La séquence des k bits ainsi obtenue est utilisée comme séquence d'information pour construire le mot de code $\mathbf{a} = (a_1, \ldots, a_k, a_{k+1}, \ldots, a_n)$ du code systématique C_1 :

$$a_{k+\ell} = \sum_{1 \le i \le k} a_i p_{i,\ell} \qquad \text{pour } 1 \le \ell \le n-k \ .$$

Les $p_{i,\ell}$ sont les composantes de la matrice de parité P de G_1. Le mot de code $\mathbf{a}$ est considéré comme le mot issu de la décision ferme prise sur la séquence $\mathbf{u}$. Le mot de code estimé $\hat{\mathbf{c}}$ du code C initial peut être retrouvé de manière simple en appliquant la permutation inverse $[\lambda_2(\lambda_1)]^{-1} = \lambda_1^{-1}(\lambda_2^{-1})$.
Cette technique, consistant à opérer un décodage ferme à partir des valeurs de fiabilité réordonnées, est proche d'un décodage optimal. En fait, au lieu de minimiser la distance euclidienne entre les mots, c'est le nombre d'erreurs sur les bits d'information qui est minimisé.

3. **Retraitement des symboles**

La dernière étape de décodage consiste à retraiter le mot de code $\mathbf{a}$ obtenu par décodage ferme, jusqu'à l'obtention d'une probabilité d'erreur quasi-optimale. Pour $1 \le m \le k$, le traitement d'ordre m de $\mathbf{a}$ se traduit par :

- Pour $0 \le \ell \le m$, on considère toutes les combinaisons possibles de ℓ bits choisis parmi les k symboles MRI de $\mathbf{a}$. Ces ℓ bits sont complémentés.
- Pour chaque combinaison, on reconstruit le mot de code $\mathbf{a}^\star$ à l'aide de la matrice génératrice G_1 et on détermine la séquence modulée correspondante $\mathbf{x}^\star$.

[10]Notons que $|u_k|$ n'est pas nécessairement plus grand que $|u_{k+1}|$

- La distance euclidienne $d_E(\mathbf{u}, \mathbf{x}^\star)$ entre la séquence $\mathbf{u}$ et la séquence $\mathbf{x}^\star$ est alors évaluée. Le mot de code $\mathbf{a}^\star$ pour lequel $\mathbf{x}^\star$ est le plus proche de $\mathbf{u}$ est stocké.
- Lorsque les $\sum_{\ell=0}^{m} C_k^\ell$ mots de code ont été testés, le traitement d'ordre m est terminé, et le dernier mot de code stocké est le mot de code final décodé.

Ces étapes sont représentées sous la forme d'un organigramme sur la figure 2.17. L'ordre k est équivalent au décodage à maximum de vraisemblance, et nécessite 2^k retraitements. En règle générale l'ordre nécessaire dépend de la longueur n et du rendement R du code [37]. Pour des longueurs inférieures à 64, l'ordre 2 permet de se rapprocher des performances optimales, tandis que l'ordre 1 réalise un bon compromis entre complexité et performances. Pour des codes plus longs, des ordres supérieurs sont nécessaires, mais ceci rend l'algorithme de décodage très complexe. Par ailleurs, des ordres trop importants n'apportent pas un gain substantiel en termes de performance. Notons qu'une autre version de cet algorithme est fondée sur l'espace dual du code [38] ; les deux méthodes sont équivalentes et donnent des résultats identiques.

L'algorithme de Fossorier-Lin est un algorithme de décodage souple des codes en blocs, c'est-à-dire qu'il utilise les informations de fiabilité disponibles en son entrée afin d'améliorer le décodage. Cependant dans sa forme initiale, cet algorithme ne produit aucune information souple. Afin de l'intégrer dans un processus de décodage itératif, nous proposons de générer des valeurs approchées de l'APP et de l'information extrinsèque pour chaque bit codé.

Dans les équations (2.10) et (2.12), nous considérons pour calculer l'APP et l'information extrinsèque d'un bit une somme sur l'ensemble des mots du code C tel que la i-ème composante de ces mots soit égale à c_i. Or, prendre en compte l'ensemble des mots de code de C est équivalent à un décodage exhaustif, ce qui est beaucoup trop complexe. Nous allons donc restreindre cet ensemble aux mots de code les plus probables, *i.e.*, aux mots les plus proches du mot reçu. Si cet ensemble est correctement choisi, les pertes sont négligeables, voire inexistantes, par rapport au calcul exact de l'APP et de l'information extrinsèque. Nous avons choisi pour cet ensemble restreint C' l'ensemble des mots de code délivrés par l'algorithme de Fossorier-Lin après chaque retraitement du mot $\mathbf{a}$. Cet ensemble est alors de cardinal $\sum_{\ell=0}^{m} C_k^\ell$, si l'on considère un retraitement d'ordre m et peut être réduit en considérant les critères d'arrêt proposés dans [37].

Une autre possibilité est de ne considérer que le mot de code le plus proche de la séquence reçue ayant pour i-ème composante c_i et le mot de code ayant pour i-ème composante $\bar{c}_i$, complément à 2 du bit c_i, afin de normaliser l'APP ($APP(c_i) + APP(\bar{c}_i) = 1$). Contrairement à l'algorithme de Chase, où il se peut que ce dernier mot de code ne soit pas dans l'ensemble C' et où l'utilisation de facteurs d'oubli est alors nécessaire [64][65], l'algorithme de Fossorier-Lin fournit suffisamment de mots de code pour garantir la présence de ces deux mots de code dans l'ensemble C' [39].

2.4.4 Interprétation graphique du décodage itératif

Considérons par exemple un code composite construit à partir de deux codes constituants et utilisant comme algorithme de décodage l'algorithme aller-retour. Si l'on représente chaque code constituant par un graphe bayésien, l'algorithme aller-retour est équivalent à une propagation "aller" des probabilités le long de la chaîne du code constituant, et une propagation "retour" des

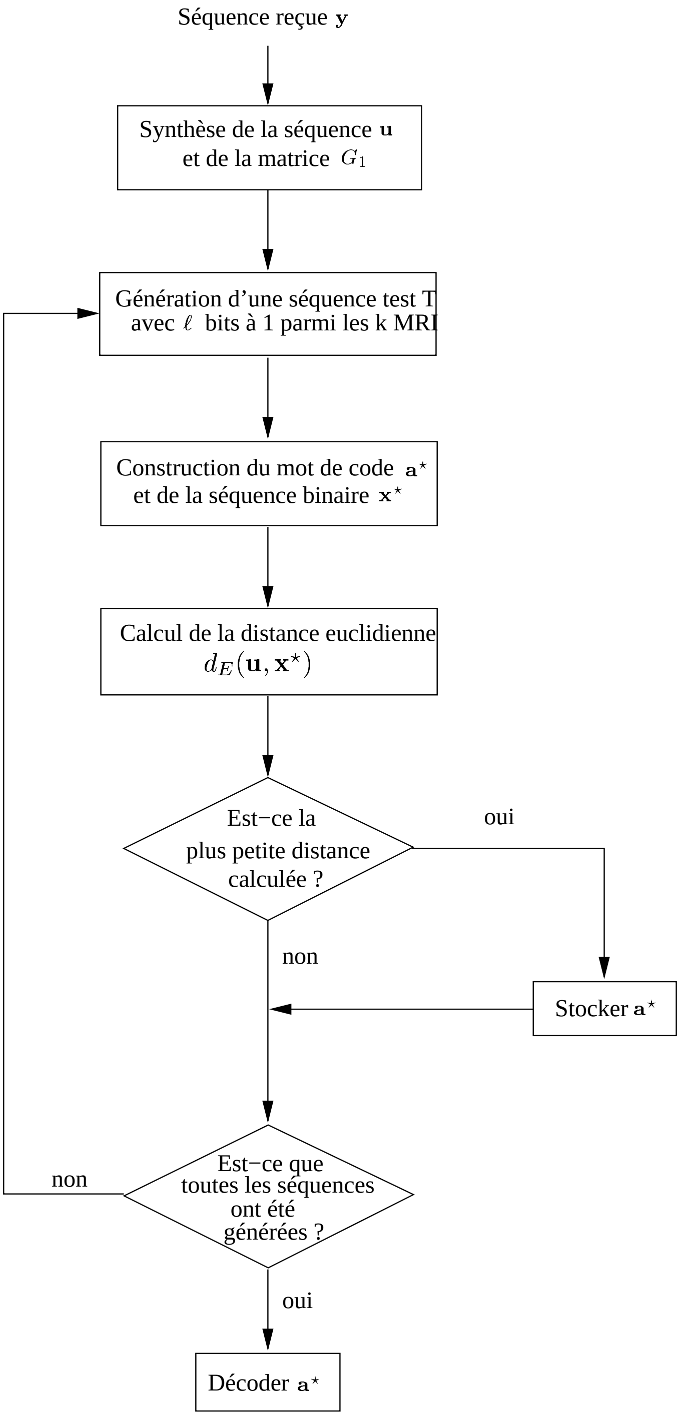


Fig. 2.17 – Organigramme de l'algorithme de Fossorier-Lin

probabilités le long de cette même chaîne (par exemple, voir la figure 2.10). Ceci garantit que les observations sont distribuées sur l'ensemble des sommets et permet de calculer de manière exacte l'APP.

Le décodage itératif SISO consiste à appliquer l'algorithme aller-retour tout d'abord sur le code supérieur, ce qui permet la génération d'informations extrinsèques sur les bits codés. Ces informations sont transmises à la seconde chaîne par l'intermédiaire des arêtes caractérisant l'entrelaceur. Puis le même algorithme est appliqué sur le code inférieur en prenant en compte les informations *a priori* fournies par le premier décodeur, ainsi que les observations du canal. Le décodeur inférieur calcule à son tour des informations extrinsèques qu'il fournit au décodeur supérieur par le biais des arêtes non orientées entre les deux chaînes, etc.

Notons que la valeur de l'APP finale obtenue à la suite des itérations n'est pas une valeur exacte de l'APP, mais simplement une valeur approchée, du fait de la présence de cycles dans le graphe du code composite. En effet, [84][52] ont montré que l'algorithme de décodage itératif ne mène au calcul de l'APP exacte que s'il n'y a pas de cycle dans le graphe du code considéré. Dans le cas contraire, l'algorithme peut ne jamais converger vers la valeur exacte de l'APP. Heureusement en pratique, nous verrons dans la section suivante que malgré la présence de cycles dans le graphe des codes composites, les performances obtenues avec de tels codes sont particulièrement remarquables. Pour de plus amples détails sur l'influence des cycles dans le graphe des codes composites, le lecteur est invité à se référer à [62][84][40].

2.5 Résultats de simulation

Les performances des différents codes composites évoqués dans ce chapitre ont été largement présentées dans la littérature : [11][8][9][48] pour les turbo-codes parallèle, [6][7] pour les turbo-codes série, [58][56] pour les codes LDPC, [14][62][55] pour les GLD blocs, [64][65] pour les codes produits décodés avec l'algorithme de Chase. Cette liste n'est bien sûr pas exhaustive.

Aussi nous ne nous intéressons dans cette section qu'aux performances des schémas de codage et de décodage que nous avons introduits à proprement parler : le décodage itératif des codes produits à l'aide de l'algorithme de Fossorier-Lin, et les codes produits construits à partir de codes convolutifs à treillis circulaire. Les performances atteintes en considérant l'algorithme sous-optimal de Fossorier-Lin sont comparées aux performances obtenues avec l'algorithme aller-retour optimal mais plus complexe. Nous observons par ailleurs comment se positionnent les codes produits ainsi décodés par rapport aux limites présentées dans le chapitre 1. Enfin, pour de grandes longueurs de code, nous introduisons les performances atteintes avec des codes produits construits à partir de codes convolutifs à treillis circulaire.

Pour l'ensemble des simulations présentées ci-après, la modulation employée est une modulation BPSK et le canal est AWGN.

La figure 2.18 précise les performances obtenues lorsque l'on considère un code produit ($J = 2$) construit à partir du code BCH étendu BCH($n = 32, k = 21, d_{H\,\min} = 6$) et décodé à l'aide de l'algorithme de Fossorier-Lin. L'ordre 1 et l'ordre 2 de retraitement sont considérés, et le décodage itératif du code produit se déroule en 4 ou en 8 itérations. Nous notons par la suite ce code CP$(32, 21, 6)^2$.
A chaque retraitement, un mot de code est généré. Au total, l'ensemble restreint C' contient 22 mots de code pour l'ordre 1 et 232 mots de code pour l'ordre 2. La figure 2.18 montre tout d'abord que peu d'itérations sont nécessaires avec un code produit : en effet pour l'ordre 1 les

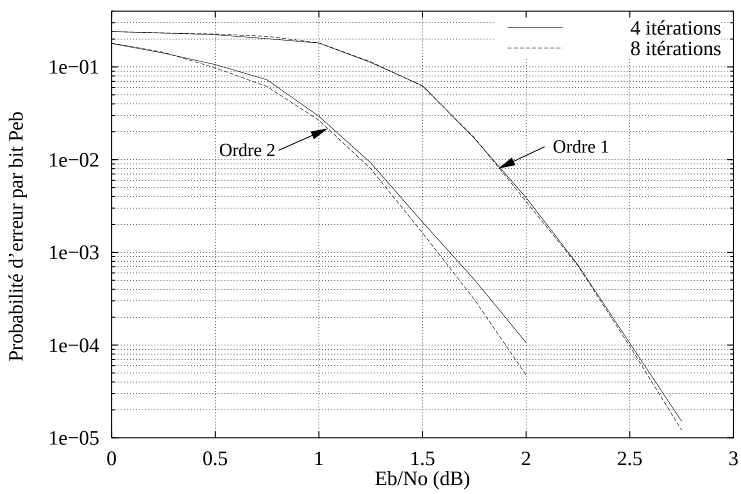


Fig. 2.18 – Performances de l'algorithme de Fossorier-Lin, ordre 1 et ordre 2

performances sont identiques selon que 4 ou 8 itérations sont utilisées pour le décodage, tandis que pour l'ordre 2, doubler le nombre d'itérations apporte un gain de 0.1 dB, ce qui est négligeable. Par ailleurs, passer de l'ordre 1 à l'ordre 2 et accroître ainsi la complexité du décodage[11] pour compléter l'ensemble C', se traduit par un gain de 0.6 dB pour atteindre une probabilité d'erreur par bit de 10^{-4}.

La figure 2.19 compare les performances du code produit $\mathrm{CP}(32,21,6)^2$ décodé par l'algorithme de Fossorier-Lin d'une part et par l'algorithme aller-retour d'autre part. L'ordre de retraitement est égal à 2 pour l'algorithme de Fossorier-Lin. Le treillis du code BCH comporte $2^{11} = 2048$ états, pour l'algorithme aller-retour.
Comme pour l'algorithme de Fossorier-Lin, peu d'itérations sont nécessaires au décodage du code produit pour converger vers sa décision finale. Les deux algorithmes ont des performances très similaires (0.2 dB de gain pour l'algorithme aller-retour), voire identiques lorsque la probabilité d'erreur est inférieure à 10^{-4}. La complexité de l'algorithme de Fossorier-Lin, même lorsque l'ordre de retraitement est 2, est cependant plus faible que la complexité de l'algorithme aller-retour appliqué à un treillis comportant 2048 états. Pyndiah *et al.* [64][65] ont obtenu des performances similaires en utilisant l'algorithme de Chase [19]. L'avantage de l'algorithme de Fossorier-Lin sur l'algorithme de Chase est qu'il ne nécessite aucun décodeur algébrique et peut être ainsi appliqué à tout code linéaire. Par ailleurs, il n'est pas nécessaire d'introduire des facteurs d'oubli pour évaluer correctement l'APP et l'information extrinsèque d'un bit codé.

La figure 2.20 permet de visualiser l'écart entre les performances du code produit $\mathrm{CP}(32,21,6)^2$ et les performances d'un code optimal de même longueur ($N = 1024$) et de même rendement ($R = 0.43$) sur canal gaussien (*cf.* chapitre 1). La distance minimale normalisée donnée par la borne de Gilbert-Varshamov est $\delta = 0.0883$ pour un code de rendement 0.43. Le code produit est décodé à l'aide de l'algorithme de Fossorier-Lin d'ordre 2, et de 8 itérations.
Pour une probabilité d'erreur de 10^{-5}, le code produit $\mathrm{CP}(32,21,6)^2$ se trouve à 1.3 dB environ de la probabilité d'erreur optimale sur canal AWGN pour $N = 1024$. Cette valeur est similaire à

[11] la complexité est multipliée par un facteur 10 environ

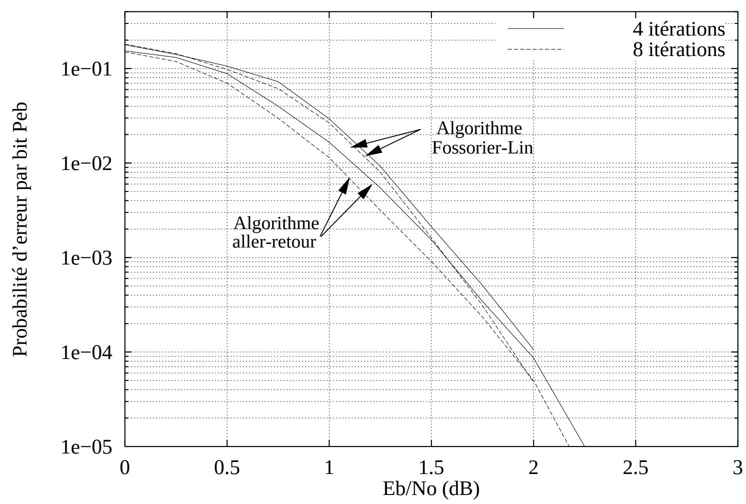


FIG. 2.19 – Comparaison de l'algorithme de Fossorier-Lin avec l'algorithme aller-retour

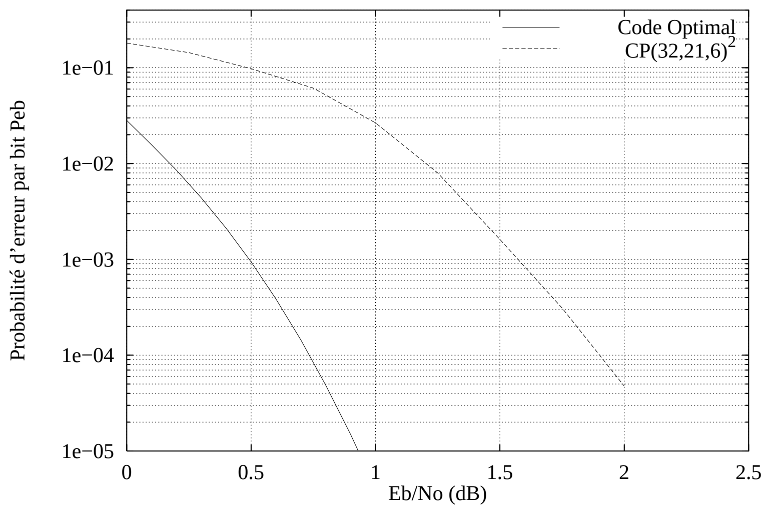


FIG. 2.20 – Position des codes produits par rapport aux codes optimaux

l'écart séparant les turbo-codes parallèle de cette même limite pour un rendement et une longueur équivalents.

Lorsque de plus grandes longueurs de code sont nécessaires, par exemple $N = 16384$ pour un rendement de code $R \approx \frac{1}{2}$, deux démarches sont possibles avec les codes produits :

- on considère des codes constituants de longueurs plus grandes, mais de rendements identiques, c'est-à-dire égaux à $\frac{2}{3}$: pour $J = 2$, le code étendu $(128, 86)$ donne une longueur $N = 16384$ et un rendement total $R = 0.45$
- on considère des codes constituants de longueurs plus faibles et de rendements plus grands, mais on augmente le nombre de niveaux dans le code produit, *i.e.*, $J > 2$

La première méthode augmente la complexité de décodage de manière non négligeable : pour un tel code il est bien sûr hors de question de considérer l'algorithme aller-retour (le treillis du code aurait alors près de 2^{42} états ! [12]). L'utilisation de l'algorithme de Fossorier-Lin impose quant à elle d'augmenter l'ordre de retraitement [37]. En ce qui concerne la seconde méthode, les codes constituants doivent avoir des rendements relativement grands pour obtenir le rendement total désiré. La différence entre le rendement de chaque code constituant et le rendement du code produit est pénalisante au niveau du décodage élémentaire des codes constituants. Prenons par exemple un code produit tridimensionnel de longueur $N = 11088$ et de rendement $R = 0.466$; ce code produit peut être obtenu à l'aide des codes en blocs $C^1(N_1 = 24, K_1 = 19)$, $C^2(N_2 = 22, K_2 = 17)$, et $C^3(N_3 = 21, K_3 = 16)$. Si le rapport signal-à-bruit du système est $\frac{E_b}{N_0} = 2.0$ dB, alors, du fait de la différence de rendement, le code C^1 de rendement R_1 voit en son entrée l'équivalent d'un rapport signal-à-bruit $\left(\frac{E_b}{N_0}\right)' = \left(2.0 - 10\log_{10}\frac{R_1}{R}\right) \approx -0.3$ dB.

Nous proposons donc d'utiliser des codes convolutifs à treillis circulaire pour construire des codes produits de grandes dimensions à deux niveaux ($J = 2$). Chaque code convolutif garde cependant une longueur raisonnable pour permettre un décodage rapide des lignes et des colonnes du code produit. Chaque ligne et chaque colonne peuvent être décodées indépendamment les unes des autres, en parallèle, ce qui permet de réduire le délai introduit par le décodage itératif. La figure 2.21 compare les performances obtenues avec le code produit tridimensionnel construit à partir des codes C^1, C^2, et C^3, et les résultats donnés par un code produit bidimensionnel construit à partir d'un code TB de rendement $\frac{2}{3}$, de polynômes générateurs $(23, 31, 35)$ et de longueur $n = 105$. Le rendement de ce code produit est $R = 0.444$. Tous les codes constituants, qu'ils soient en blocs ou convolutifs, sont décodés à l'aide de l'algorithme aller-retour. 4 ou 8 itérations sont utilisées pour le décodage itératif. Le gain atteint est de l'ordre de 1 dB pour une probabilité d'erreur $Pe_b = 10^{-4}$.

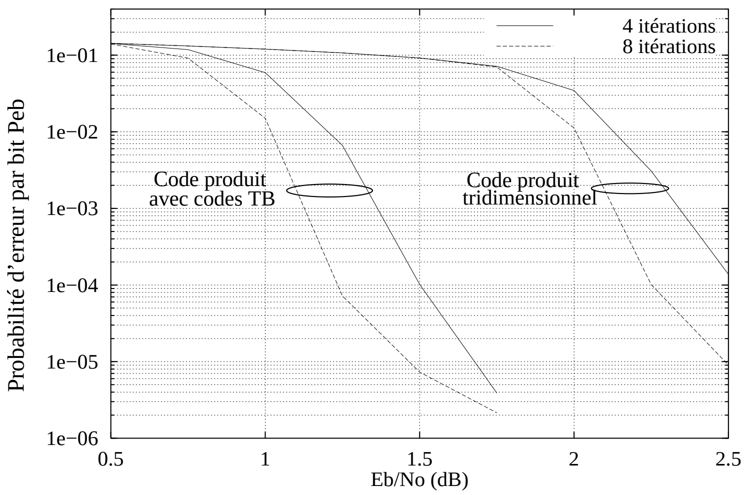


FIG. 2.21 – Performances des codes produits construits à partir de codes convolutifs à treillis circulaire

[12]même si des techniques de simplification sont utilisées pour la construction du treillis, la complexité de celui-ci reste prohibitive.

2.6 Conclusions

Nous avons présenté dans ce chapitre les différentes structures utilisées pour construire un code composite. Un tel code repose sur l'association de plusieurs codes constituants plus simples, possédant chacun un décodeur élémentaire à entrée et à sortie souples. Le décodage d'un code composite se fait par échange d'informations probabilistes entre les décodeurs élémentaires, au cours d'un processus itératif. Ainsi, les turbo-codes, les codes LDPC, les codes de Tanner, ou les codes produits appartiennent à la famille des codes composites.

Les performances analytiques de ces codes composites, lorsque le décodage optimise le critère du maximum de vraisemblance, ont été brièvement rappelées. Par ailleurs, les principes du décodage itératif ont été exposés, et dans ce cadre un nouvel algorithme de décodage a été proposé pour les codes produits. Cet algorithme s'appuie sur l'algorithme de Fossorier-Lin et permet de décoder de manière itérative les codes produits. Contrairement à l'algorithme de Chase, il possède l'avantage de pouvoir être appliqué à un code en blocs linéaire quelconque. Cet algorithme a montré de très bonnes performances moyennant une complexité réduite, lorsque l'ordre de retraitement est faible. Enfin, un schéma de code produit construit à partir de codes convolutifs à treillis circulaire a été introduit pour améliorer les performances des codes produits de grandes longueurs, tout en gardant un décodage parallèle des lignes et des colonnes du code produit.

Notons que dans ce chapitre, seuls des codes composites construits à partir de graphes bipartites réguliers ont été considérés. Il est possible pour améliorer les performances de ces schémas d'introduire de l'irrégularité dans leur graphe. Luby *et al.* [56] ou encore Chung *et al.* [21] ont ainsi proposé des codes LDPC irréguliers permettant de se rapprocher davantage des limites fixées par la capacité.

Chapitre 3

Codes GLD construits à partir de codes convolutifs *

Les codes LDPC [42] et les codes de Tanner [76], longtemps oubliés, ont été redécouverts depuis l'introduction des turbo-codes en 1993 par Berrou *et al.* [11]. De nouveaux schémas exploitant les principes fondamentaux de ces codes ont été proposés. Ainsi, le remplacement des codes de parité des LDPC par des codes en blocs plus complexes, tels que des codes de Hamming ou des codes BCH, a été étudié par [62] et [55]. L'analyse des performances ML de ces codes a montré que, comme les codes LDPC de paramètres (N, j, ℓ) avec $j \geq 3$, les codes GLD blocs sont asymptotiquement bons même lorsque deux sous-matrices seulement sont considérées.

Nous proposons dans ce chapitre une généralisation différente des codes LDPC. Les codes de parité sont maintenant remplacés par les équations de parité d'un code convolutif. Nous appelons code *GLD convolutif* le code ainsi construit à partir de plusieurs codes convolutifs. Chaque sous-matrice de la matrice de parité du code GLD convolutif correspond ainsi à la matrice de parité d'un code convolutif.

Le chapitre est organisé comme suit. Nous présentons tout d'abord la définition et la structure des codes GLD convolutifs et construisons leur matrice de parité à partir de la matrice de parité du code convolutif constituant. Puis nous envisageons la représentation graphique d'un code GLD convolutif. Nous montrons alors comment les turbo-codes, issus d'une concaténation série ou parallèle, peuvent être interprétés comme des cas particuliers de codes GLD convolutifs. Nous analysons ensuite les performances de ces codes lorsqu'un décodage à maximum de vraisemblance est mis en œuvre. Il s'agit de performances moyennes obtenues en considérant l'ensemble de toutes les permutations aléatoires possibles entre les codes constituants. Cette étude permet de comparer ces codes aux codes composites introduits au chapitre précédent, notamment en termes de distance minimale asymptotique et de gain d'entrelacement. Puis nous nous intéressons au décodage itératif des codes GLD convolutifs. En effet, comme pour les autres codes composites, le décodage à maximum de vraisemblance des codes GLD convolutifs est difficile à réaliser en pratique. Par conséquent, le décodage itératif est envisagé pour approcher les performances ML.

*Le contenu de ce chapitre a été présenté en partie au *Workshop on Coding and Cryptography (WCC)*, Paris, France, 11-14 janvier 1999.

Or ce décodage itératif est limité par la présence de cycles dans le graphe du code considéré. Nous proposons alors un entrelacement particulier afin de diminuer le nombre des cycles et d'améliorer le décodage itératif. Enfin nous présentons quelques résultats numériques obtenus par simulation des codes GLD.

3.1 Définition et structure d'un code GLD convolutif

Soit $C^1(N, K_1)$ un code en blocs linéaire obtenu à partir d'un code convolutif $C_c^1(n, k)$ par fermeture de son treillis[1]. Le code C_c^1 peut être systématique ou non, récursif ou non.
Soient $C^2, \ldots, C^J$, $J-1$ codes en blocs linéaires, tels que $C^i = \pi_i(C^1)$, $i = 2, \ldots, J$, où π_i, $i = 2, \ldots, J$, est une permutation aléatoire sur les bits, appliquée au code en blocs initial C^1. Nous avons alors la définition suivante :

Définition 3.1.1 *(Code à faible densité généralisé construit à partir de codes constituants convolutifs)*
Un code à faible densité généralisé, construit à partir de J codes constituants convolutifs identiques, est un code en blocs linéaire $\mathcal{C}(N, K)$ défini comme l'intersection des codes C^i, $i = 1, \ldots, J$. Un tel code est appelé code GLD convolutif.

Notons que cette définition peut être élargie à des codes constituants non identiques. Par la suite, lorsque cela n'est pas spécifié, nous nous restreignons au cas où tous les codes constituants sont identiques.

Comme pour les codes LDPC, la matrice de parité d'un code GLD convolutif est obtenue à partir des matrices de parité de ses codes constituants, *i.e.*, de la matrice de parité H^1 du code C^1, puis de matrices résultant de l'application de permutations aléatoires sur les colonnes de H^1. Il est donc nécessaire de déterminer la matrice de parité d'un code convolutif.

3.1.1 Matrice de parité d'un code convolutif

Nous envisageons deux types de codes convolutifs : récursifs systématiques ou non récursifs non systématiques. La matrice de parité d'un code est constituée d'équations de parité, c'est-à-dire qu'elle est élaborée à partir des syndromes du code convolutif. Nous allons étudier comment trouver ces syndromes dans les deux cas envisagés. Par ailleurs, un code convolutif étant infini par définition, sa matrice de parité est elle aussi de dimension infinie. Nous analysons donc également comment prendre en compte la fermeture du treillis du code convolutif dans sa matrice de parité.

Matrice de parité d'un code récursif systématique

Pour des raisons de simplicité, nous considérons uniquement des codes convolutifs récursifs systématiques de rendement $r = \frac{k}{k+1}$, représentés sous forme canonique : dans ce cas, un seul

[1]Par abus de langage, nous appellerons parfois code convolutif le code C^1.

registre à décalage est nécessaire pour le codeur. Le codeur RSC lit k bits d'information et génère un bit de parité. Le nombre de bits à la sortie du codeur convolutif est donc $n = k+1$. La longueur de contrainte du code est notée $L = \nu + 1$, où ν est la mémoire du code. Le treillis du code RSC a alors 2^ν états : ν bits de fermeture sont nécessaires pour ramener le treillis à l'état 0, ce qui correspond à $\lceil \frac{\nu}{k} \rceil$ branches dans le treillis. Le code convolutif est défini par n polynômes générateurs, $g_0(x), \ldots, g_{n-1}(x)$ tels que :

$$g_i(x) = \sum_{j=0}^{L-1} g_i^j x^j \quad \text{pour} \quad i = 0, \ldots, n-1 \ ,$$

$g_0(x)$ est le polynôme de rétroaction du code récursif.

Les n séquences en sortie du codeur, notées $s_i(x)$, $i = 0, \ldots, n-1$, et les k séquences en entrée du codeur, notées $e_i(x)$, $i = 1, \ldots, k$, sont liées par les équations suivantes :

$$s_i(x) = e_i(x) \ \ \text{pour } i = 1, \ldots, k \qquad \text{et} \qquad s_0(x) = \sum_{i=1}^{k} \frac{g_i(x)}{g_0(x)} e_i(x) \ . \tag{3.1}$$

La figure 3.1 donne la représentation canonique d'un code récursif systématique de rendement 3/4, de longueur de contrainte $L = 4$ et de polynômes générateurs exprimés en octal $(15, 16, 13, 17)$. Nous notons C_{ex}^{RSC} ce code.

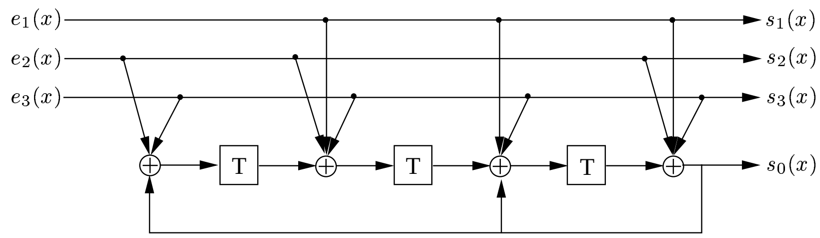


FIG. 3.1 – Codeur du code RSC C_{ex}^{RSC}, de rendement $r = 3/4$, $(15, 16, 13, 17)$.

Les équations de parité du code RSC sont définies par le syndrome obtenu à l'aide des équations (3.1) :

$$g_0(x)s_0(x) + g_1(x)s_1(x) + \ldots + g_k s_k(x) = 0 \qquad \text{avec } k = n-1 \ . \tag{3.2}$$

Ce syndrome permet de construire la matrice de parité semi-infinie $H_{RSC\infty}$ d'un code convolutif systématique récursif. En effet, l'équation (3.2) s'écrit également :

$$\sum_{j=0}^{L-1} \left(\sum_{i=0}^{n-1} g_i^j s_i(x) \right) x^j = 0 \ . \tag{3.3}$$

Soit m l'indice de colonne de la matrice de parité $H_{RSC\infty}$. Si[2] $m[n] = 0, 1, \ldots, k-1$ alors la colonne m est associée à un bit d'information ; sinon, elle est associée à un bit de parité. Chaque

[2] $m[n]$: m modulo n

équation de parité de la matrice $H_{RSC\infty}$ est obtenue en décalant l'équation (3.3) de n bits vers la droite. La longueur d'un syndrome étant nL, les équations de parité de cette matrice ne sont par conséquent pas nécessairement disjointes : elles partagent en effet $n(L-1)$ bits. De plus, un code convolutif étant de longueur infinie, sa matrice de parité $H_{RSC\infty}$ est elle aussi de dimension infinie :

$$H_{RSC\infty} = \begin{bmatrix} g_{n-1}^{0} & \cdots & g_0^0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{n-1}^{1} & \cdots & g_0^1 & g_{n-1}^{0} & \cdots & g_0^0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{n-1}^{2} & \cdots & g_0^2 & g_{n-1}^{1} & \cdots & g_0^1 & g_{n-1}^{0} & \cdots & g_0^0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{n-1}^{L-1} & \cdots & g_0^{L-1} & g_{n-1}^{L-2} & \cdots & g_0^{L-2} & \cdots & g_{n-1}^{0} & \cdots & g_0^0 & 0 & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & g_{n-1}^{L-1} & \cdots & g_0^{L-1} & g_{n-1}^{L-2} & \cdots & g_0^{L-2} & \cdots & g_{n-1}^{0} & \cdots & g_0^0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} .$$

Si l'on convertit le code convolutif représenté par la matrice de parité $H_{RSC\infty}$ en un code en blocs de longueur N_{RSC} et de dimension K_{RSC} par fermeture du treillis (*i.e.*, retour à l'état 0), $\lceil \frac{\nu}{k} \rceil$ branches de fermeture sont nécessaires. Cette fermeture correspond aux $\lceil \frac{\nu}{k} \rceil \times n$ dernières colonnes et aux $\left(\lceil \frac{\nu}{k} \rceil + \nu\right)$ dernières lignes de la matrice de parité H_{RSC} du code en blocs résultant. Les ν dernières lignes représentent les équations de fermeture du treillis : le registre est vidé, il n'y a plus de nouveaux bits codés émis. Ces lignes sont obtenues en décalant le syndrome de n bits jusqu'à ce qu'il n'y ait plus que des 0 à décaler. Notons, que si $\lceil \frac{\nu}{k} \rceil \neq \frac{\nu}{k}$, *i.e.*, si ν n'est pas un multiple de k, alors il faut ajouter $\lceil \frac{\nu}{k} \rceil \times k - \nu$ bits à 0 dans la trame d'information pour rester dans l'état 0. Dans ce cas, en effet, il y a plus de bits sur les branches de fermeture que de bits nécessaires réellement pour fermer le treillis, *i.e.*, $\lceil \frac{\nu}{k} \rceil \times k$ bits au lieu de ν. Par conséquent une fois dans l'état 0, il est nécessaire de fournir en entrée du codeur des bits à 0 pour ne plus changer d'état.

La dimension du code est donnée par :

$$K_{RSC} = \left(\frac{N_{RSC}}{n} - \left\lceil \frac{\nu}{k} \right\rceil \right) \times k \ ,$$

car les bits de fermeture et les bits supplémentaires à 0 qui doivent être ajoutés à la trame d'information ne font pas partie des bits utiles. Le rendement du code en blocs obtenu est donc :

$$R_{RSC} = \frac{K_{RSC}}{N_{RSC}} = \frac{\left(\frac{N_{RSC}}{n} - \lceil \frac{\nu}{k} \rceil \right) \times k}{N_{RSC}} \ .$$

R_{RSC} est proche du rendement $r = \frac{k}{n}$ du code convolutif initial si le nombre de bits de fermeture est négligeable devant le nombre total de bits codés.

Pour illustrer la construction de la matrice de parité H_{RSC}, nous allons considérer la matrice de parité H_{ex}^{RSC} du code récemment introduit C_{ex}^{RSC}. Les polynômes générateurs du code C_{ex}^{RSC}

s'écrivent :

$$
\begin{aligned}
g_0(x) &= 1 + x + x^3 \ , \\
g_1(x) &= 1 + x + x^2 \ , \\
g_2(x) &= 1 + x^2 + x^3 \ , \\
g_3(x) &= 1 + x + x^2 + x^3 \ .
\end{aligned}
$$

L'équation du syndrome devient donc :

$$
\begin{aligned}
(s_0(x) + s_1(x) + s_2(x) + s_3(x)) &\ + \\
x(s_0(x) + s_1(x) + s_3(x)) &\ + \\
x^2(s_1(x) + s_2(x) + s_3(x)) &\ + \\
x^3(s_0(x) + s_2(x) + s_3(x)) &= 0 \ .
\end{aligned}
$$

Supposons que C_{ex}^{RSC} est transformé en un code en blocs de longueur $N_{RSC} = 40$. Une branche est nécessaire pour fermer le treillis, soit 4 bits codés. La matrice de parité de C_{ex}^{RSC} est représentée sur la figure 3.2.

$N_{RSC} = 40$ — fermeture du treillis

$$
H_{ex}^{RSC} = \left[\begin{array}{c:c}
1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0 \\
1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0 \\
1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0 \\
1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0 \\
0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0 & 0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1 & 0\,0\,0\,0 \\
\hdashline
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1 & 1\,1\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0 & 1\,0\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1 & 1\,1\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,1\,0\,1
\end{array}\right]
$$

fermeture du treillis

syndrome de longueur nL

FIG. 3.2 – Représentation de la matrice de parité du code RSC C_{ex}^{RSC} de longueur 40 bits

La dimension de ce code est $K_{RSC} = 27$, par conséquent, son rendement est égal à 0.675. Cette valeur est faible par rapport au rendement initial du code convolutif qui est égal à $3/4$; $1/10$-ème des bits codés sont en effet des bits de fermeture.

Matrice de parité d'un code NRNS

Considérons maintenant un code convolutif non récursif non systématique (noté code *NRNS*) de rendement $r = \frac{k}{n}$ et de longueur de contrainte L. Le treillis de ce code possède $2^{k(L-1)}$ états ; 2^k branches partent et arrivent à chaque état. La fermeture du treillis d'un tel code nécessite $k\nu$ bits à «0». Ce code est défini par les polynômes générateurs $g_0(x), g_1(x), \ldots, g_{n-1}(x)$.

La figure 3.3 donne le schéma du codeur d'un code NRNS de rendement 3/4, de longueur de contrainte $L = 2$ et de polynômes générateurs en octal $(13, 25, 61, 47)$. Nous notons par la suite C_{ex}^{NRNS} ce code, et H_{ex}^{NRNS} sa matrice de parité.

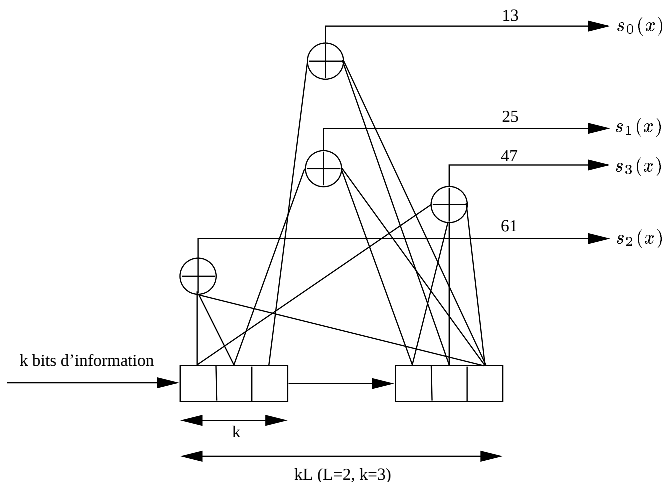


Fig. 3.3 – Codeur du code NRNS de rendement 3/4 et de polynômes générateurs $(13, 25, 61, 47)$

Pour un code convolutif non récursif non systématique de rendement $\frac{k}{n}$, il existe $n-k$ syndromes qui sont obtenus à partir des n équations du code convolutif reliant les k entrées $e_i(x),\ i = 0, \ldots, k-1$, aux n sorties $s_i(x),\ i = 0, \ldots, n-1$, du codeur. Chaque syndrome est de longueur $n(k(L-1)+1)$. Comme dans le cas d'un code RSC, chaque syndrome est décalé dans la matrice de n bits, à chaque instant de codage. La fermeture du treillis correspond aux $n(L-1)$ dernières colonnes et aux $n(L-1)$ dernières lignes de la matrice de parité ; les dernières lignes sont les syndromes de “contrôle” de la fermeture du treillis.

Par exemple, pour le code C_{ex}^{NRNS} de rendement 3/4, les 4 équations reliant les entrées aux sorties du codeur sont :

$$\begin{aligned}
s_0(x) &= e_2(x) + x(e_1(x) + e_2(x))\ , \\
s_1(x) &= e_1(x) + x(e_0(x) + e_2(x))\ , \\
s_2(x) &= e_0(x) + e_1(x) + xe_2(x)\ , \\
s_3(x) &= e_0(x) + x(e_0(x) + e_1(x) + e_2(x))\ .
\end{aligned}$$

Elles définissent un seul syndrome donné par :

$$s_1(x) + s_2(x) + s_3(x) + x(s_0(x) + s_2(x)) + x^2(s_2(x)) + x^3(s_0(x) + s_1(x) + s_2(x) + s_3(x)) = 0\ .$$

La matrice de parité de ce code est représentée sur la figure 3.4 ; la fermeture du treillis nécessite $k*(L-1) = 3$ bits à 0. Le rendement de ce code pour $N_{NRNS} = 40$ est égal à $27/40 = 0.675$.

Ces deux constructions donnent des matrices de parité à faible densité lorsque la longueur N du code en blocs résultant de la fermeture du code convolutif est suffisamment grande par rapport aux nombres de bits impliqués dans les syndromes. Elles permettent de construire la matrice de parité d'un code GLD convolutif.

$N_{NRNS} = 40$ — fermeture du treillis

$$
H_{ex}^{NRNSC} = \left[\begin{array}{c|c}
011100000000000000000000000000000000 & 0000 \\
110001110000000000000000000000000000 & 0000 \\
001011000111000000000000000000000000 & 0000 \\
111100101100011100000000000000000000 & 0000 \\
000011110010110001110000000000000000 & 0000 \\
000000001111001011000111000000000000 & 0000 \\
000000000000111100101100011100000000 & 0000 \\
000000000000000011110010110001110000 & 0000 \\
000000000000000000001111001011000111 & 0000 \\
\hline
000000000000000000000000111100101100 & 0111 \\
000000000000000000000000000011110010 & 1100 \\
000000000000000000000000000000001111 & 0010 \\
000000000000000000000000000000000000 & 1111
\end{array}\right] \quad \text{fermeture du treillis}
$$

syndrome de longueur n(k(L-1)+1)

FIG. 3.4 – Représentation de la matrice de parité du code RSC C_{ex}^{NRNS} de longueur 40 bits

3.1.2 Matrice de parité d'un code GLD

Par définition d'un code GLD convolutif $\mathcal{C}$ construit à partir d'un code convolutif $C_c^1(n,k)$ dont le treillis est fermé (le code en blocs résultant est noté C^1) et comportant J niveaux, nous avons :

$$\mathcal{C} = \bigcap_{i=1}^{J} C^i \quad \text{avec } C^i = \pi_i(C^1),\ i = 1, \ldots, J\ .$$

où π_1 est la permutation identité ($C^1 = \pi_1(C^1)$).

La matrice de parité de $\mathcal{C}$ est donc constituée de J sous-matrices H^i, $i = 1, \ldots, J$, chacune d'entre elles correspondant à un code C^i. Par analogie avec les codes LDPC, chaque équation de parité de la première sous-matrice H^1 est remplacée par les $n-k$ syndromes du code convolutif. Les $J-1$ autres sous-matrices sont obtenues en appliquant les permutations aléatoires $\pi_2, \ldots, \pi_J$ sur les colonnes de la matrice H^1. Les colonnes correspondant à la fermeture du treillis ne sont pas entrelacées. La matrice résultante reste à faible densité ; cependant, les équations de parité à l'intérieur d'une sous-matrice ne sont plus disjointes, ce qui rend inefficace le décodage proposé par Gallager [42]. Il faut, par conséquent, utiliser la structure du code convolutif pour le décodage, comme nous le verrons dans le paragraphe 3.4.

La forme de la matrice de parité d'un code GLD convolutif est représentée sur la figure 3.5. Pour plus de clarté, la fermeture du treillis a été omise. Cette matrice est utilisée pour le codage du code GLD convolutif. Elle peut être aisément mise sous forme systématique par élimination gaussienne du fait de sa structure. La matrice génératrice correspondante est alors immédiatement déduite, et utilisée pour le codage du code GLD convolutif, aux permutations de colonnes près. Nous n'avons malheureusement trouvé aucun schéma de codage s'appuyant sur la structure du code GLD, c'est-à-dire sur les codes convolutifs le constituant. La complexité du codage des codes GLD convolutifs augmente donc avec la longueur du code et peut nécessiter, dans certains cas, des ressources importantes en termes de mémoire et d'opérations afin de définir la matrice génératrice du code et d'effectuer l'opération de codage.

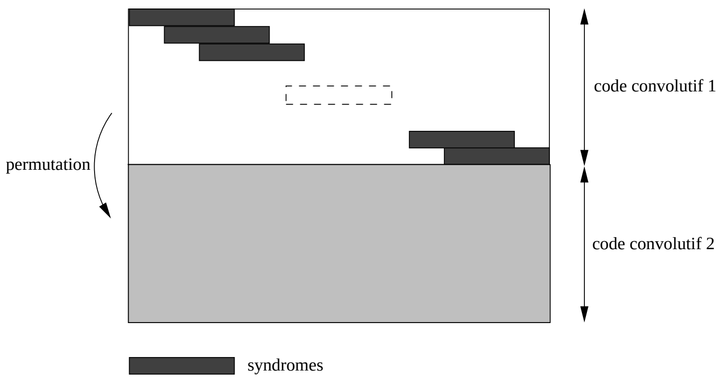


Fig. 3.5 – Structure de la matrice de parité d'un code GLD convolutif à $J = 2$ niveaux

Calculons maintenant le rendement d'un code GLD convolutif construit à partir de J codes constituants, identiques à une permutation près au code C^1. Chaque sous-matrice est de taille $(N - K_1) \times N$, où K_1 est la dimension du code en blocs C^1 et N est la longueur du code GLD convolutif $\mathcal{C}$. La matrice du code de parité est, par conséquent, de dimension $J(N - K_1) \times N$. Comme les permutations $\pi_i,\ 2 = 1, \dots, J$, sont aléatoires, le rendement du code GLD convolutif est minoré par :

$$R = \frac{K}{N} \geq 1 - \frac{J(N - K_1)}{N} \ ,$$

soit, si r_1 est le rendement du code en blocs C^1,

$$R \geq 1 - J(1 - r_1) \ , \tag{3.4}$$

où l'inégalité provient du fait que certaines lignes de la matrice peuvent être dépendantes suite aux permutations aléatoires. Cette inégalité montre que, pour avoir un rendement total positif, le rendement du code constituant ne doit pas être inférieur à $1 - \frac{1}{J}$. Dans le cas d'un GLD convolutif construit à partir de $J = 2$ codes, le rendement total du code GLD s'écrit $R \geq 2r_1 - 1$. Par conséquent, le rendement du code constituant ne doit pas être inférieur à $1/2$.

Après avoir étudié la représentation matricielle des codes GLD convolutifs, intéressons-nous maintenant à leur représentation graphique, très utile pour mieux comprendre la propagation de l'information lors du décodage.

3.1.3 Représentation graphique

Sans perdre en généralités, nous considérons dans ce paragraphe un code GLD convolutif $\mathcal{C}$ construit à partir de deux codes identiques de longueur N. Ce code peut être représenté de manière très simple à l'aide d'un graphe bayésien. En effet, nous avons vu au chapitre 2 que chaque code convolutif C^i est caractérisé par une chaîne de plusieurs super-sommets, où chaque super-sommet inclut l'état du codeur, les n bits codés à la sortie du codeur convolutif à chaque instant, et la sortie du canal correspondante. Le réseau bayésien du code GLD convolutif

$\mathcal{C}$ est alors obtenu en reliant les super-sommets des deux chaînes représentant les deux codes convolutifs par un entrelaceur, via n branches. La figure 3.6 donne le graphe bayésien d'un code GLD convolutif construit à partir de deux codes de rendement 3/4.

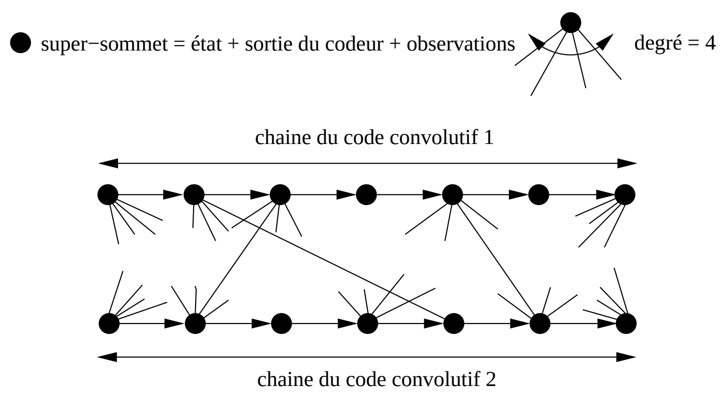


FIG. 3.6 – Graphe bayésien d'un code GLD convolutif construit à partir de deux codes de rendement 3/4

La représentation de Tanner d'un code GLD convolutif est également très simple. Par analogie avec les codes LDPC, chaque bit appartient à $J = 2$ codes convolutifs, au lieu de j équations de parité. Chaque sous-code agit sur N bits codés. Par conséquent, dans le graphe de Tanner, les sommets sous-codes sont maintenant des codes convolutifs. Dans notre cas particulier où $J = 2$, nous obtenons le graphe représenté sur la figure 3.7.

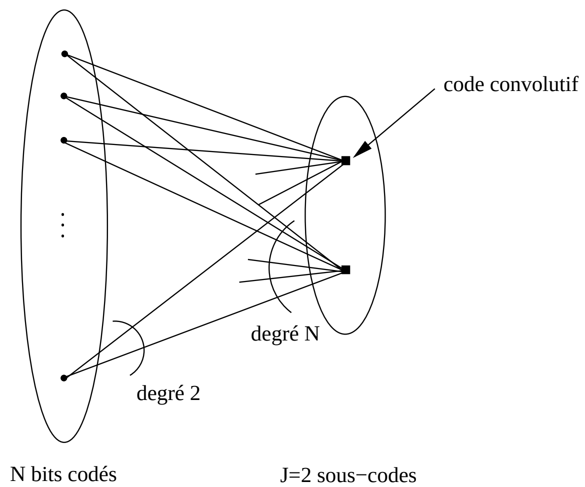


FIG. 3.7 – Graphe de Tanner d'un code GLD convolutif construit à partir de deux codes

3.2 Interprétation des turbo-codes en termes de GLD convolutif

Les turbo-codes série [7] et parallèle [11] peuvent être interprétés comme des cas particuliers des codes GLD convolutifs, c'est-à-dire comme l'intersection de deux (ou plus) codes convolutifs entrelacés. Par la suite, nous négligeons la fermeture du treillis dans la représentation de la matrice de parité de ces codes.

Soit $\mathcal{C}_{\text{PCCC}}(N,K)$ un turbo-code parallèle construit à partir de deux codes RSC identiques liés via un entrelaceur de longueur K. La matrice de parité H^1 du premier sous-code $C^1(N_1,K)$ est similaire à la matrice de parité définie dans le paragraphe 3.1.1, mais les K colonnes associées aux bits d'information peuvent maintenant être regroupées dans la partie gauche de la matrice pour constituer la partie systématique du code. Le code C^1 est ensuite étendu en ajoutant N_1-K colonnes à 0 dans la partie droite de la matrice H^1. On obtient le code C^1_{ext} de matrice de parité H^1_{ext}. Cette matrice est donc constituée de trois blocs :
- le premier bloc est de dimension $(N_1-K)\times K$ et regroupe les bits d'information
- le second bloc est de dimension $(N_1-K)\times(N_1-K)$ et est associé aux bits de parité
- enfin le troisième bloc est de dimension $(N_1-K)\times(N-(N_1-2K))$ et ne contient que des "0".

Le turbo-code parallèle $\mathcal{C}_{\text{PCCC}}(N,K)$ est un code GLD convolutif obtenu en prenant l'intersection de C^1_{ext} et de $\pi(C^1_{ext})$, où la permutation π agit de manière aléatoire sur les K bits d'information, c'est-à-dire sur les K premières colonnes de la matrice H^1_{ext}, et permute les deuxième et troisième blocs de cette matrice. La figure 3.8 illustre la matrice de parité d'un turbo-code parallèle ainsi construite. Du fait de la structure de cette matrice et, en particulier, de la présence de colonnes

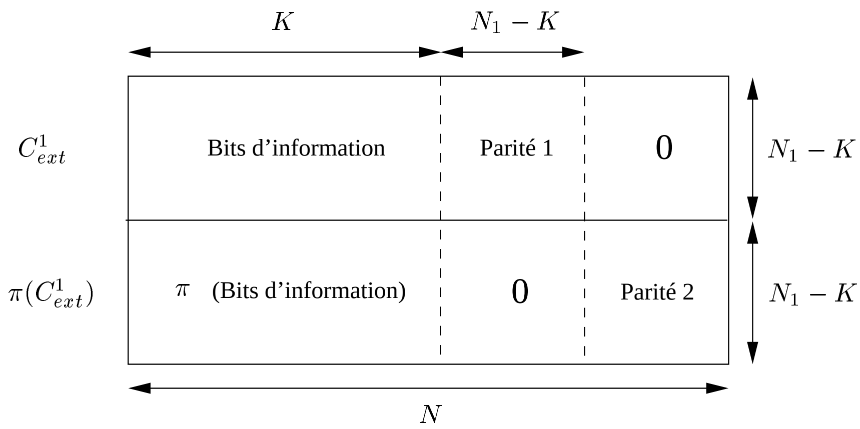


FIG. 3.8 – Matrice de parité d'un turbo-code parallèle

nulles dans les sous-matrices et de la définition de la permutation, le rendement du turbo-code parallèle ne peut être obtenu en appliquant directement la formule (3.4) sur le rendement r_1 du code C^1. En effet, si $r_1=\frac{K}{N_1}$, le rendement total du turbo-code est égal à :

$$R_{\text{PCCC}} = 1-2\frac{N_1-K}{K+2(N_1-K)} = \frac{r_1}{2-r_1}\ ,$$

soit $R_{\text{PCCC}}=1/3$ lorsque $r_1=1/2$. Pour retrouver cette valeur $R_{\text{PCCC}}=1/3$, il nous faut appliquer la formule (3.4) en considérant le rendement du code étendu $r_1^{ext}=\frac{2}{3}$.

Le raisonnement est similaire pour un turbo-code série $\mathcal{C}_{\text{SCCC}}$ construit à partir de deux codes convolutifs $C^1(N_1, K)$ (code extérieur NRNS) et $C^2(N, N_1)$ (code intérieur RSC), liés par un entrelaceur de longueur égale à N_1. La première sous-matrice H^1 du code NRNS C^1, de dimension $(N_1 - K) \times N$, est obtenue en étendant la matrice H^1 du code NRNS C^1 à l'aide de $N - N_1$ colonnes nulles. Le code résultant est appelé C^1_{ext} et sa matrice de parité est notée H^1_{ext}. Notons que les N_1 premières colonnes de H^1_{ext} représentent N_1 bits codés, dans le cas d'un code non récursif non systématique : il n'y a plus de sous-blocs séparant les bits d'information des bits de parité, *i.e.*, la matrice H^1_{ext} n'est plus constituée que de deux blocs, l'un contenant les bits codés et l'autre uniquement des bits à "0". Le code $C^2(N, N_1)$ est, quant à lui, un code récursif systématique. Sa matrice de parité est donc composée de deux sous-blocs, l'un de dimension $(N - N_1) \times N_1$ contenant les bits d'information, l'autre de dimension $(N - N_1) \times (N - N_1)$ contenant les bits de parité. Le turbo-code série est un code GLD convolutif obtenu en prenant l'intersection du code C^1_{ext} et d'une version permutée du code C^2, notée $\pi(C^2)$, où π est un entrelaceur particulier agissant sur les N_1 premières colonnes de H_2 seulement. La figure 3.9 présente la matrice de parité d'un turbo-code série ainsi construite. De même que pour le turbo-

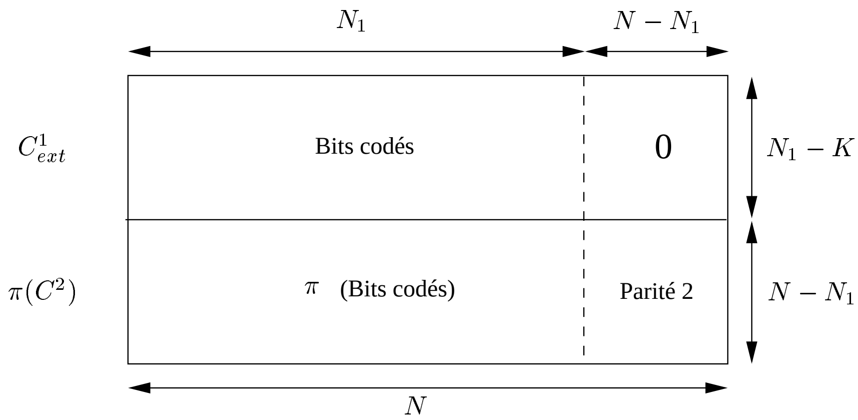


Fig. 3.9 – Matrice de parité d'un turbo-code série

code parallèle, la formule (3.4) ne peut être directement appliquée pour calculer le rendement du turbo-code série. Si $r_1 = \frac{K}{N_1}$ et $r_2 = \frac{N_1}{N}$ sont les rendements respectifs des codes C^1 et C^2, alors le rendement du turbo-code série est :

$$R_{\text{SCCC}} = r_1 \ r_2 \ .$$

La construction des codes GLD convolutifs étant décrite, nous pouvons maintenant examiner les performances que l'on peut espérer de tels codes.

3.3 Analyse des performances des codes GLD convolutifs

Nous nous intéressons dans cette partie aux performances des codes GLD convolutifs, lorsque ceux-ci sont décodés en optimisant le critère du maximum de vraisemblance (critère ML). Nous déterminons dans un premier temps la distribution de poids moyenne de ces codes, ce qui fournit une borne supérieure de leur distance minimale. Puis nous déduisons de cette distribution une borne de la probabilité d'erreur par bit et par mot sur le canal gaussien. Enfin, nous étudions une borne inférieure sur la distance minimale des codes GLD convolutifs, ce qui nous permettra d'établir une propriété asymptotique sur la distance de ces codes.

Nous supposons pour simplifier notre étude que le code GLD convolutif considéré est construit à partir d'un code convolutif transformé en un code en blocs linéaire C^1 par fermeture de son treillis et de $J-1$ permutations aléatoires de ce code, $\pi_2(C^1), \ldots, \pi_J(C^1)$. Le code GLD convolutif $\mathcal{C}$ et le code constituant C^1 sont de longueur N, de dimensions respectives K et K_1 et de distances minimales respectives $d_{\mathrm{H\,min}}$ et d_1.

3.3.1 Distribution de poids moyenne

Soit $A_1(w)$ le nombre de mots de code de C^1 de poids w. La distribution de poids moyenne d'un code GLD convolutif s'obtient en moyennant sur l'ensemble des entrelaceurs aléatoires $\pi_i,\ i = 2, \ldots, J$, de longueur N.

Soient $P(w)$ la probabilité qu'une séquence de poids w soit un mot de code de $\mathcal{C}$, et $A(w)$ le nombre total de mots de code de $\mathcal{C}$ de poids w. Il existe C_N^w mots de poids w et de longueur N ; par conséquent, nous avons la relation suivante :

$$A(w) = C_N^w P(w) \ .$$

Soient $P_1(w), \ldots, P_J(w)$ les probabilités qu'une séquence de poids w soit un mot de code de $C^1, \ldots, C^J$ respectivement. Les codes $C^2, \ldots, C^J$ étant des versions permutées du code C^1, les probabilités $P_i(w)$ sont toutes identiques et égales à $P_1(w)$. Or, par définition, le code GLD convolutif $\mathcal{C}$ est l'intersection des codes indépendants $C^1, \ldots, C^J$, donc la probabilité $P(w)$ s'écrit :

$$P(w) = \prod_{i=1}^{J} P_i(w) = (P_1(w))^J \ .$$

De plus,

$$A_1(w) = C_N^w P_1(w) \ .$$

La distribution moyenne de poids du code GLD convolutif est donnée par le théorème suivant :

Théorème 3.3.1 *(Nombre moyen de mots de code)*
Soit $\mathcal{C}(N, K)$ un code GLD convolutif. Le nombre moyen $A(w)$ de mots de code de $\mathcal{C}$ de poids w est :

$$A(w) = \frac{(A_1(w))^J}{\left(C_N^w\right)^{J-1}} \ . \tag{3.5}$$

La distribution de poids $A_1(w)$ du code convolutif C^1 peut être calculée à l'aide de sa matrice de transition T_1 élevée à la puissance $\frac{N}{n}$, où $\frac{N}{n}$ correspond au nombre de branches dans le treillis du code [27]. La construction de la matrice T_1 et l'obtention de $A_1(w)$ sont expliqués en annexe C.

La formule (3.5) a été appliquée à un code GLD convolutif de longueur $N = 200$, construit à partir de deux codes RSC de rendement 3/4 et de polynômes générateurs $(15, 16, 13, 17)$. Le rendement total du code $\mathcal{C}$ est égal à 1/2. Les premiers termes de sa distribution de poids sont donnés dans la table 3.1. Cette distribution de poids est comparée à la distribution de poids d'un code aléatoire pour $N = 200$ sur la figure 3.10. Les deux distributions sont très proches, voire confondues au niveau de leur partie centrale.

Poids w	Coefficient $A(w)$	Poids w	Coefficient $A(w)$
4	3.4e-3	17	3.7e-1
5	1.4e-3	18	9.1e-1
6	8.9e-4	19	2.3e+0
7	6.3e-4	20	6.1e+0
8	1.3e-3	21	1.66e+1
9	2.0e-3	22	4.65e+1
10	3.1e-3	23	1.34e+2
11	4.8e-3	24	3.96e+2
12	8.6e-3	25	1.19e+3
13	1.6e-2	26	3.71e+3
14	3.4e-2	27	1.16e+4
15	7.2e-2	28	3.75e+4
16	1.6e-1	29	1.22e+5

Tab. 3.1 – Distribution de poids moyenne du code GLD construit à partir du code C_{ex}^{RSC} avec $J = 2$

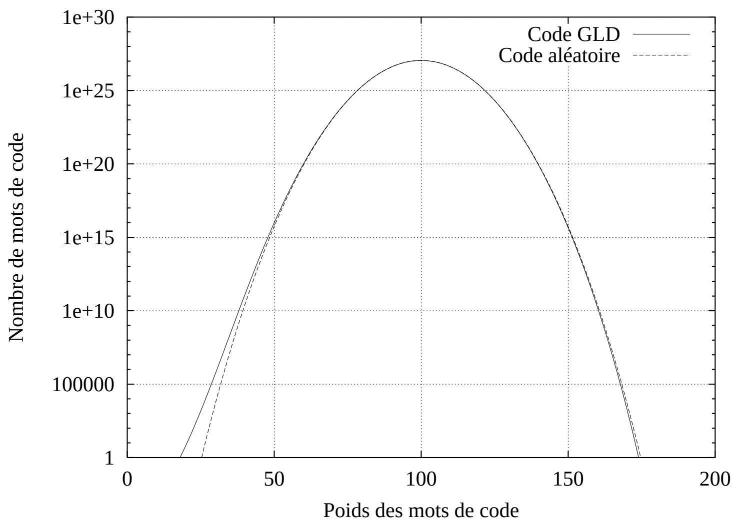


Fig. 3.10 – Comparaison de la distribution de poids du code GLD construit à partir de C_{ex}^{RSC} $(J = 2)$ et de la distribution de poids d'un code aléatoire, $N = 200$

La distribution de poids moyenne d'un code GLD convolutif permet d'obtenir une borne supérieure Δ de sa distance minimale $d_{\mathrm{H\,min}}$, pour une longueur de code N fixée. En effet,

$$\Pr(d_{\mathrm{H\,min}} \leq \Delta) = \Pr(\exists\ \mathbf{v} \in (GF(2))^N, w_{\mathbf{v}} \leq \Delta, \mathbf{v} \in \mathcal{C})\ ,$$

où $w_{\mathbf{v}}$ désigne le poids de la séquence $\mathbf{v}$. La borne de l'union appliquée à cette égalité donne :

$$\Pr(d_{\mathrm{H\,min}} \leq \Delta) \leq \sum_{i=1}^{\Delta} \Pr(\exists\ \mathbf{v} \in (GF(2))^N, \mathbf{v} \in \mathcal{C}, w_{\mathbf{v}} = i)\ .$$

Or il existe $M = C_N^i$ séquences de poids i dans $(GF(2))^N$, notées $\mathbf{s}_1, \ldots, \mathbf{s}_M$, donc :

$$\Pr(d_{\mathrm{H\,min}} \leq \Delta) \leq \sum_{i=1}^{\Delta} \Pr((\mathbf{s}_1 \in \mathcal{C}, w_{\mathbf{s}_1} = i) \cup (\mathbf{s}_2 \in \mathcal{C}, w_{\mathbf{s}_2} = i) \cup \ldots \cup (\mathbf{s}_M \in \mathcal{C}, w_{\mathbf{s}_M} = i))\ ,$$

soit,

$$\Pr(d_{\mathrm{H\,min}} \leq \Delta) \leq \sum_{i=1}^{\Delta} \sum_{j=1}^{M} \Pr(\mathbf{s}_j \in \mathcal{C}, w_{\mathbf{s}_j} = i)\ .$$

La probabilité $\Pr(\mathbf{s}_j \in \mathcal{C}, w_{\mathbf{s}_j} = i)$ est, par définition, égale à la probabilité $P(i)$, $\forall j$, d'où :

$$\begin{aligned} \Pr(d_{\mathrm{H\,min}} \leq \Delta) &\leq \sum_{i=1}^{\Delta} M\ P(i) \\ &\leq \sum_{i=1}^{\Delta} C_N^i P(i) \\ &\leq \sum_{i=1}^{\Delta} A(i)\ . \end{aligned} \tag{3.6}$$

Par conséquent, une borne supérieure est donnée par Δ, le plus petit entier vérifiant :

$$\left\lfloor \sum_{w=1}^{\Delta} A(w) \right\rfloor = 1\ .$$

Le tableau 3.2 donne quelques valeurs de Δ pour le code GLD convolutif construit à partir du code C_{ex}^{RSC}, pour différentes longueurs de N.

Longueur N	Borne Supérieure Δ
100	11
200	18
300	24
400	31
500	37
600	43
700	49
800	56

TAB. 3.2 – Borne supérieure Δ en fonction de la longueur du code N, $R = \frac{1}{2}$

Une borne supérieure de la distance minimale des turbo-codes série et parallèle peut également être obtenue de manière identique à partir de leurs distributions de poids moyennes [6]. La figure

3.11 représente l'évolution de la borne supérieure de la distance minimale, pour des longueurs de code allant de $N = 100$ à $N = 800$, un rendement total $R = 1/3$, et un entrelacement pseudo-aléatoire, dans les trois cas suivants :

- GLD convolutif, $J = 2$, code constituant $(23, 31, 35)$ (16 états),
- turbo-code parallèle, $J = 2$, code constituant $(23, 35)$ (16 états),
- turbo-code série, $J = 2$, codes constituants $(7, 5)$ (4 états) et $(23, 31, 35)$ (16 états).

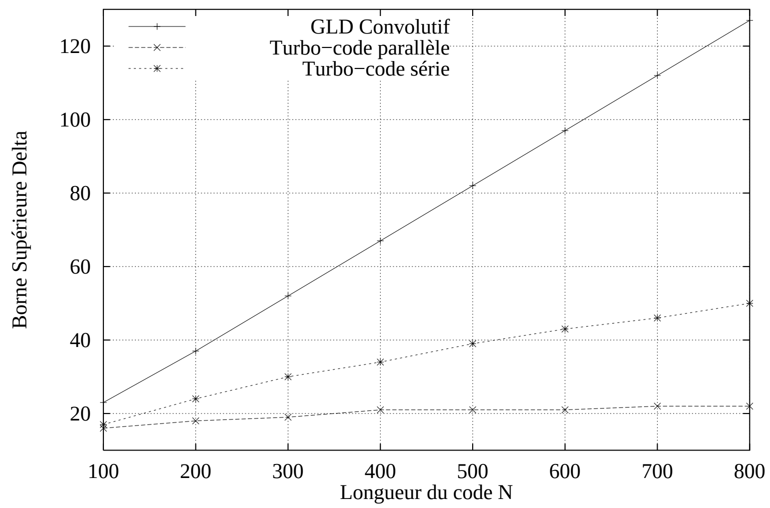


Fig. 3.11 – Évolution de la borne supérieure Δ de la distance minimale

Plus que les valeurs mêmes de la borne Δ, il est intéressant de noter sur cette figure la nature de l'évolution de Δ en fonction de N. En effet, dans le cas des codes GLD convolutifs, pour $J = 2$, cette évolution est linéaire, contrairement aux turbo-codes parallèle et série. Pour le turbo-code parallèle, la distance ne varie que très peu en fonction de la longueur N du code. Ce comportement se rapproche des résultats sur la distance minimale présentés par Kahale et Urbanke [50] et rappelés au chapitre 2.

3.3.2 Probabilité d'erreur sur canal gaussien

La distribution moyenne de poids du code GLD convolutif peut également nous permettre de déduire une borne de la probabilité d'erreur de ce code sur le canal gaussien. En effet, l'entrelaceur présent dans les codes GLD convolutifs agit sur l'ensemble des bits codés, de telle sorte que tous les bits codés sont protégés de manière égale. Par conséquent, seul le polynôme énumérateur de poids $A(w)$ est nécessaire pour calculer la probabilité d'erreur par mot et par bit lorsque le décodage du code GLD $\mathcal{C}$ optimise le critère ML : le polynôme énumérateur de poids conditionnel par rapport au poids de l'entrée n'est plus nécessaire contrairement au cas des turbo-codes (*cf.* chapitre 2). Les probabilités d'erreur par mot et par bit sont alors obtenues en appliquant la borne de l'union et sont majorées par :

$$
\begin{aligned}
Pe_{\mathrm{w}} &\leq \sum_{w=d_1}^{N} A(w) Q\left(\sqrt{Rw\frac{2E_b}{N_0}}\right) ,\\
Pe_{\mathrm{b}} &\leq \sum_{w=d_1}^{N} \frac{w}{N} A(w) Q\left(\sqrt{Rw\frac{2E_b}{N_0}}\right) ,
\end{aligned}
$$

où $\frac{E_b}{N_0}$ est le rapport signal-à-bruit moyen par bit et $Q(x)$ désigne la fonction d'erreur. La figure 3.12 donne la probabilité d'erreur moyenne par bit obtenue après le décodage ML du code GLD convolutif de rendement $R = 1/2$ construit à partir du code C_{ex}^{RSC} ($J = 2$). Deux longueurs N du code sont envisagées : $N = 200$ et $N = 800$. Le canal est un canal AWGN.

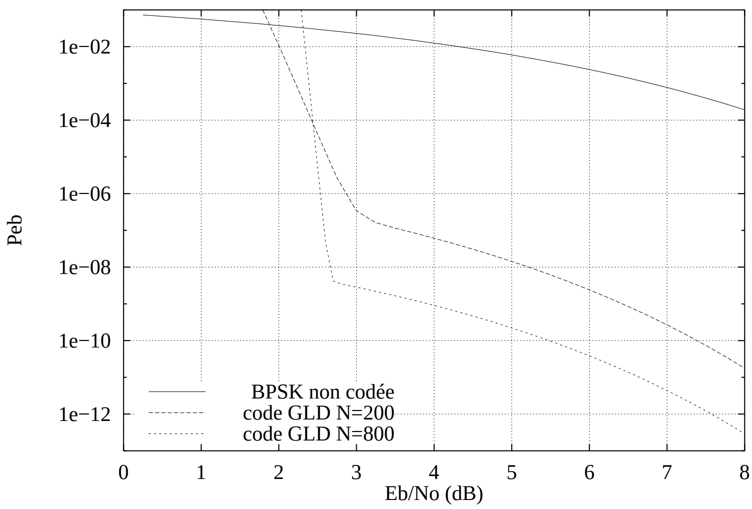


FIG. 3.12 – Probabilité d'erreur moyenne par bit pour $N = 200$ et $N = 800$

Notons que des bornes plus précises à faible rapport signal-à-bruit peuvent être considérées pour estimer les probabilités d'erreur par mot et par bit. Le lecteur est en particulier invité à se référer aux travaux de Duman et Salehi [30], ou de Sason et Shamai [69].

Nous avons vu que la distribution moyenne de poids des codes GLD convolutifs nous permet de déduire une borne supérieure de la distance minimale moyenne de ces codes, et d'obtenir une borne supérieure de leurs probabilités d'erreur par bit et par mot pour le canal gaussien. Nous allons maintenant nous intéresser à une borne inférieure de la distance minimale moyenne.

3.3.3 Borne inférieure de la distance minimale des codes GLD convolutifs

Gallager [42] a montré que les codes LDPC construits à partir de simples équations de parité sont asymptotiquement bons lorsque $j \geq 3$ (j “1” par colonne), c'est-à-dire que la distance minimale moyenne $d_{\mathrm{H\,min}}(LDPC)$ d'un code LDPC de longueur N est bornée inférieurement par δN, où δ est une constante positive. Par ailleurs, il a également été démontré [14][62] que les codes de Tanner construits à partir de graphes bipartites aléatoires et de codes BCH sont asymptotiquement bons pour $J = 2$ seulement (2 sous-matrices). Le théorème 3.3.2 établit que la distance minimale des codes GLD convolutifs vérifie la même propriété.

Théorème 3.3.2 *(GLD convolutifs asymptotiquement bons)*
Le code GLD convolutif $\mathcal{C}(N, K, d_{H\min})$ construit à partir d'un code convolutif C_c^1 de rendement $\frac{k}{n}$, transformé en un code en blocs $C_1(N, K_1)$ par fermeture de son treillis, est asymptotiquement bon pour $J \geq 2$. Quand N est suffisamment grand, sa distance minimale normalisée $\delta_{min} = d_{H\min}/N$ est bornée inférieurement par une constante positive δ. La constante δ est la plus petite racine positive non nulle de l'équation $B(\lambda) = 0$, où

$$B(\lambda) = (J-1)H(\lambda) - J\left(\left(\frac{1}{n} - \frac{\lambda}{d_1}\right)H\left(\frac{\lambda}{\frac{d_1}{n} - \lambda}\right) + \lambda H\left(\frac{1}{d_1}\right) + \beta\lambda H\left(\frac{1}{\beta d_1}\right) + \frac{\lambda}{d_1}\ln(a_{d_1})\right) , \tag{3.7}$$

et $B(\lambda) > 0$ pour $\lambda \in]0 \dots \delta[$.
Dans l'équation ci-dessus,
- *$H(x) = -x\ln(x) - (1-x)\ln(1-x)$ désigne la fonction entropie naturelle,*
- *β est une constante positive dépendant de la fonction de transfert du code convolutif C_c^1,*
- *d_1 est la distance libre du code C_c^1,*
- *a_{d_1} est le nombre de chemins de poids d_1 dans le diagramme d'état du code C_c^1.*

Démonstration :
Sans perdre en généralités, nous considérons que le code C_c^1 est un code convolutif récursif systématique. Après fermeture de son treillis ce code est noté C^1 et sa longueur est N.

Pour démontrer le théorème 3.3.2, nous procédons comme suit. Nous évaluons tout d'abord une borne supérieure du nombre de mots de code de poids w de C^1, noté $A_1(w)$, en décomposant chaque mot de code en événements d'erreurs simples. Nous introduisons alors le coefficient $A_1(w, e, \rho)$, qui désigne le nombre de mots de code de poids w de C^1 constitués de e événements d'erreurs et de ρ branches. A l'aide des bornes supérieures obtenues sur $A_1(w, e, \rho)$ puis sur $A_1(w)$ et du théorème 3.3.1, nous déduisons une borne supérieure du nombre $A(w)$ de mots de code de poids w du code $\mathcal{C}$. Cette borne supérieure est une borne exponentielle en N de la forme $\exp(-NB(\lambda))$. L'étude de la fonction B permet d'achever la démonstration du théorème et d'évaluer la distance minimale normalisée δ.

1. Borne supérieure du coefficient $A_1(w, e, \rho)$:

Pour évaluer le nombre de mots de code de C^1 de poids w, nous introduisons le coefficient $A_1(w, e, \rho)$ désignant le nombre de mots de code de C^1 de poids w, formés par la concaténation de e événements d'erreurs simples consécutifs dans le treillis et constitués de ρ branches [8]. Un mot de code de poids w formé de 3 événements d'erreurs simples consécutifs et de ρ branches est représenté sur la figure 3.13.

Il existe $C^e_{\frac{N}{n}-\rho+e}$ événements d'erreurs composés de e événements d'erreurs simples (non obligatoirement consécutifs) pour un total de ρ branches : $\frac{N}{n} - \rho + e$ correspond en effet au nombre de positions sur la fenêtre de longueur $\frac{N}{n}$ branches où peut commencer un événement d'erreur simple. Par conséquent, le nombre total de mots de code $A_1(w)$ de poids w peut s'écrire

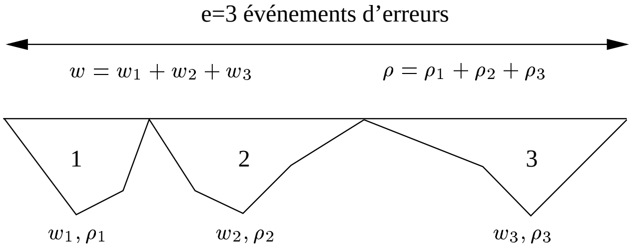


FIG. 3.13 – Mot de code constitué de e événements d'erreurs consécutifs et de ρ branches

comme la somme sur tous les mots de code constitués de e événements d'erreurs et de ρ branches :

$$A_1(w) = \sum_e \sum_\rho C^e_{\frac{N}{n}-\rho+e} A_1(w,e,\rho) \ . \tag{3.8}$$

Le coefficient $A_1(w,e,\rho)$ s'exprime ainsi :

$$A_1(w,e,\rho) = \sum_{\substack{w_1,w_2,\cdots,w_e \\ w_1+w_2+\cdots+w_e=w}} \sum_{\substack{\rho_1,\rho_2,\cdots,\rho_e \\ \rho_1+\rho_2+\cdots+\rho_e=\rho}} \prod_{i=1}^{e} a(w_i,\rho_i) \ , \tag{3.9}$$

où $a(w_i,\rho_i)$ est le nombre de mots de code de poids w_i composés de ρ_i branches correspondant à un seul événement d'erreur. Le coefficient $a(w_i,\rho_i)$ peut être relié à la fonction de transfert $T(D,L)$ du code convolutif C_c^1 par :

$$T(D,L) = \sum_{w,\rho} a(w,\rho) D^w L^\rho \ ,$$

pour D et L positifs définis dans le domaine de convergence de $T(D,L)$. La fonction $T(D,L)$ est une somme de termes positifs ou nuls, par conséquent, en appliquant un raisonnement similaire à celui de Gallager [42], nous obtenons :

$$a(w_i,\rho_i) \leq T(D,L) D^{-w_i} L^{-\rho_i} \quad \forall \ w_i,\rho_i \ . \tag{3.10}$$

En combinant les équations (3.9) et (3.10) nous avons :

$$A_1(w,e,\rho) \leq \sum_{\substack{w_1,w_2,\cdots,w_e \\ w_1+w_2+\cdots+w_e=w}} \sum_{\substack{\rho_1,\rho_2,\cdots,\rho_e \\ \rho_1+\rho_2+\cdots+\rho_e=\rho}} T(D,L)^e D^{-w} L^{-\rho} \ , \tag{3.11}$$

car

$$\begin{aligned} \prod_{i=1}^{e} a(w_i,\rho_i) &\leq \prod_{i=1}^{e} T(D,L) D^{-w_i} L^{-\rho_i} \\ &\leq T(D,L)^e D^{-\sum_{i=1}^{e} w_i} L^{-\sum_{i=1}^{e} \rho_i} \\ &\leq T(D,L)^e D^{-w} L^{-\rho} \ . \end{aligned}$$

La somme sur toutes les valeurs de w_i , $i = 1,\ldots,e$, telles que $w_1+w_2+\ldots+w_e = w$, correspond au nombre de partitions ordonnées d'un entier positif w en e éléments positifs $w_1,\ldots,w_e$. Il existe exactement C^{e-1}_{w-1} partitions ainsi définies. Il en est de même pour la somme sur toutes les valeurs de ρ_i , $i = 1,\ldots,e$, telles que $\rho_1 + \rho_2 + \ldots + \rho_e = \rho$: $C^{e-1}_{\rho-1}$ partitions ordonnées peuvent être déterminées pour décomposer un entier positif ρ en e parties positives. Par conséquent, l'inégalité (3.11) s'écrit :

$$A_1(w,e,\rho) \leq C_{w-1}^{e-1} C_{\rho-1}^{e-1} T(D,L)^e D^{-w} L^{-\rho} \ .$$

Pour $L = 1$, l'équation précédente devient :

$$A_1(w,e,\rho) \leq C_{w-1}^{e-1} C_{\rho-1}^{e-1} T(D)^e D^{-w} \ . \tag{3.12}$$

Considérons maintenant les valeurs maximales de e et de ρ :

- Chaque événement d'erreurs a un poids au moins égal à la distance libre d_1 du code C_c^1. Par conséquent, le nombre maximal d'événements d'erreurs e correspondant à un poids w est $e_{\max} = \frac{w}{d_1}$.
- Par ailleurs, une borne supérieure du nombre de branches ρ est βw, où β est une constante donnée par le cycle du treillis (ou du diagramme d'état) du code C_c^1 qui maximise le rapport $\frac{\rho}{w}$. Considérons par exemple le cas d'un code récursif systématique de rendement $\frac{k}{k+1}$, de polynôme de rétroaction primitif. En envoyant un “1” en entrée du codeur suivis de “0” uniquement, tous les états non nuls du codeur sont parcourus avant de revenir à l'état initial, puis ils sont de nouveau passés en revue de manière cyclique tant qu'il n'y a pas de “1” en entrée du codeur pour sortir de ce cycle. Ceci nous permet de trouver :
  - le cycle produisant un poids minimal,
  - le nombre de branches maximal pour atteindre ce poids.

  La figure 3.14 représente la succession des états pour le code C_{ex}^{RSC}, et les différentes branches de poids 1 reliant les états. L'analyse de cette figure permet de dégager le cycle dessiné sur la figure 3.15 : sa longueur est égale à 5 et son poids à 2 ce qui donne une valeur de β égale à $\frac{5}{2}$ pour ce code.

  Les figures 3.16 et 3.17 donnent le cycle maximisant le rapport $\frac{\rho}{w}$ pour les codes récursifs systématiques de rendement $R = 3/4$ et de polynômes générateurs respectifs $(31, 23, 27, 37)$ et $(51, 63, 73, 57)$. Dans ces deux cas, les cycles sont de longueurs et de poids respectifs $(\rho = 7, w = 2)$ et $(\rho = 4, w = 1)$, ce qui correspond aux valeurs de $\beta = 7/2$ et $\beta = 4$ respectivement.

  L'évaluation directe de la distribution de poids de C^1, *i.e.*, le calcul des coefficients $A_1(w,e,\rho)$, nous a permis de trouver le rapport maximal exact du nombre de branches ρ sur le poids w et de valider les valeurs de β trouvées en observant le treillis. Par conséquent, nous prenons $\rho_{\max} = \beta w$.

A l'aide des valeurs maximales de e et de ρ, nous pouvons maintenant majorer le coefficient $A_1(w,e,\rho)$. Il est possible de montrer numériquement que lorsque D tend vers 0, et pour $r = \frac{k}{n} > \frac{1}{2}$ (pour que le rendement du code GLD convolutif ne soit pas nul, le rendement du code constituant doit vérifier $r > 1 - \frac{1}{2}$), nous avons l'inégalité :

$$A_1(w,e,\rho) \leq C_{w-1}^{\lceil \frac{w}{d_1} \rceil - 1} C_{\lceil \beta w \rceil - 1}^{\lceil \frac{w}{d_1} \rceil - 1} T(D)^{\frac{w}{d_1}} D^{-w} \ . \tag{3.13}$$

La figure 3.18 compare la valeur maximale du coefficient $A_1(w,e,\rho)$ à la borne supérieure de l'inégalité (3.13) en fonction du poids w, pour le code RSC de rendement $\frac{3}{4}$ et de polynômes générateurs $(31, 23, 27, 37)$.

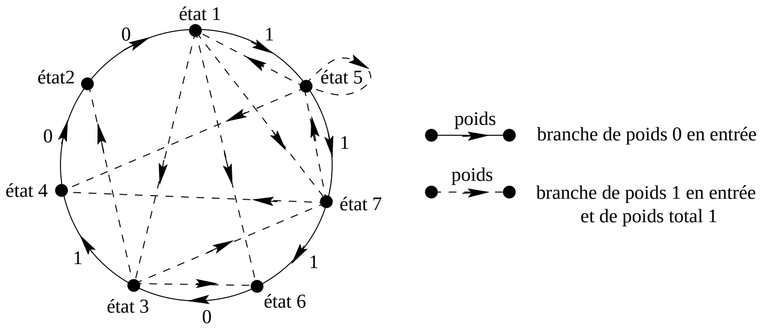


Fig. 3.14 – Succession des états pour le code C_{ex}^{RSC}

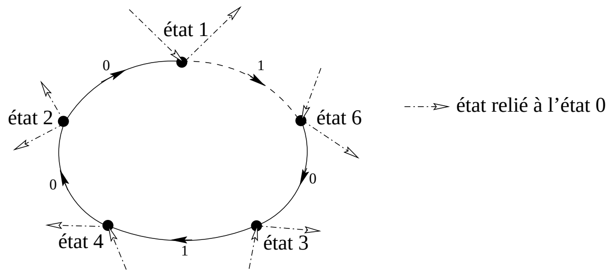


Fig. 3.15 – Cycle maximisant le rapport $\frac{\rho}{w}$ pour le code $(15, 16, 13, 17)$

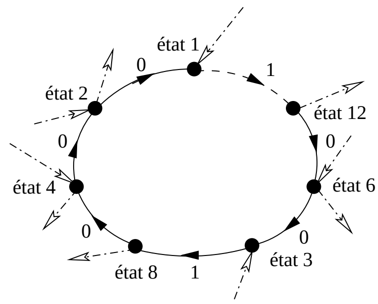


Fig. 3.16 – Cycle maximisant le rapport $\frac{\rho}{w}$ pour le code $(31, 23, 27, 37)$

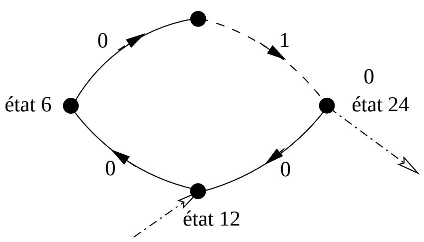


Fig. 3.17 – Cycle maximisant le rapport $\frac{\rho}{w}$ pour le code $(51, 63, 73, 57)$

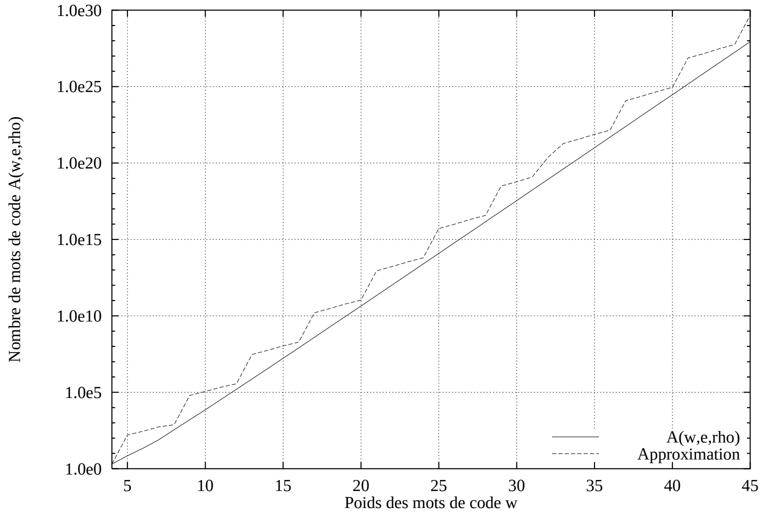


FIG. 3.18 – Borne supérieure du coefficient $A_1(w,e,\rho)$ en fonction du poids w

2. Borne supérieure du coefficient $A_1(w)$:

En combinant les inégalités (3.8) et (3.13), nous obtenons :

$$A_1(w) \leq C_{w-1}^{\lceil \frac{w}{d_1} \rceil - 1} C_{\lceil \beta w \rceil - 1}^{\lceil \frac{w}{d_1} \rceil - 1} T(D)^{\frac{w}{d_1}} D^{-w} \sum_{e=1}^{\lceil \frac{w}{d_1} \rceil} \sum_{\rho=1}^{\lceil \beta w \rceil} C_{\frac{N}{n}-\rho+e}^{e} \ . \tag{3.14}$$

Par ailleurs, un événement d'erreurs est constitué d'au moins deux branches. Ainsi, e événements d'erreurs sont constitués d'au moins $2e$ branches. Or, dans notre cas, e événements d'erreurs représentent ρ branches, d'où :

$$2e \leq \rho \Leftrightarrow e \leq \rho - e \ .$$

Par conséquent,

$$C_{\frac{N}{n}-\rho+e}^{e} \leq C_{\frac{N}{n}-e}^{e} \ .$$

Pour pouvoir majorer le coefficient binômial $C_{\frac{N}{n}-e}^{e}$, nous devons analyser le comportement de la fonction $h(x) = C_{p-x}^{x}$ pour x et p entiers positifs, et p fixé.
Posons $u = \frac{h(x+1)}{h(x)}$. Par définition,

$$u = \frac{(p-x-1)! \ x! \ (p-2x)!}{(p-x)! \ (x+1)! \ (p-2(x+1))!} \ ,$$

soit,

$$u = \frac{(p-2x)(p-2x-1)}{(x+1)(p-x)} \ .$$

La fonction h est croissante si $u \geq 1$, ce qui équivaut à résoudre l'inégalité :

$$5x^2 + (3-5p)x + p(p-2) \geq 0 \ .$$

Le discriminant de l'équation $5x^2 + (3-5p)x + p(p-2) = 0$ est égal à $5p^2 + 10p + 9$ et est strictement positif ; les solutions de l'équation sont :

$$\begin{aligned} x_1(p) &= \frac{-(3-5p) - \sqrt{5p^2+10p+9}}{10} \geq 0 \quad \forall\, p \geq 2 \ , \\ x_2(p) &= \frac{-(3-5p) + \sqrt{5p^2+10p+9}}{10} \geq 0 \quad \forall\, p \geq 0 \ . \end{aligned}$$

La fonction $h(x)$ est une fonction croissante pour $x \in ([0, x_1(p)] \cup [x_2(p), +\infty[) \cap \mathbb{N}$. Dans notre cas, x est égal à e, le nombre d'événements d'erreurs pour un poids total w, et $p = \frac{N}{n}$ $(p \geq 2)$. Nous avons vu que le nombre d'événements d'erreurs e est majoré par $\frac{w}{d_1}$. De plus, nous nous intéressons à des poids w proches de la distance minimale du code GLD. Cette distance minimale est au mieux égale à la valeur de la distance δN donnée par la borne de Gilbert Varshamov pour les codes binaires, soit 0.11 N dans le cas d'un code de rendement 1/2 ou 0.1739 N dans le cas d'un code de rendement 1/3. Examinons dans ce cas la position de $x_1\left(\frac{N}{n}\right)$ et $x_2\left(\frac{N}{n}\right)$ par rapport à $\frac{\delta N}{d_1}$. Nous pouvons aisément démontré que la valeur minimale de N pour laquelle

$$e \leq \frac{\delta N}{d_1} \leq x_1\left(\frac{N}{n}\right) \leq x_2\left(\frac{N}{n}\right)$$

est donnée par :

$$N_{\min} = \frac{(2d_1 - 3\delta n)nd_1}{d_1^2 - 5\delta n d_1 + 5n^2\delta^2}$$

lorsque $d_1 \geq \frac{(5+\sqrt{5})\delta n}{2}$. Par exemple, si on considère le code GLD construit à partir du code C_{ex}^{RSC}, tel que $n = 4$, $d_1 = 4$ et $\delta = 0.11$ donné par la borne de Gilbert Varshamov, la valeur minimale de N est 14.

Par conséquent, nous nous plaçons dans l'intervalle où la fonction h est croissante, donc $h(e) \leq h(\lceil \frac{w}{d_1} \rceil)$, soit :

$$C_{N-e}^{e} \leq C_{N-\lceil \frac{w}{d_1} \rceil}^{\lceil \frac{w}{d_1} \rceil} \ .$$

En remarquant de plus que $C_{w-1}^{\lceil \frac{w}{d_1} \rceil -1} \leq C_{w}^{\lceil \frac{w}{d_1} \rceil -1}$ et $C_{\lceil \beta w \rceil -1}^{\lceil \frac{w}{d_1} \rceil -1} \leq C_{\lceil \beta w \rceil}^{\lceil \frac{w}{d_1} \rceil -1}$, l'inégalité (3.14) s'écrit :

$$\begin{aligned} A_1(w) &\leq \left(\left\lceil \frac{w}{d_1} \right\rceil \; \lceil \beta w \rceil \right) C_{w}^{\lceil \frac{w}{d_1} \rceil -1} C_{\lceil \beta w \rceil}^{\lceil \frac{w}{d_1} \rceil -1} T(D)^{\frac{w}{d_1}} D^{-w} C_{\frac{N}{n} - \lceil \frac{w}{d_1} \rceil}^{\lceil \frac{w}{d_1} \rceil} \\ &\leq \left(N - 2\left\lceil \frac{w}{d_1} \right\rceil + 1 \right) \lceil \beta w \rceil C_{w}^{\lceil \frac{w}{d_1} \rceil -1} C_{\lceil \beta w \rceil}^{\lceil \frac{w}{d_1} \rceil -1} T(D)^{\frac{w}{d_1}} D^{-w} C_{\frac{N}{n} - \lceil \frac{w}{d_1} \rceil}^{\lceil \frac{w}{d_1} \rceil -1} \ . \end{aligned} \tag{3.15}$$

3. Borne supérieure du coefficient $A(w)$:

La borne de Stirling sur les factorielles conduit à :

$$\frac{1}{\sqrt{2\pi N\lambda(1-\lambda)}} \exp\left(NH(\lambda) - \frac{1}{12N\lambda(1-\lambda)} \right) \leq C_N^{N\lambda} \leq \frac{1}{\sqrt{2\pi N\lambda(1-\lambda)}} \exp\left(NH(\lambda)\right) \ ,$$

avec $\lambda = \frac{w}{N}$ et H est la fonction entropie naturelle.

En combinant l'approximation ci-dessus avec l'inégalité (3.15) et le résultat du théorème 3.3.1, nous obtenons que le coefficient $A(w)$ est majoré par :

$$A(w) \leq C(N,\lambda)\exp(-Nf(\lambda)) \ , \tag{3.16}$$

avec

$$C(N,\lambda) = \frac{\exp\left(\frac{J-1}{12N\lambda(1-\lambda)}\right)\left(\sqrt{2\pi N\lambda(1-\lambda)}\right)^{J-1}((N+1)(\beta\lambda N+1))^{J}}{\left(\sqrt{(2\pi)^3(\frac{\lambda N}{d_1}-1)^3\left(1-\frac{1}{d_1}\right)\left(1-\frac{1}{\beta d_1}\right)\left(1-\frac{\lambda}{\frac{d_1}{n}-\frac{\lambda}{d_1}+\frac{1}{N}}\right)}\right)^{J}}$$

et

$$\begin{aligned} f(\lambda) \ &= \ (J-1)H(\lambda) - J\left(\left(\frac{1}{n}-\frac{1}{N}\left\lceil\frac{N\lambda}{d_1}\right\rceil\right)H\left(\frac{\lceil\frac{\lambda N}{d_1}\rceil-1}{\frac{N}{n}-\lceil\frac{\lambda N}{d_1}\rceil}\right)+\lambda H\left(\frac{\lceil\frac{\lambda N}{d_1}\rceil-1}{\lambda N}\right)+\right.\\ &\left.\frac{\lceil\beta\lambda N\rceil}{N}H\left(\frac{\lceil\frac{\lambda N}{d_1}\rceil-1}{\lceil\beta\lambda N\rceil}\right)+\frac{\lambda}{d_1}\ \ln\left(\frac{T(D)}{D^{d_1}}\right)\right) \ . \end{aligned} \tag{3.17}$$

Dans l'expression de $f(\lambda)$, toutes les quantités x dont on considère l'entropie $H(x)$ vérifient $0 \leq x \leq 1/2$. Il est donc possible de conserver l'inégalité (3.16) en considérant un majorant de x inférieur à $\frac{1}{2}$. Ainsi :

$$H\left(\frac{\lceil\frac{\lambda N}{d_1}\rceil-1}{\lambda N}\right)\leq H\left(\frac{1}{d_1}\right), \quad H\left(\frac{\lceil\frac{\lambda N}{d_1}\rceil-1}{\lceil\beta\lambda N\rceil}\right)\leq H\left(\frac{1}{\beta d_1}\right) \ \textit{et}\ H\left(\frac{\lceil\frac{\lambda N}{d_1}\rceil-1}{\frac{N}{n}-\lceil\frac{\lambda N}{d_1}\rceil}\right)\leq H\left(\frac{\lambda}{\frac{d_1}{n}-\lambda}\right) \ .$$

Par ailleurs, lorsque $D \to 0$, $\ln\left(\frac{T(D)}{D^{d_1}}\right) \sim a_{d_1}$, où a_{d_1} est le nombre de chemins de poids d_1 dans le diagramme d'états de C_c^1. Nous obtenons ainsi la fonction B donnée par :

$$B(\lambda) = (J-1)H(\lambda) - J\left(\left(\frac{1}{n}-\frac{\lambda}{d_1}\right)H\left(\frac{\lambda}{\frac{d_1}{n}-\lambda}\right)+\lambda H\left(\frac{1}{d_1}\right)+\beta\lambda H\left(\frac{1}{\beta d_1}\right)+\frac{\lambda}{d_1}\ \ln(a_{d_1})\right)$$

et telle que

$$A(w) \leq C(N,\lambda)\exp(-NB(\lambda)) \ . \tag{3.18}$$

<u>4. Étude de la fonction B et évaluation de la distance δ</u> :

Examinons l'allure de la fonction B. Asymptotiquement lorsque N tend vers l'infini, le nombre de mots de code $A(w)$ de poids w tend vers 0 si $B(\lambda)$ est strictement positive.
On peut voir aisément que $B(\lambda = 0) = 0$ et lorsque λ tend vers 0^+, $B(\lambda)$ tend vers 0^+. En effet, un équivalent de $B(\lambda)$ en 0^+ est :

$$B(\lambda) \sim -\left(J\frac{d_1-1}{d_1}-1\right)\lambda\ln(\lambda) \ ,$$

avec $d_1 \geq 2$ et $J \geq 2$. La dérivée de la fonction B s'écrit au voisinage de 0 :

$$B'(\lambda) \sim -\left(J\frac{d_1-1}{d_1}-1\right)\ln\lambda \ .$$

Cette dérivée tend vers $+\infty$ lorsque $\lambda \to 0$. Ainsi, $B(\lambda)$ est positive au voisinage de 0.

Par ailleurs, la dérivée seconde de la fonction B est donnée par :

$$B''(\lambda) = \frac{Jd_1(1-\lambda) - (J-1)(d_1 - n\lambda)(d_1 - 2n\lambda)}{\lambda(1-\lambda)(d_1 - n\lambda)(d_1 - 2n\lambda)} \; .$$

Le dénominateur de $B''(\lambda)$ est positif pour les valeurs de λ considérées (λ inférieur à λ_0 donné par la borne de Gilbert Varshamov). Le numérateur $num(x)$ est une équation du second degré en λ majoré par :

$$num(x) \leq Jd_1 - (J-1)(d_1 - n\;\lambda_0)(d_1 - 2n\;\lambda_0) \; .$$

Le tableau 3.3 donne la condition que doit vérifier d_1 pour que la partie droite de l'inégalité soit négative, pour différents rendements de C_c^1 et du code GLD convolutif correspondant.

J	n	$R_{C_c^1}$	R_{GLD}	λ_0	d_1
2	3	2/3	1/3	0.1739	≥ 4
2	4	3/4	1/2	0.11	≥ 3
3	4	3/4	1/4	0.2145	≥ 5

Tab. 3.3 – Valeurs de la distance minimale d_1

Pour ces valeurs de la distance minimale[3] d_1, la dérivée seconde de la fonction B est nécessairement négative pour $\lambda \leq \lambda_0$. Par conséquent, cela signifie que la fonction B est concave. Nous recherchons la valeur (si elle existe !) de $0 < \delta < \lambda_0$ telle que $B(\lambda)$ soit positive pour $0 \leq \lambda \leq \delta$. La figure 3.19 donne l'allure de la fonction B pour différents codes GLD convolutifs tels que $J = 2$ et construits à partir de codes RSC de rendement $R = \frac{3}{4}$ dont les paramètres sont les suivants :

- GLD_1 : C_c^1 est de polynômes générateurs $(15, 16, 13, 17)$ (8 états) : $\beta = \frac{5}{2}$, $d_1 = 4$, $a_{d_1} = 10$
- GLD_2 : C_c^1 est de polynômes générateurs $(31, 23, 27, 37)$ (16 états) : $\beta = \frac{7}{2}$, $d_1 = 4$, $a_{d_1} = 2$
- GLD_3 : C_c^1 est de polynômes générateurs $(51, 63, 73, 57)$ (64 états) : $\beta = 4$, $d_1 = 5$, $a_{d_1} = 7$

La valeur de δ peut être déterminée sur cette figure.

c.q.f.d.

Ce théorème indique que le nombre de mots de code de poids w inférieur à δN tend vers 0 lorsque la longueur N du code tend vers l'infini ; lorsque $w < \delta N$, la fonction $B(\lambda)$ est en effet positive. Nous obtenons donc une borne inférieure de la distance minimale moyenne du code GLD convolutif.

Plus que la valeur exacte de cette borne, il est intéressant de noter qu'il s'agit d'une borne linéaire en N, ce qui signifie que la distance minimale moyenne d'un code GLD convolutif croît linéairement avec sa longueur. Les codes GLD convolutifs sont par conséquent asymptotiquement

[3] pour des valeurs inférieures de la distance d_1, il est nécessairement d'étudier plus en détails le signe de $B''(\lambda)$.

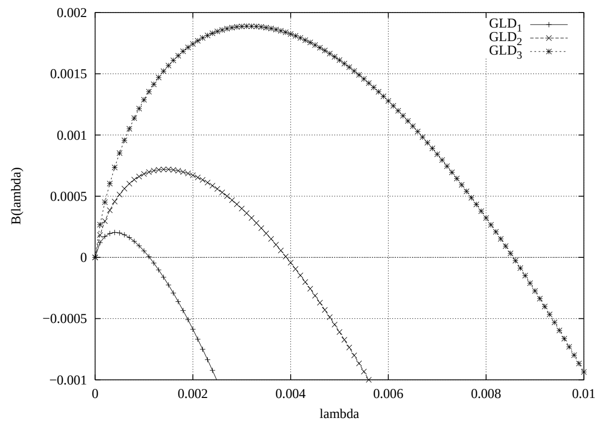


FIG. 3.19 – Allure de la fonction B pour différents codes constituants, et $J = 2$

bons pour $J \geq 2$. Ceci est en accord avec la borne supérieure de la distance minimale moyenne évaluée dans la partie 3.3.1, qui elle-même présentait un comportement linéaire en fonction de la longueur N. Cependant, les valeurs de δ trouvées sont relativement faibles par rapport aux valeurs de la distance Δ normalisée, notée Δ_{nm}. Ainsi pour comparaison, nous avons :

- pour le code GLD, construit à partir de $J = 2$ codes RSC de rendement $r = \frac{3}{4}$ et de polynômes générateurs $(15, 16, 13, 17)$:

$$\delta = 0.0012 \quad \text{et} \quad \Delta_{nm} = 0.063 \; .$$

- pour le code GLD, construit à partir de $J = 2$ codes RSC de rendement $r = \frac{3}{4}$ et de polynômes générateurs $(31, 23, 27, 37)$:

$$\delta = 0.0041 \quad \text{et} \quad \Delta_{nm} = 0.074 \; .$$

Cette différence peut s'expliquer par le manque de précisions des majorations utilisées pour borner le nombre $A(w)$ de mots de code de poids w.

Le tableau 3.4 donne quelques valeurs de δ pour différents codes constituants de rendement $\frac{3}{4}$ et différentes valeurs de J.

J	Nombre d'états	Polynômes générateurs	δ
2	8	(15,16,13,17)	0.0012
2	16	(31,23,27,37)	0.0041
2	32	(51,63,73,57)	0.0086
3	8	(15,16,13,17)	0.0252
3	16	(31,23,27,37)	0.0539
3	32	(51,63,73,57)	0.0661

TAB. 3.4 – Borne inférieure de la distance minimale moyenne d'un code GLD convolutif.

Nous avons étudié les performances asymptotiques des codes GLD convolutifs avec décodage ML. Cependant, comme la plupart des codes composites, ces codes ne peuvent être décodés simplement de manière optimale. Du fait de leur structure, *i.e.*, intersection de codes convolutifs, ils peuvent être facilement décodés par un processus itératif.

3.4 Décodage itératif

Dans [42], Gallager a présenté un algorithme relativement simple pour décoder de manière itérative les codes LDPC. Cet algorithme peut être considéré comme l'ancêtre du décodage "turbo" [11]. En effet, il calcule itérativement la probabilité de chaque bit codé sachant l'ensemble des observations du canal, *i.e.*, la probabilité *a posteriori* de chaque bit codé. De plus, il exploite la faible densité de la matrice de parité du code LDPC pour réduire la complexité de décodage lors du calcul des probabilités *a posteriori*. Tanner [76] a également donné plusieurs versions d'un algorithme de décodage itératif pouvant être appliqué à la construction déterministe de codes composites. Cependant, malgré une structure très proche de celle des LDPC, les codes GLD convolutifs ne peuvent être décodés à l'aide de l'algorithme introduit par Gallager. En effet, comme nous l'avons vu précédemment, les équations de parité de la matrice H du code GLD convolutif ne sont plus disjointes, ce qui rend inefficace le décodage de Gallager qui est d'autant plus performant que les équations de parité sont indépendantes. Nous proposons de décoder les codes GLD convolutifs en utilisant le fait qu'ils sont construits à partir de J codes convolutifs entrelacés aléatoirement, c'est-à-dire, à partir de J codes convolutifs indépendants.

3.4.1 Principe du décodage SISO binaire

La structure des codes GLD convolutifs encourage naturellement l'utilisation d'une procédure de décodage itératif fondée sur le décodage successif de chacun de ses codes constituants.

Décrivons brièvement le décodage d'un code GLD convolutif $\mathcal{C}$ constitué d'un entrelaceur π reliant $J = 2$ codes réalisés à partir du code convolutif C^1. Nous appelons le code C^1 code supérieur et le code $C^2 = \pi(C^1)$ code inférieur. Si $J > 2$, la stratégie de propagation de l'information d'un code à l'autre a une influence importante sur les performances des GLD convolutifs, comme dans le cas des turbo-codes : l'entrée du décodeur i, $i = 1, \ldots, J$, doit être la combinaison des informations souples obtenues en sortie des $J-1$ autres décodeurs.

La structure du décodeur d'un code GLD convolutif est similaire à celle du décodeur utilisé pour les turbo-codes (*cf.* figure 2.12). Le décodeur convertit la sortie du canal $\mathbf{y}$ en une probabilité conditionnelle $\Pr(\mathbf{y}|\mathbf{x})$, où $\mathbf{x}$ représente le mot de code $\mathbf{c}$ du code GLD $\mathcal{C}$ modulé par une BPSK. Les informations *a priori* $P(c_i)$ sur les composantes du mot de code $\mathbf{c}$ sont initialisées à 0.5 pour le décodeur du code supérieur, à la première itération. Une information *a priori* est disponible pour chaque bit codé, puisque chaque bit codé appartient à deux codes via l'entrelacement ; dans le cas des turbo-codes parallèle par exemple, seule une information *a priori* sur les bits d'information était nécessaire, car les bits de parité n'étaient pas communs aux deux codes constituants. Les deux décodeurs élémentaires sont des décodeurs à entrées et sorties souples utilisant par exemple l'algorithme aller-retour décrit au chapitre 2. A la sortie de chaque décodeur se trouvent donc une information *a posteriori* et une information extrinsèque sur chaque bit codé. Cette information extrinsèque est entrelacée (respectivement désentrelacée) et fournie comme information *a priori* au décodeur du code inférieur (respectivement supérieur). Puis le second décodeur génère à son tour, à partir de l'observation du canal et des informations *a priori* disponibles, une information extrinsèque et une probabilité *a posteriori* sur chaque bit codé. L'information extrinsèque est alors donnée au premier décodeur. L'ensemble des deux décodages constitue une itération.

L'entrelacement employé jusqu'à présent se fait bit à bit. Nous proposons dans le paragraphe suivant de regrouper les bits en symboles et de considérer un entrelacement symbole par symbole, afin d'améliorer les performances du décodage itératif des codes GLD convolutifs.

3.4.2 Entrelacement et décodage par symbole

Soit $\mathcal{C}$ un code GLD convolutif construit à partir de $J = 2$ codes constituants C^1 et $\pi(C^1)$ (C^1 est obtenu par fermeture du treillis du code convolutif $C_c^1(n,k)$). La permutation aléatoire π envisagée dans cette section entrelace les bits deux par deux. L'entrelaceur du code GLD convolutif permute donc des symboles quaternaires au lieu de permuter des bits. Les codes convolutifs considérés restent cependant des codes convolutifs binaires.

Soit $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ un mot de code de $\mathcal{C}$. Les bits codés sont modulés par une BPSK pour donner la séquence $\mathbf{x} = (x_1, x_2, \ldots, x_N)$. Cette séquence est émise sur un canal AWGN, et $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ est la séquence reçue à l'entrée du décodeur. Les bits c_i, $i = 1, \ldots, N$ ont été entrelacés deux par deux, et peuvent être ainsi regroupés en symboles quaternaires $S_i = (c_{2i-1}\ c_{2i}) \in \{0,1,2,3\}$ pour $i = 1, \ldots, \frac{N}{2}$. Dans le treillis de chaque code convolutif ceci revient à considérer sur chaque branche $\frac{n}{2}$ symboles quaternaires si n est pair, ou à regrouper deux branches du treillis en une seule pour obtenir n symboles quaternaires si n est impair.

Chaque symbole peut être constitué indifféremment de deux bits d'information, de deux bits de parité, ou encore d'un bit de parité et d'un bit d'information, puisque tous les bits codés sont entrelacés et protégés de manière identique. Les séquences $\mathbf{x}$ et $\mathbf{y}$ s'écrivent alors sous la forme :

$$\mathbf{X} = (X_1, \ldots, X_{\frac{N}{2}}) \text{ avec } (X_i = \{x_{2i-1}\ x_{2i}\}) \in \{0,1,2,3\} \quad \forall\, i = 1, \ldots, \frac{N}{2}$$

et

$$\mathbf{Y} = (Y_1, \ldots, Y_{\frac{N}{2}}) \text{ avec } (Y_i = \{y_{2i-1}\ y_{2i}\}) \in \{0,1,2,3\} \quad \forall\, i = 1, \ldots, \frac{N}{2} \ .$$

L'entrelacement se faisant symbole par symbole, les probabilités *a priori* de deux bits du même symbole ne peuvent plus être considérées comme indépendantes à l'entrée du décodeur. Il faut donc adapter le décodeur opérant sur des bits en un décodeur opérant sur des symboles quaternaires. Les probabilités *a posteriori* générées sur les symboles au cours du décodage sont ensuite transformées en probabilités *a posteriori* sur les bits, afin de prendre une décision à la dernière itération.

Pour un canal discret sans mémoire, les observations sur les bits sont indépendantes à la sortie du canal. L'observation de chaque symbole X_i s'écrit donc :

$$\Pr(Y_i|X_i) = \Pr(y_{2i-1}|x_{2i-1})\ \Pr(y_{2i}|x_{2i}) \ \ \forall\, i \ .$$

Les probabilités *a priori* utilisées par le décodeur sont maintenant des probabilités *a priori* sur les symboles S_i et sont ainsi initialisées à $\frac{1}{4}$ à la première itération, puisqu'il existe 4 valeurs possibles pour S_i. Le décodeur utilise alors les informations fournies par le canal et les informations *a priori* sur les symboles pour générer une information extrinsèque et une probabilité *a posteriori* sur les symboles S_i. L'algorithme de décodage (algorithme aller-retour) reste identique à celui utilisé dans le cas binaire, si ce n'est qu'il observe maintenant des symboles sur les branches du treillis et traite des probabilités correspondant à des symboles au lieu de bits.

En sortie du décodeur, l'information extrinsèque sur les symboles est (dés)entrelacée et fournie au décodeur suivant en tant qu'information *a priori* sur les symboles. L'information *a posteriori* générée, c'est-à-dire l'APP, est également une information portant sur les symboles quaternaires. Cette APP doit être modifiée afin de pouvoir prendre une décision sur chaque bit formant le symbole. Pour cela nous écrivons :

$$APP(c_i) = \sum_{S \in \{0,1,2,3\} | c_i} APP(S) \quad i = 1, \ldots, N.$$

où $APP(S)$ est l'APP relative au symbole S.

Les performances obtenues à l'aide de cet entrelaceur sont présentées dans la section suivante, et sont comparées aux performances obtenues avec un entrelaceur binaire.

3.5 Résultats Numériques

Nous supposons dans cette section que le code GLD considéré est construit à partir de $J = 2$ codes convolutifs identiques à une permutation près, et de rendement $r = \frac{3}{4}$. Le rendement du code GLD résultant est $R = \frac{1}{2}$. La modulation est binaire (BPSK), et le canal est un canal AWGN.

La figure 3.20 illustre les performances, en termes de taux d'erreur binaire, d'un code GLD de longueur $N = 800$ construit à partir d'un code RSC de rendement $r = \frac{1}{2}$ et de polynômes générateurs $(23, 35)$. Ce code RSC est perforé afin d'obtenir un code de rendement $r = \frac{3}{4}$. Une probabilité d'erreur par bit $Pe_b = 10^{-5}$ est atteinte pour un rapport signal-à-bruit $\frac{E_b}{N_0} = 2.9$ dB après 16 itérations de décodage. Le passage de 8 à 16 itérations apporte un gain négligeable, de l'ordre de 0.05 dB, pour une complexité de décodage doublée.

La figure 3.21 montre l'évolution de ces performances lorsque la longueur des codes GLD devient $N = 2000$. Rappelons que la longueur du code GLD est égale à la taille de l'entrelaceur. Le code utilisé est le même que celui considéré sur la figure 3.20. Une probabilité d'erreur par bit $Pe_b = 10^{-5}$ est maintenant atteinte pour un rapport signal-à-bruit $\frac{E_b}{N_0} = 2.4$ dB après 16 itérations. Le code de longueur $N = 2000$ présente en fait un gain de 0.5 dB par rapport aux performances atteintes par le code de longueur $N = 800$.

Dans l'étude des performances analytiques réalisée à la section 3.3, aucun gain d'entrelacement n'a pu être mis en évidence pour les codes GLD convolutifs. Il en est de même pour les codes GLD blocs [62] ou les codes LDPC [42]. Ceci explique le faible gain réalisé lorsque la longueur du code est augmentée. Ainsi la figure 3.22 illustre les performances du même code GLD lorsque la longueur est cette fois-ci $N = 20000$. Le gain est de 0.5 dB pour une probabilité d'erreur $Pe_b = 10^{-5}$ après 16 itérations.
En revanche, nous constatons sur les figures 3.20, 3.21, 3.22, que la «pente» de la probabilité

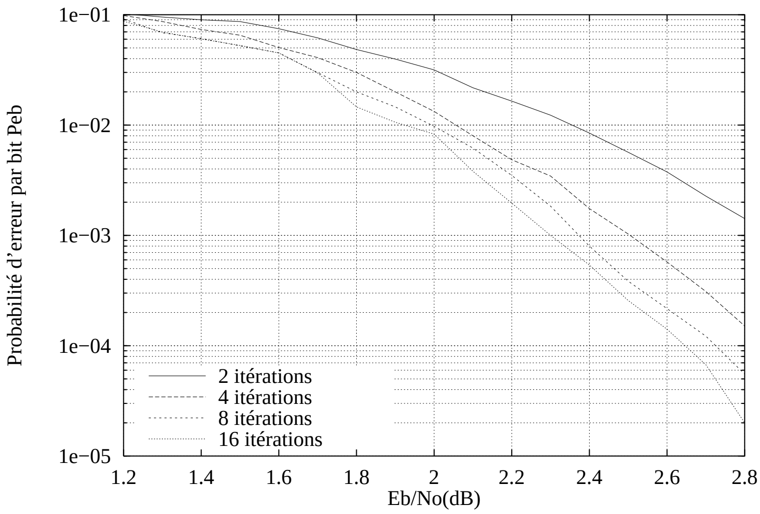


FIG. 3.20 – Probabilité d'erreur par bit d'un code GLD de longueur $N = 800$

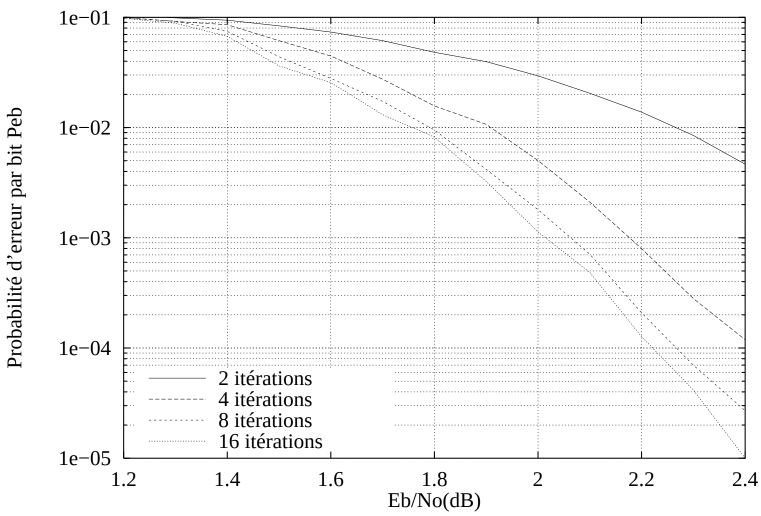


FIG. 3.21 – Probabilité d'erreur par bit d'un code GLD de longueur $N = 2000$

d'erreur par bit évolue en fonction de la longueur du code N. Ainsi, pour passer d'une probabilité d'erreur $Pe_b = 10^{-2}$ à une probabilité d'erreur $Pe_b = 10^{-4}$, il faut :

- 0.75 dB, pour le code de longueur $N = 800$,
- 0.5 dB, pour le code de longueur $N = 2000$,
- 0.05 dB, pour le code de longueur $N = 20000$.

Par ailleurs, les performances des codes GLD convolutifs ne présentent aucun error-floor dans les probabilités d'erreur observées ($Pe_b = 10^{-5} - 10^{-6}$) du fait de leur grande distance minimale. Ceci est en accord avec l'étude théorique réalisée à la section 3.3 selon laquelle la distance minimale des codes GLD convolutifs est une fonction linéaire de la longueur du code et peut donc être très grande.

Les codes constituants choisis jusqu'à présent ont été soigneusement sélectionnés en opti-

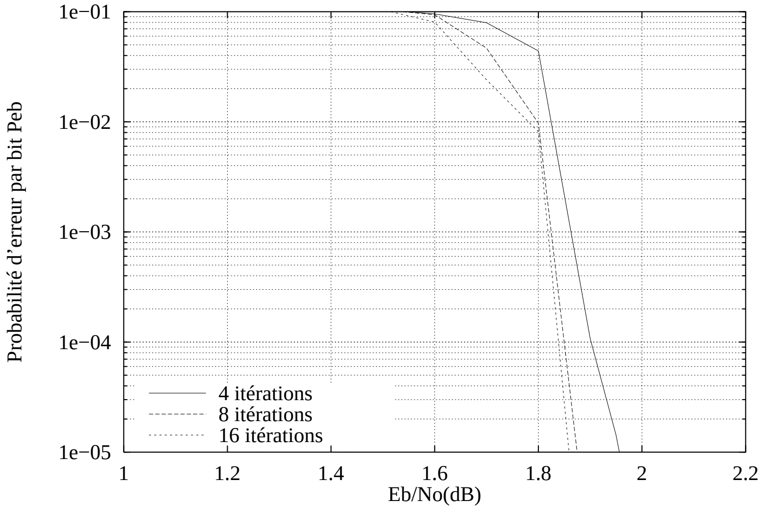


Fig. 3.22 – Probabilité d'erreur par bit d'un code GLD de longueur $N = 20000$

misant leur propre distance minimale et en minimisant le nombre de chemins de poids faibles présents dans leur treillis. Nous avons, en particulier, choisi des codes dont le nombre d'états est supérieur à 8. La figure 3.24 montre les performances d'un code GLD construit à partir d'un code convolutif NRNS de rendement $\frac{3}{4}$ et de polynômes générateurs $(13, 15, 2, 14)_D$ (écrit selon la convention de Daut[4] [26]). Un schéma du codeur de ce code convolutif est donné sur la figure 3.23. Le treillis de ce code contient 2 états ; 8 branches partent et arrivent sur chaque état. Ce treillis

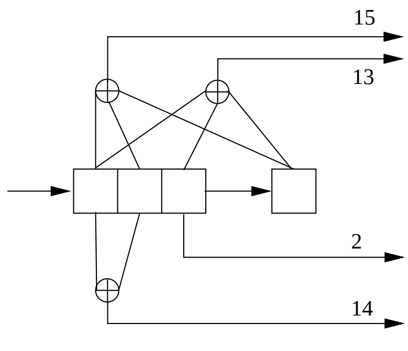


Fig. 3.23 – Codeur du code NRNS de rendement $\frac{3}{4}$ et de polynômes générateurs $(13, 15, 2, 14)_D$

contient ainsi de nombreuses branches parallèles. La longueur du code GLD est $N = 131072$. Il atteint une probabilité d'erreur $Pe_b = 10^{-5}$ à environ 1.03 dB, soit à seulement 0.55 dB de la capacité du canal AWGN après 32 itérations. A titre de comparaison, un turbo-code parallèle de même rendement et de même longueur, construit à partir d'un code convolutif de rendement $r = \frac{1}{2}$ et de polynômes générateurs $(23, 35)$ (*i.e.*, treillis à 16 états, 2 branches partent et arrivent sur chaque état), atteint une probabilité d'erreur $Pe_b = 10^{-5}$ pour un rapport $\frac{E_b}{N_0} = 0.70$ dB environ et 20 itérations de décodage. Le code GLD bénéficie cependant, d'après la section 3.3, d'une distance minimale qui croît linéairement avec la longueur du code pour $J = 2$ codes

[4]nous ajoutons l'indice D aux polynômes générateurs lorsque nous utilisons cette convention

constituants, contrairement aux turbo-codes pour lesquels la distance minimale est indépendante de la longueur N et qui souffrent d'un changement de pente dans leurs performances dû à cette faible distance minimale[5]. L'inconvénient des codes GLD réside malheureusement dans l'absence de schémas de codage simples, ce qui conduit à une complexité prohibitive lorsque la longueur N devient très grande.

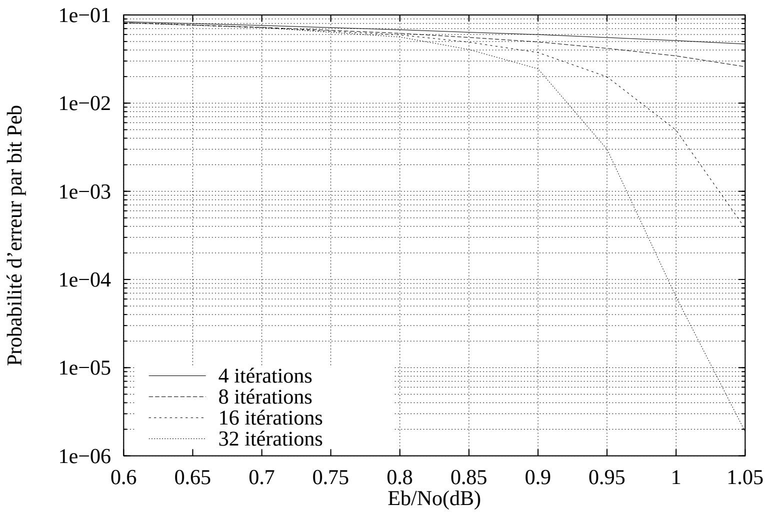


FIG. 3.24 – Probabilité d'erreur par bit d'un code GLD de longueur $N = 131072$

Pour la même longueur $N = 131072$, le code GLD considéré sur les figures 3.20, 3.21 et 3.22, donne des performances de l'ordre de 0.80 dB moins bonnes que les performances du code représenté sur la figure 3.24. Une tentative d'explication pourrait s'appuyer sur la différence de rendements entre le code constituant et le code GLD. Ainsi par exemple, pour obtenir un code GLD de rendement $R = \frac{1}{2}$, il est nécessaire de considérer des codes constituants de rendement $r = \frac{3}{4}$. Or, chaque code constituant est décodé seul en s'aidant des informations souples fournies par le décodeur précédent. Pour un rapport signal-à-bruit $\frac{E_b}{N_0} = 1.0$ dB vu par le code GLD (*i.e.*, rapport signal-à-bruit si un décodage optimisant le critère du maximum de vraisemblance était possible pour le code GLD), le rapport signal-à-bruit vu par le code constituant est :

$$\left(\frac{E_b}{N_0}\right)' = 1.0 - 10\log_{10}\frac{\frac{3}{4}}{\frac{1}{2}} = -0.76 \text{ dB.}$$

Par conséquent, il faut choisir un code convolutif dont les performances sont bonnes (de manière relative) à un tel rapport signal-à-bruit, c'est-à-dire pour des probabilités d'erreur proches de 10^{-1}. Il s'agit la plupart du temps de codes qui ne donnent pas forcément de très bonnes performances à fort rapport signal-à-bruit et à faibles probabilités d'erreur par bit.

Enfin, la figure 3.25 présente les performances d'un code GLD convolutif de longueur $N = 2000$ utilisant un entrelacement quaternaire comme celui décrit dans la section 3.4. Le code convolutif est un code RSC ayant pour polynômes générateurs $(15, 16, 13, 17)$. Pour le décodage, 2, 4, 8 et 16 itérations sont considérées. Sur cette même figure, se trouvent les performances d'un code

[5]il est possible d'éloigner ce changement de pente vers des probabilités d'erreur plus faibles en optimisant le choix de l'entrelaceur [29]

GLD convolutif construit à partir du même code RSC, mais utilisant un entrelacement binaire, et 16 itérations de décodage. Le gain obtenu en considérant un entrelacement quaternaire est de l'ordre de 0.3 dB pour une complexité faiblement accrue par rapport au cas binaire. En fait, il semblerait que l'entrelacement quaternaire a pour effet de diminuer le nombre de cycles présents dans le graphe de Tanner du code GLD convolutif, ce qui permet au décodage itératif d'être plus performant.

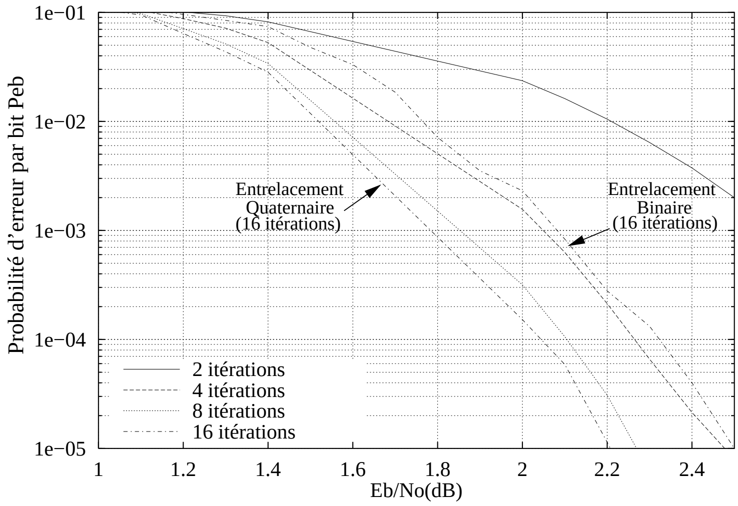


Fig. 3.25 – Influence de l'entrelacement quaternaire sur les performances

3.6 Conclusion

Nous avons introduit dans ce chapitre une construction de Tanner utilisant des codes convolutifs. Cette construction est une généralisation des codes LDPC introduits par Gallager [42] et possède des propriétés de distance similaires. En effet, une analyse des performances analytiques des codes GLD convolutifs lorsqu'un décodage à maximum de vraisemblance est envisagé a été menée et a permis de montrer qu'ils sont asymptotiquement bons, *i.e.*, que leur distance minimale croît linéairement avec leur longueur. De plus, comme pour les codes GLD blocs, cette propriété est valable pour $J = 2$ codes constituants seulement (pour les codes LDPC, $j = 3$ sous-matrices sont nécessaires).

Comme pour l'ensemble des codes composites présentés au chapitre 2, le décodage ML des codes GLD convolutifs s'avère trop complexe pour être réalisé en pratique. Nous avons ainsi considéré un processus de décodage itératif. Par ailleurs, un entrelacement des bits codés deux à deux a montré un gain de 0.3 dB par rapport à un entrelacement bit à bit classique pour une complexité de décodage légèrement accrue. Cet entrelacement par symbole requiert l'adaptation de l'algorithme de décodage élémentaire afin de prendre en compte les informations souples sur les symboles quaternaires entrelacés. A partir de ces informations souples, il est aisé de revenir à des informations de fiabilité sur les bits pour permettre la décision finale.

Nous avons également montré comment les turbo-codes parallèle et série peuvent être interprétés en termes de codes GLD convolutifs. La nature de l'entrelacement des turbo-codes rend l'analyse de leurs performances asymptotiques quelque peu différente de celle réalisée pour les codes GLD et mène à des conclusions distinctes. Ainsi, par exemple, la distance minimale des turbo-codes parallèle ne varie que très peu en fonction de la longueur N du turbo-code par opposition à la croissance linéaire de la distance minimale des codes GLD convolutifs. Inversement, aucun gain d'entrelacement n'a pu être mis en évidence pour les codes GLD convolutifs au cours de l'analyse de leurs performances, contrairement aux turbo-codes.

Chapitre 4

Limites asymptotiques des codes composites décodés itérativement *

Dans les trois premiers chapitres, nous avons considéré les performances analytiques de différents codes composites ou codes optimaux, en supposant que le décodage de ces codes était un décodage à maximum de vraisemblance. Or, comme nous l'avons vu, un tel décodage est difficilement envisageable pour ces codes, du fait de sa trop grande complexité. Ainsi, nous adoptons un processus de décodage itératif fondé sur le décodage élémentaire de chaque code constituant et la propagation d'informations de fiabilité à chaque étape de décodage. Nous proposons dans ce chapitre de mener une étude des performances des codes composites sous l'hypothèse d'un décodage itératif. Nous généralisons ainsi à tout code composite l'approche proposée par [68] pour les codes LDPC.

Tous les codes composites présentés aux chapitres précédents, qu'ils soient construits à partir de codes en blocs ou de codes convolutifs, peuvent être modélisés par un graphe bipartite contenant deux types de sommets représentant les bits codés et les codes constituants (ou de manière équivalente, les contraintes sur les bits, *cf.* chapitre 2). Cette représentation graphique est très utile pour étudier le comportement d'un processus de décodage itératif appliqué à un code composite, lorsque la longueur de l'entrelaceur (ou des entrelaceurs si $J > 2$) est infinie. En effet, le voisinage de chaque sommet du graphe est équivalent à un arbre : l'entrelaceur est supposé suffisamment grand pour éviter les cycles dans le voisinage direct du sommet considéré. L'analyse de ce graphe de Tanner permet de déterminer la limite en termes de performances de ce code lorsqu'il est décodé de manière itérative. Cette limite est définie comme le rapport signal-à-bruit minimal pour lequel la probabilité d'erreur du système tend vers zéro lorsque le nombre d'itérations de décodage tend vers l'infini. La limite ainsi obtenue est à comparer avec la limite donnée par la capacité du canal pour un code de longueur infinie et de même rendement, lorsqu'un décodage à maximum de vraisemblance est mis en œuvre.

L'étude des performances asymptotiques d'un code composite décodé itérativement se fait en observant l'évolution de la probabilité d'erreur au cours du décodage, ce qui requiert la

*Le contenu de ce chapitre a été en partie présenté à l'*International Symposium on Information Theory (ISIT)*, Sorrento, Italie, 25-30 juin 2000.

connaissance de la distribution des probabilités *a posteriori* des bits à chaque itération. Après une brève présentation des hypothèses et des notations, nous introduisons une propriété du code, appelée isotropie, permettant de simplifier l'analyse de la propagation de l'information *a posteriori* dans le graphe. Puis, nous décrivons la propagation des différentes informations souples au cours du décodage itératif et donnons quelques résultats numériques pour les performances asymptotiques des codes composites présentés aux chapitres 2 et 3.

Nous présentons ensuite une méthode graphique permettant d'expliquer le décodage itératif des codes composites, et de trouver une estimation des limites de ces codes. Cette technique est fondée sur l'étude des fonctions de transfert des deux types de sommets du graphe, c'est-à-dire sur l'étude des modifications engendrées sur la probabilité d'erreur lors du passage des informations souples par chaque sommet du graphe. En représentant sur la même figure la fonction de transfert d'un sommet bit et celle d'un sommet code, il est possible d'analyser le comportement du décodage itératif d'un ensemble de codes composites. Nous appelons cette technique la *méthode de la fonction de transfert*. Indépendamment, deux méthodes équivalentes ont été développées par [78] et [45]. Elles sont respectivement fondées sur l'observation de l'évolution de l'information mutuelle et du rapport signal-à-bruit de l'information extrinsèque au cours des itérations. Nous utilisons la méthode de la fonction de transfert pour envisager des voies d'amélioration des performances des turbo-codes parallèle, par exemple en considérant des schémas de codage irréguliers. Dans une dernière partie, nous optimisons les profils de ces schémas, *i.e.*, les degrés d'irrégularité et leurs probabilités associées, grâce à l'étude de la propagation de l'information dans le graphe irrégulier et à l'évaluation des limites atteintes par ces profils.

4.1 Hypothèses et Notations

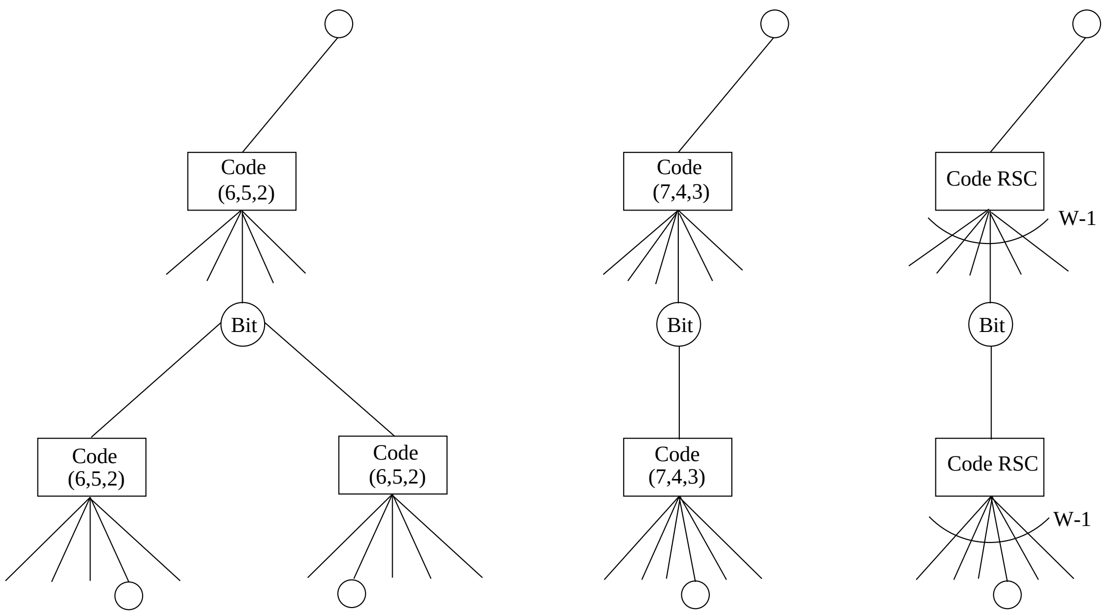
Nous nous restreignons dans ce chapitre, sauf mention contraire, à l'étude de codes réguliers, c'est-à-dire à l'étude de codes composites pouvant être représentés par un graphe bipartite régulier : chaque sommet de même type a alors le même degré dans le graphe. Nous appelons d_b le degré d'un sommet bit et d_c le degré d'un sommet code. Par la suite, lorsqu'il sera fait référence à un sommet bit ou à un sommet code dans le graphe, nous omettrons le terme "sommet" par souci de simplicité. Un exemple de code composite irrégulier est traité en dernière partie de ce chapitre comme une voie d'amélioration des performances des turbo-codes parallèle.

Soit $\mathcal{C}(N,K)$ un code composite construit à partir de J codes constituants $C^1, C^2, \dots, C^J$ reliés par $J-1$ permutations $\pi_2, \dots, \pi_J$. Rappelons que ces codes sont à une permutation près :
- la somme directe d'équations de parité pour les codes LDPC ($J = j$),
- la somme directe de codes en blocs linéaires $C_0(n_0, k_0)$ pour les codes GLD blocs et les codes produits,
- des codes convolutifs, RSC ou non, pour les turbo-codes série et parallèle et pour les codes GLD convolutifs.

Le graphe du code composite étant supposé régulier, les codes constituants sont identiques, à une permutation près, par exemple au code $C(n,k)$. Nous notons $\mathbf{c} = (c_1, \dots, c_n)$ un mot de code de C. Ce mot de code est modulé en une séquence $\mathbf{x} = (x_1, \dots, x_n)$ qui est alors émise sur un canal DMC. La sortie du canal est notée $\mathbf{y} = (y_1, \dots, y_n)$.

Afin d'analyser la propagation de l'information au cours des itérations de décodage, nous considérons la représentation graphique de Tanner du code composite $\mathcal{C}$. Nous supposons que le voisinage de chaque sommet du graphe est un arbre, c'est-à-dire qu'il ne contient pas de cycle. Pour ceci, il suffit de considérer un entrelaceur dont la longueur tend vers l'infini. Plaçons-nous au voisinage d'un sommet et observons son environnement local. La figure 4.1 montre ce voisinage pour trois types de codes de longueur N :

- **un code LDPC de paramètres** $(N, j=3, \ell=6)$ **:**
  les bits sont de degré 3 et les codes sont de degré 6. Les contraintes sont des codes de parité $(6,5,2)$. Le rendement total de ce code est $R=\frac{1}{2}$.
- **un code GLD bloc à 2 niveaux** $(J=2)$ **construit à partir du code de Hamming** $C_0(n_0=7, k_0=4)$ **:**
  les bits sont de degré 2, et les codes sont de degré 7. Chaque code est un code de Hamming C_0. Le rendement total de ce code est $R=\frac{1}{7}$.
- **un turbo-code parallèle construit à partir de deux codes RSC de rendement** $\frac{1}{2}$ **:**
  chaque bit d'information est de degré 2 car il appartient à deux codes RSC. Les bits de parité n'appartiennent, quant à eux, qu'à un seul code RSC, et ne sont par conséquent que de degré 1. Ces bits n'interviennent pas dans la propagation de l'information.
  L'information souple se propageant dans un code convolutif subit des atténuations [13] : nous pouvons définir une fenêtre de dépendance autour de chaque bit codé au-delà de laquelle les bits n'ont plus d'influence sur les informations utilisées par le bit codé considéré. Cette fenêtre contient W bits et définit le degré d'un code. Notons que dans le cas d'un code en blocs $C_0(n_0, k_0)$, W est égale à sa longueur n_0.



(a) *Code LDPC* (b) *Code GLD* (c) *Turbo-code*

Fig. 4.1 – Représentation graphique des codes composites

Les performances asymptotiques d'un code composite sont étudiées à partir d'un ensemble de codes et l'ensemble des graphes associés, construits en considérant toutes les permutations aléatoires possibles permettant de lier les sommets du graphe du code composite entre eux. Les permutations sont équiprobables. Cet ensemble de codes est caractérisé par certains paramètres fixés, comme les nombres j et ℓ pour les codes LDPC, ou encore les polynômes générateurs et le rendement du code constituant pour les turbo-codes parallèle. Ces différents paramètres définissent les degrés des sommets dans le graphe de Tanner du code composite.

Nous limitons notre étude aux codes binaires, modulés par une BPSK, dont les symboles sont émis sur un canal AWGN discret sans mémoire. Le bruit est une variable aléatoire gaussienne de moyenne nulle et de variance $\sigma^2 = N_0$. Les symboles prennent les valeurs $\pm\sqrt{2E_c} = \pm 1$ où $E_c = R\ E_b$ est l'énergie moyenne par bit codé sur fréquence porteuse, E_b est l'énergie moyenne par bit d'information et R est le rendement du code composite. Le canal AWGN est un canal symétrique ; en effet, si x et y sont respectivement l'entrée et la sortie du canal, nous avons :

$$\Pr(y|x = +1) = \frac{1}{\sqrt{2\pi N_0}} \exp\left(-\frac{(y-1)^2}{2N_0}\right) = \Pr(-y|x = -1)\ .$$

Deux algorithmes de décodage sont envisagés pour les codes constituants : le décodage probabiliste de Gallager [42], et l'algorithme aller-retour [2]. Pour ces algorithmes, l'information est échangée dans le graphe en faisant circuler des messages à partir de chaque sommet vers les sommets voisins le long des arêtes. Par définition, ces messages sont des informations extrinsèques sur les bits et ne dépendent pas des informations *a priori* de ces derniers. Ils permettent de prendre une décision sur les bits et contiennent une estimation de leurs fiabilités par le biais de l'APP. Si l'algorithme délivre en sortie des probabilités (respectivement des rapports logarithmiques de probabilités), alors l'espace des messages est l'intervalle $[0, 1]$ (respectivement l'espace réel $\mathbb{R}$). Dans les deux cas, les messages générés par les codes sont symétriques par rapport à $\frac{1}{2}$ s'il s'agit de probabilités, ou par rapport à 0 s'il s'agit de rapports logarithmiques de probabilités. Il en est de même au niveau de chaque bit. Sous ces hypothèses de symétrie du canal et des différents sommets du graphe de Tanner, nous pouvons supposer que les probabilités d'erreur par bit et par mot sont indépendantes du mot de code émis. Par conséquent, nous restreignons notre étude à l'étude des performances conditionnellement à l'émission d'un seul mot de code : le mot de code $\mathbf{c} = (0, 0, \dots, 0)$ par exemple. Ce mot de code, une fois modulé, correspond à la séquence de symboles $\mathbf{x} = (-1, -1, \dots, -1)$.

Enfin, nous cherchons à estimer les performances asymptotiques d'un code composite utilisant un processus de décodage itératif. Ceci consiste à évaluer le rapport signal-à-bruit minimal pour lequel la probabilité d'erreur par bit atteinte par un tel code est nulle après un nombre suffisant d'itérations. Les décisions sur les bits, et par conséquent la probabilité d'erreur, sont évaluées à partir des probabilités *a posteriori* (APP) des bits. Ainsi, si le mot de code $\mathbf{c} = (0, 0, \dots, 0)$ a été émis, une erreur apparaît lorsque la probabilité *a posteriori* qu'un bit c_i soit égal à 0 est inférieure à $\frac{1}{2}$ ou lorsque le rapport logarithmique des APP de ce bit, noté *appLR*, et défini par

$$appLR(c_i) = \log \frac{APP(c_i = 1)}{APP(c_i = 0)}\ ,$$

est positif. L'APP d'un bit ou le rapport logarithmique des APP sont également des variables aléatoires et sont déterminés par leur distribution de probabilité. La probabilité d'erreur par bit à chaque itération m s'obtient alors en intégrant la distribution de l'APP de chaque bit du mot de code sur toutes les valeurs positives. Il nous faut donc déterminer la distribution des probabilités *a posteriori* à chaque itération m, pour *chaque* bit du mot de code. Cette étude peut être cependant simplifiée en ne considérant la distribution de l'APP que d'un nombre limité de bits lorsque les codes composites ou les codes constituants vérifient certaines propriétés comme l'isotropie.

4.2 Isotropie d'un code

Chaque décodeur élémentaire dispose en entrée d'une observation et d'une information *a priori* sur les différents bits impliqués dans le graphe de Tanner du code composite $\mathcal{C}(N, K)$. Il délivre en sortie une information extrinsèque sur ces mêmes bits. Lorsqu'un bit appartient à plusieurs codes, *i.e.,* lorsque plusieurs répliques de ce bit sont considérées dans le code composite, les informations extrinsèques apportées par les différents codeurs doivent être combinées afin d'évaluer l'APP et l'information extrinsèque totales de ce bit. Ces diverses opérations se traduisent sur le graphe de Tanner au niveau des codes (décodage et génération des extrinsèques) et des bits (combinaison des extrinsèques). Nous étudierons plus en détails la propagation de l'information à travers ces sommets au fil des itérations dans la section 4.3.

Les informations circulant au cours du décodage itératif peuvent être indifféremment des probabilités ou des rapports logarithmiques. Nous nous intéressons aux distributions de ces différentes informations en entrée et en sortie des différents sommets du graphe pour pouvoir évaluer la probabilité d'erreur à chaque itération de décodage. Pour limiter le nombre de distributions à étudier, nous introduisons les notions d'isotropie et d'anisotropie d'un code.

4.2.1 Définition

Soit $C(n, k)$ un code constituant du code composite $\mathcal{C}(N, K)$.

Définition 4.2.1 *(Isotropie d'un code)*
Le code constituant C est dit isotrope si, lorsque les probabilités en entrée du code (i.e., les observations du canal et les probabilités a priori*) sont indépendantes et identiquement distribuées, la densité de probabilité de l'information* a posteriori *est indépendante de la position du bit. Sinon le code C est dit anisotrope.*

Si le code est isotrope, les distributions des informations extrinsèques sont également indépendantes de la position du bit.

Lorsque le code n'est pas isotrope, il faut déterminer le degré d'anisotropie δ_a du code : δ_a est donc le nombre de distributions distinctes de l'APP qui se propagent dans le graphe et vérifie $1 \leq \delta_a(C) \leq n$, où n est la longueur du code. La connaissance du degré d'anisotropie permet de n'étudier la propagation qu'au voisinage d'un nombre limité de bits égal au degré δ_a.

Nous proposons dans la section suivante une condition suffisante pour qu'un code soit isotrope. Cette condition s'appuie sur l'analyse du polynôme énumérateur de poids du code constituant $C(n, k)$.

4.2.2 APP et polynômes énumérateurs de poids conditionnels

Soient $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ un mot du code constituant $C(n,k)$ et $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ le vecteur des symboles reçus à l'entrée du décodeur. Nous avons vu au chapitre 2 que la probabilité *a posteriori* d'un bit codé c_i, $i = 1, \ldots, n$ s'écrit :

$$APP(c_i) = \Pr(c_i|\mathbf{y}) = \sum_{\mathbf{c}\in C} \Pr(c_i, \mathbf{c}|\mathbf{y}) = \sum_{\mathbf{c}\in C} \frac{\Pr(c_i, \mathbf{c})\Pr(\mathbf{y}|c_i, \mathbf{c})}{\Pr(\mathbf{y})} .$$

La probabilité conjointe $\Pr(c_i, \mathbf{c})$ est nulle si le i-ème bit du mot de code $\mathbf{c}$ est différent de c_i. Le vecteur $\mathbf{y}$ des observations du canal est connu. Par hypothèse, le canal est discret et sans mémoire, l'entrelacement est aléatoire et infini. Nous pouvons alors supposer que les observations du canal et les probabilités *a priori* sont indépendantes. L'APP du bit c_i devient :

$$APP(c_i) \propto \sum_{\mathbf{c}\in C|c_i} \prod_{\ell=1}^{n} \Pr(y_\ell|c_\ell)\Pr(c_\ell) . \tag{4.1}$$

L'APP d'un bit s'écrit donc comme la somme sur tous les mots de code dont la i-ème composante est égale à c_i d'un produit de probabilités sur toutes les composantes du mot de code. En termes de distributions de probabilité, ceci signifie que les mots de code ayant le même nombre de "1", *i.e.*, le même poids de Hamming, auront la même distribution de probabilité de leur APP respective.

Soit $A(x)$ le polynôme énumérateur de poids du code C. Nous avons la définition suivante :
Définition 4.2.2 *(Polynômes énumérateurs de poids conditionnels)*
Les deux polynômes énumérateurs de poids conditionnels associés à un bit c_i d'un mot de code de C sont définis par :

$$A_i^0(x) = \sum_\ell A_{i\ell}^0 x^\ell \qquad \text{et} \qquad A_i^1(x) = \sum_\ell A_{i\ell}^1 x^\ell ,$$

pour $i = 1, \ldots, n$. Les entiers $A_{i\ell}^0$ et $A_{i\ell}^1$ sont les nombres de mots de code de C de poids de Hamming ℓ tels que $c_i = 0$ ou $c_i = 1$ respectivement.

Il est parfois utile d'introduire une indéterminée z dans le polynôme énumérateur de poids conditionnel, associée au nombre de bits à "0" dans le mot de code. Par exemple,

$$A_i^0(x,z) = \sum_\ell A_{i\ell}^0 x^\ell z^{n-\ell}$$

Le polynôme énumérateur de poids total $A(x)$ du code C s'écrit :

$$A(x) = A_i^0(x) + A_i^1(x) \quad \forall\, i = 1, \ldots, n,$$

ou si deux indéterminées sont considérées,

$$A(x,z) = A_i^0(x,z) + A_i^1(x,z) \quad \forall\, i = 1, \ldots, n .$$

La définition 4.2.2 associée à l'équation 4.1 permet d'écrire la distribution de l'APP en fonction des polynômes énumérateurs de poids conditionnels de C. En effet, si nous supposons que les variables d'entrée $\Pr(y_\ell|c_\ell)\Pr(c_\ell)$, $\ell = 1, \ldots, n$, sont identiquement distribuées, l'APP d'un bit c_i est donnée par les équations :

$$APP(c_i = 1) \propto \sum_{\ell=1}^{n} A^1_{i\ell} \otimes [\Pr(y_1|c_1 = 1)\Pr(c_1 = 1)]^{\otimes \ell} \, [\Pr(y_1|c_1 = 0)\Pr(c_1 = 0)]^{\otimes n-\ell} \tag{4.2}$$

et

$$APP(c_i = 0) \propto \sum_{\ell=1}^{n} A^0_{i\ell} \otimes [\Pr(y_1|c_1 = 1)\Pr(c_1 = 1)]^{\otimes \ell} \, [\Pr(y_1|c_1 = 0)\Pr(c_1 = 0)]^{\otimes n-\ell} \ , \tag{4.3}$$

où la somme $x + y$ et le produit xy de deux variables aléatoires sont respectivement notés $2 \otimes x$ et $x^{\otimes 2}$ si x et y sont identiquement distribués. Dans les équations précédentes, l'indice ℓ a été remplacé par l'indice 1 puisque les distributions des probabilités $\Pr(y_\ell|c_\ell)\Pr(c_\ell)$ sont identiques. Nous déduisons de l'équation (4.2) (respectivement de l'équation (4.3)) que la densité de probabilité de $APP(c_i = 1)$ (respectivement de $APP(c_i = 0)$) et les distributions en entrée du décodeur de l'observation et d'information *a priori* sont liées via une transformation définie par le polynôme énumérateur de poids conditionnel $A^1_i(x, y)$ (respectivement $A^0_i(x, y)$) ou de manière équivalente par l'ensemble des coefficients $\{A^1_{i\ell}\}$ (respectivement $\{A^0_{i\ell}\}$). Ainsi dans le cas particulier où tous les bits ont les mêmes polynômes de poids conditionnels, $A^1_i(x)$ et $A^0_i(x)$ sont indépendants de la position du bit i, et sont notés respectivement $A^1(x)$ et $A^0(x)$. Tous les bits du code ont alors la même distribution de leur probabilité *a posteriori*. Le code C est donc isotrope.

Exemple 1
Le code de Hamming de longueur 7*, de dimension* 4 *et de distance minimale* 3*, noté* $C_0(7, 4, 3)$ *est isotrope. En effet, les polynômes énumérateurs de poids conditionnels du code* C *s'écrivent :*

$$A^0(x) = 1 + 4x^3 + 3x^4 \quad \textit{et} \quad A^1(x) = 3x^3 + 4x^4 + x^7$$

et sont indépendants de la position du bit i. De même, il est aisé de vérifier que le code de Hamming étendu $(8, 4, 4)$ *reste isotrope, ses polynômes énumérateurs de poids conditionnels étant :*

$$A^0(x) = 1 + 7x^4 \quad \textit{et} \quad A^1(x) = 7x^4 + x^7 \ .$$

L'indépendance des polynômes énumérateurs de poids conditionnels d'un code par rapport à la position du bit considéré constitue donc *une condition suffisante* pour que le code soit isotrope. Lorsque cette condition n'est pas vérifiée, une étude plus approfondie de la protection des bits dans le graphe est nécessaire.

4.2.3 Isotropie des codes en blocs et des codes convolutifs

A l'aide de la condition suffisante mise en évidence ci-dessus, nous pouvons démontrer les résultats suivants concernant l'isotropie des codes en blocs linéaires.

Théorème 4.2.1 *(Isotropie des codes cycliques)*
Un code linéaire binaire cyclique C *est isotrope.*

Démonstration :
Soient deux bits c_i et c_j d'un mot de code $\mathbf{c}$, placés aux positions i et j respectivement, avec $j > i$. Le coefficient $A^0_{i\ell}$ est le nombre de mots de code de poids ℓ satisfaisant $c_i = 0$. Comme le code C est cyclique, un décalage circulaire de $j - i$ bits convertit n'importe quel mot de code avec $c_i = 0$ en un autre mot de code de poids identique avec $c_j = 0$. Par conséquent $A^0_{i\ell} = A^0_{j\ell}$.
c.q.f.d.

Lemme 1 *(Extension d'un code isotrope)*
Si le polynôme énumérateur de poids conditionnel du code initial C satisfait $A^1_\ell = A^0_{\ell-1}$, pour tout poids pair ℓ, alors l'extension d'un code isotrope C est isotrope.

Démonstration :
Le code isotrope $C(n,k)$ est étendu en un code $C_e(n+1,k)$ en ajoutant un bit de parité c_{n+1} tel que la somme de tous les bits du mot de code soit nulle. Soient $A(x) = A^0(x)+A^1(x)$ le polynôme énumérateur de poids complet de C, et $A_e(x) = A^0_e(x)+A^1_e(x)$ le polynôme énumérateur de poids complet de C_e.
Soit j un entier compris entre 1 et n. Alors :

$$A^0_e(x) = \sum_{\ell \text{ pair}} A^0_\ell x^\ell + \sum_{\ell \text{ impair}} A^0_\ell x^{\ell+1} \; .$$

Le code C étant isotrope, $A^0_e(x)$ ne dépend pas de la position du bit j comme somme de deux polynômes indépendants de j. Le code C_e est donc isotrope.
Considérons maintenant le bit de parité c_{n+1}. Le polynôme énumérateur de poids conditionnel associé à c_{n+1} s'écrit :

$$A^0_e(x) = \sum_{\ell \text{ pair}} A_\ell x^\ell = \sum_{\ell \text{ pair}} A^0_\ell x^\ell + \sum_{\ell \text{ pair}} A^1_\ell x^\ell \; .$$

Ce polynôme est égal au précédent si et seulement si $A^1_\ell = A^0_{\ell-1}$ pour tous les poids ℓ.
c.q.f.d.

Théorème 4.2.2 *(Isotropie des codes BCH étendus)*
Tout code BCH binaire primitif étendu $C_e(n+1,k)$ est isotrope.

Démonstration :
Soit $C(n,k)$ un code BCH binaire de longueur primitive. Comme le code C est un code linéaire cyclique, il est isotrope. Soit $c(x) = \sum_{i=1}^n c_i x^i$ un mot de code de C de poids de Hamming ℓ. Par définition de la construction d'un code BCH, le polynôme générateur de C contient un facteur primitif, *i.e.*, $c(\alpha) = 0$ pour α élément primitif du corps fini $GF(n+1)$. Observons le polynôme énumérateur de poids conditionnel relatif au premier bit c_1 du mot de code $c(x)$. Prenons par exemple $c(x)$ de poids pair ℓ tel que $c_1 = 1$. Il existe $\ell - 1$ entiers $m_1, \ldots, m_{\ell-1}$ tels que :

$$1 + \alpha^{m_1} + \alpha^{m_2} + \ldots + \alpha^{m_{\ell-1}} = 0 \; .$$

Ce mot de code peut être facilement converti en $\ell - 1$ mots de code distincts de poids $\ell - 1$ avec $c_1 = 0$ en écrivant que $1 + \alpha^{m_j} = \alpha^{m'_j}$, pour $j = 1, \ldots, \ell - 1$. L'opération inverse est similaire : elle complémente le bit c_1 et transforme un mot de poids $\ell - 1$ en $\ell - 1$ mots de code distincts de poids ℓ. Ceci implique que $A^1_\ell = A^0_{\ell-1}$. D'après le lemme 1, le code BCH étendu $C_e(n+1,k)$ est isotrope. *c.q.f.d.*

L'étude de l'isotropie des codes convolutifs est moins formelle. Pour un code convolutif de longueur infinie et de rendement $\frac{k_c}{n_c}$, la distribution de la probabilité *a posteriori* du bit codé se trouvant en position fixée j, $j = 1, \ldots, n_c$, sur une branche du treillis est la même sur toutes les branches. Ceci provient de la structure cyclique (par branche) du treillis d'un code convolutif. Cependant, deux bits appartenant à la même branche du treillis peuvent ne pas avoir la même distribution pour leurs APP respectives. Nous en concluons que le degré d'anisotropie d'un code convolutif admet comme borne supérieure n_c.

Dans le cas des turbo-codes parallèle, les codes constituants sont reliés par leurs bits d'information seulement. Si les codes constituants sont des codes de rendement $\frac{1}{n_c}$, un seul bit par branche propage les informations extrinsèques au cours des itérations. Ceci garantit donc l'isotropie du code pour la propagation de l'APP à travers tout le turbo-code.

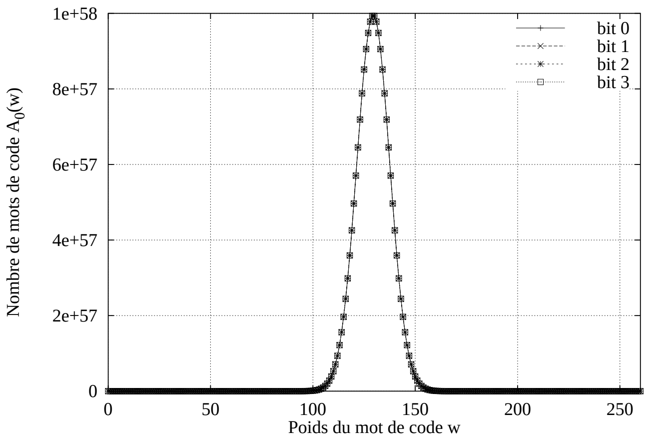
Pour les codes GLD convolutifs ou les turbo-codes série, tous les bits codés sont entrelacés et appartiennent à plusieurs codes. L'ensemble des bits codés fait partie intégrante du graphe et participe à la propagation de l'information au cours des itérations. Il est donc nécessaire pour ces codes d'observer la protection de chaque bit codé sur une branche du treillis du code convolutif. La figure 4.2 représente les distributions de poids conditionnelles par rapport à chaque bit d'une branche du treillis d'un code convolutif RSC de rendement $\frac{3}{4}$ et de polynômes générateurs $(13, 15, 16, 17)$, et d'un code convolutif NRNS de rendement $\frac{3}{4}$ et de polynômes générateurs $(13, 15, 2, 14)_D$. Le calcul des polynômes énumérateurs de poids conditionnels d'un code convolutif est étudié en annexe C. Une fenêtre contenant 65 branches du treillis de chaque code convolutif a été considérée pour calculer ces distributions. Nous voyons sur cette figure que les distributions de poids sont confondues quelle que soit la position du bit sur la branche. Ces codes convolutifs sont donc isotropes. Nous nous sommes également assurés de cette propriété pour l'ensemble des codes convolutifs envisagés au cours de ce chapitre.

4.2.4 Isotropie du code composite

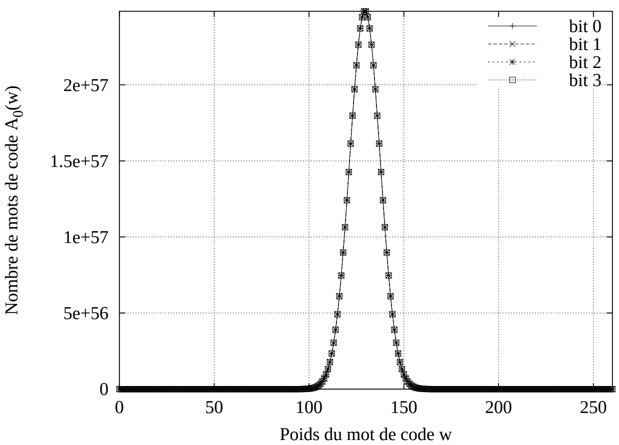
L'étude de l'isotropie des codes constituants nous permet de déterminer combien de distributions différentes doivent être considérées en sortie de chaque décodeur élémentaire lorsque toutes les probabilités en entrée de ce décodeur sont identiquement distribuées. Cette propriété caractérise la propagation au niveau d'un code constituant dans le graphe. Il est possible d'étendre cette notion au code composite $\mathcal{C}$, en considérant également les sommets bits, et d'identifier combien de distributions différentes se propagent dans le graphe du code $\mathcal{C}$.

Si le code composite est régulier, l'étude de l'isotropie de $\mathcal{C}$ se limite à l'étude de l'isotropie du code constituant. Si, en revanche, le code composite $\mathcal{C}$ est irrégulier, alors son anisotropie provient de son irrégularité et éventuellement de l'anisotropie des codes constituants. Prenons, par exemple, le cas d'un code LDPC irrégulier pour lequel une fraction λ des bits appartient à 3 équations de parité et une fraction $1 - \lambda$ des bits appartient à 4 équations de parité. Deux distributions de l'information *a posteriori* sont à étudier selon que le bit est de degré 3 ou de degré 4 dans le graphe. Le code LDPC ainsi défini est donc anisotrope de degré $\delta_a = 2$.

Décrivons maintenant la propagation de l'information dans le graphe d'un code composite



(a) *Code RSC de polynômes générateurs* $(13, 15, 16, 17)$



(b) *Code NRNS de polynômes générateurs* $(13, 15, 2, 14)_D$

Fig. 4.2 – Polynômes énumérateurs de poids conditionnels pour les bits d'une branche du treillis d'un code convolutif de rendement $R = \frac{3}{4}$

au fil des itérations.

4.3 Propagation de l'information dans le graphe

Nous considérons les notations et les hypothèses définies dans la partie 4.1. Le code composite $\mathcal{C}(N, K)$ est régulier et ses codes constituants $C^1, \ldots, C^J$ sont supposés identiques à une permutation près au code $C(n, k)$. Pour simplifier les expressions, nous faisons l'hypothèse que ces codes sont isotropes : par conséquent, une seule distribution de la probabilité *a posteriori* est à évaluer au cours des itérations, tous les bits codés ayant la même distribution de leur APP.

Au cours du décodage, des informations circulent le long des arêtes du graphe entre les bits et les codes. Ces informations sont des informations extrinsèques. Nous désignons par *itération* le passage de ces informations à travers un niveau code et un niveau bit dans le graphe. Les informations extrinsèques générées à la fin de chaque itération sont fournies au décodeur suivant comme informations *a priori*.

La probabilité *a priori* d'un bit est souvent associée à l'observation du canal ; nous appelons probabilité *a posteriori* partielle (ou APP partielle) d'un bit codé c_i, notée $\mathcal{A}(c_i)$, le produit de son information *a priori* $\Pr(c_i)$ et de l'observation du canal correspondante $\Pr(y_i|x_i)$ ($x_i = 2c_i - 1$ pour une modulation BPSK) :

$$\mathcal{A}(c_i) \propto \Pr(y_i|x_i)\ \Pr(c_i)\ .$$

Le coefficient de proportionnalité est choisi de telle sorte que $\mathcal{A}(c_i = 1) + \mathcal{A}(c_i = 0) = 1$.

La figure 4.3 illustre la propagation des informations extrinsèques Ext_m et des APP partielles $\mathcal{A}_m$ dans le graphe de Tanner d'un code composite à l'itération m. Rappelons que chaque code a pour degré d_c, et chaque bit a pour degré d_b.

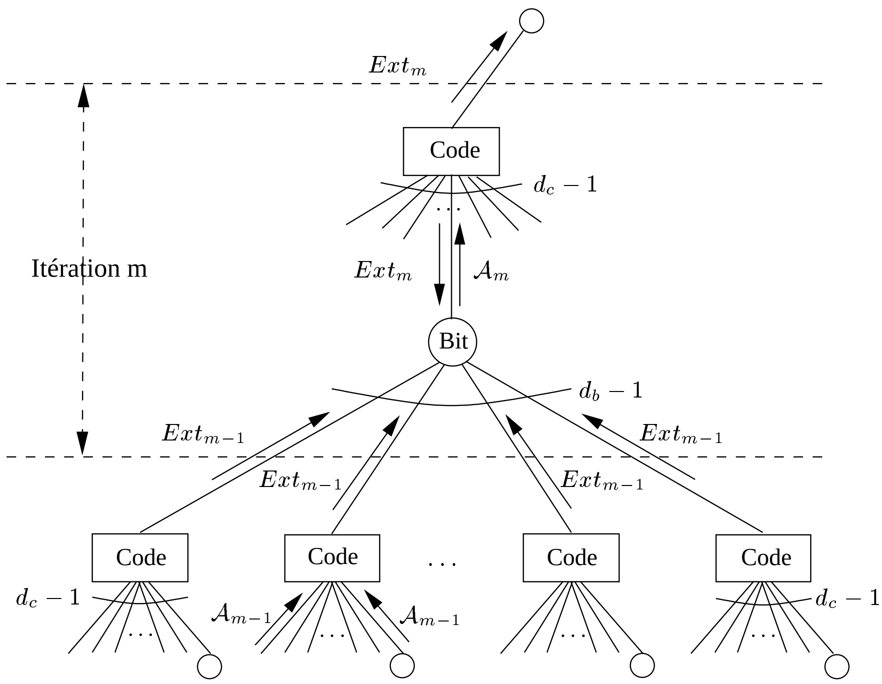


FIG. 4.3 – Propagation de l'information dans le graphe d'un code composite

Par définition, l'information extrinsèque circulant sur une arête e partant d'un sommet est générée en prenant en compte toutes les informations circulant sur les autres arêtes arrivant à ce sommet, sauf celle située sur l'arête e. Nous supposons par la suite que l'information se propage vers le haut du graphe. Étudions maintenant comment sont évaluées ces informations aux différents niveaux du graphe.

– **Au niveau d'un code** : le décodeur SISO génère une information extrinsèque sur le bit codé c_i à partir de $d_c - 1$ observations et probabilités *a priori*, *i.e.*, à partir de $d_c - 1$

probabilités *a posteriori* partielles, selon l'équation suivante :

$$\begin{aligned} Ext_{m-1}(c_i) &= \sum_{\mathbf{c}\in C|c_i} \prod_{\substack{\ell=1\\ \ell\neq i}}^{d_c} \Pr(y_\ell|c_\ell)\Pr_{m-1}(c_\ell) \\ &= \sum_{\mathbf{c}\in C|c_i} \prod_{\substack{\ell=1\\ \ell\neq i}}^{d_c} \mathcal{A}_{m-1}(c_\ell) \ . \end{aligned} \tag{4.4}$$

où $\Pr_{m-1}(c_\ell)$ désigne la probabilité *a priori* du bit c_ℓ à l'itération $m-1$.
Les $d_c - 1$ APP partielles arrivant sur le code correspondent aux APP partielles des bits se trouvant dans la fenêtre de dépendance de c_i. Si cette fenêtre est de longueur W (incluant c_i) alors le degré d'un code est $d_c = W$.
Signalons que l'équation 4.4 suppose l'indépendance des probabilités *a priori* et des observations du canal : l'indépendance des observations provient de l'hypothèse faite sur le canal (canal DMC), tandis que l'indépendance des probabilités *a priori* est due à l'entrelacement de longueur infinie appliqué sur les bits. Un tel entrelacement garantit en effet que le voisinage des sommets du graphe est sans cycle et que les informations à l'entrée de chaque décodeur sont décorrélées.

– **Au niveau d'un bit** : chaque bit évalue son information *a posteriori* partielle en combinant l'observation du canal et les $d_b - 1$ informations extrinsèques indépendantes arrivant des $d_b - 1$ codes voisins et évaluées à l'itération $m-1$:

$$\mathcal{A}_m(c_i) \propto \Pr(y_i|x_i) \prod_{\ell=1}^{d_b-1} \left(Ext_{m-1}(c_i)\right)_\ell \ . \tag{4.5}$$

Cette APP partielle est fournie au code supérieur (dans le graphe) comme information *a priori*. L'APP totale du bit c_i, qui permet de prendre une décision sur ce bit, est alors calculée en combinant l'APP partielle $\mathcal{A}_m$ et la nouvelle extrinsèque $Ext_m(c_i)$ délivrée par ce code à l'itération m :

$$APP_m(c_i) \propto \mathcal{A}_m(c_i)\ Ext_m(c_i) \ .$$

L'APP partielle $\mathcal{A}_m$ et l'information extrinsèque Ext_m circulent en sens opposé sur la même arête, comme le montre la figure 4.3.

D'après l'équation (4.5), l'information *a priori* utilisée à l'itération m est la combinaison de $d_b - 1$ informations extrinsèques générées à l'itération $m-1$. En associant l'équation (4.4) à la définition de l'APP partielle, nous pouvons alors écrire :

$$\mathcal{A}_m(c_i = 1) \propto \Pr(y_i|c_i = 1) \left[\sum_{\mathbf{c}\in C|c_i=1} \prod_{\substack{\ell=1\\ \ell\neq i}}^{d_c} \mathcal{A}_{m-1}(c_\ell) \right]^{\otimes d_b - 1}$$

et

$$\mathcal{A}_m(c_i = 0) \propto \Pr(y_i|c_i = 0) \left[\sum_{\mathbf{c}\in C|c_i=0} \prod_{\substack{\ell=1\\ \ell\neq i}}^{d_c} \mathcal{A}_{m-1}(c_\ell) \right]^{\otimes d_b - 1} \ .$$

En introduisant dans ces équations la définition 4.2.2 du polynôme énumérateur de poids conditionnel nous obtenons les équations finales :

$$\mathcal{A}_m(c_i = 1) \propto \Pr(y_i/c_i = 1) \left[\sum_{\ell=1}^{d_c} A_\ell^1 \otimes [\mathcal{A}_{m-1}(c_i = 1)]^{\otimes \ell - 1} [\mathcal{A}_{m-1}(c_i = 0)]^{\otimes d_c - \ell} \right]^{\otimes d_b - 1} \quad (4.6)$$

et

$$\mathcal{A}_m(c_i = 0) \propto \Pr(y_i/c_i = 0) \left[\sum_{\ell=0}^{d_c - 1} A_\ell^0 \otimes [\mathcal{A}_{m-1}(c_i = 1)]^{\otimes \ell} [\mathcal{A}_{m-1}(c_i = 0)]^{\otimes d_c - \ell - 1} \right]^{\otimes d_b - 1} . \quad (4.7)$$

Nous rappelons que la somme $x+y$ et le produit xy sont notés respectivement $2 \otimes x$ et $x^{\otimes 2}$ lorsque x et y sont des variables aléatoires identiquement distribuées. Ces deux équations définissent la propagation de l'information dans le graphe à chaque itération : l'APP partielle d'un bit à l'itération m est reliée à l'APP partielle de ce même bit délivrée à l'itération $m-1$.

Pour s'affranchir des coefficients de proportionnalité présents dans les définitions de l'APP et de l'APP partielle et dans les relations utilisant ces quantités, nous définissons différents rapports logarithmiques de probabilités, à l'itération m, pour le bit codé c_i :

- **Rapport logarithmique des APP partielles :**

$$\mathcal{B}_m(c_i) = \log \frac{\mathcal{A}_m(c_i = 1)}{\mathcal{A}_m(c_i = 0)} .$$

- **Rapport logarithmique des informations extrinsèques :**

$$extLR_m(c_i) = \log \frac{Ext_m(c_i = 1)}{Ext_m(c_i = 0)} .$$

  D'après l'équation (4.4), le rapport logarithmique des informations extrinsèques, $extLR_m(c_i)$, est calculé par un code à partir des rapports logarithmiques des APP partielles $\mathcal{B}_m(c_\ell)$ pour $\ell \neq i$.

- **Rapport logarithmique des APP totales :**

$$appLR_m(c_i) = \log \frac{APP_m(c_i = 1)}{APP_m(c_i = 0)} = \mathcal{B}_m(c_i) + extLR_m(c_i) .$$

- **Rapport logarithmique des observations du canal :**

$$obsLR_m(c_i) = \log \frac{\Pr(y_i/c_i = 1)}{\Pr(y_i/c_i = 0)} .$$

  Le rapport logarithmique des observations du canal est indépendant de l'itération m. De plus, à l'itération $m = 0$, l'information *a priori* de chaque bit codé est égale à $\frac{1}{2}$ pour un code binaire. Par conséquent, le rapport logarithmique des observations du canal est égal à :

$$obsLR_m(c_i) = \mathcal{B}_0(c_i) = appLR_0(c_i) .$$

Notons que le passage des rapports logarithmiques aux probabilités se fait aisément. Ainsi, par exemple,

$$\mathcal{A}_m(c_i = 1) = \frac{\exp(\mathcal{B}_m(c_i))}{1 + \exp(\mathcal{B}_m(c_i))} \quad \text{et} \quad \mathcal{A}_m(c_i = 0) = \frac{1}{1 + \exp(\mathcal{B}_m(c_i))} \ . \tag{4.8}$$

Aussi, par souci de simplicité et par abus de langage, nous appelons désormais observations, informations *a priori*, informations extrinsèques, APP et APP partielles les rapports logarithmiques de ces différentes quantités.

Le code est supposé isotrope. La distribution de l'APP partielle ne dépend donc pas de la position i du bit codé et nous pouvons omettre par la suite l'indice i pour simplifier les expressions. En utilisant les équations (4.6) et (4.7), nous obtenons la formule générale décrivant la propagation de la densité de probabilité de l'APP partielle au cours des itérations :

$$\mathcal{B}_m = \mathcal{B}_0 + (d_b - 1) \otimes \left[\log \frac{\sum_{i=1}^{d_c} A_i^1 \otimes [\exp(\mathcal{B}_{m-1})]^{\otimes i-1}}{\sum_{i=0}^{d_c-1} A_i^0 \otimes [\exp(\mathcal{B}_{m-1})]^{\otimes i}} \right] \ . \tag{4.9}$$

L'expression ci-dessus montre la manière dont la distribution de probabilité de $\mathcal{B}$ à l'itération m est liée à la distribution de probabilité de $\mathcal{B}$ à l'itération $m-1$. L'équation (4.9) s'écrit sous la forme :

$$\mathcal{B}_m = appLR_0 + (d_b - 1) \otimes extLR_{m-1} \ . \tag{4.10}$$

Du fait de la longueur infinie de l'entrelaceur, les informations extrinsèques $extLR_{m-1}$ sont indépendantes. La densité de probabilité de $\mathcal{B}_m$ s'écrit donc comme la convolution de d_b densités de probabilité, la densité de probabilité de $appLR_0$ et $d_b - 1$ densités de probabilité identiques définies à partir de la distribution de $\mathcal{B}_{m-1}$.

La distribution p_m de l'APP totale s'obtient ensuite en convoluant la distribution de l'APP partielle à la distribution de l'information extrinsèque d'après la relation suivante :

$$appLR_m = \mathcal{B}_m + extLR_m \ .$$

Pour un code isotrope, la probabilité d'erreur Pe_m, lorsque la séquence $(-1, -1, \dots, -1)$ a été émise, s'écrit alors :

$$Pe_m = \int_0^{+\infty} p_m(x) dx \ . \tag{4.11}$$

L'étude de l'évolution de cette probabilité d'erreur au cours des itérations permet de déduire les limites de l'ensemble des codes composites considéré. Il s'agit de déterminer le rapport signal-à-bruit minimum $\left(\frac{E_b}{N_0}\right)_{\min}$ pour que la probabilité d'erreur Pe_m tende vers 0 lorsque le nombre d'itérations m tend vers l'infini. Pour trouver ce rapport signal-à-bruit, il est donc nécessaire d'évaluer la distribution de probabilité de l'APP partielle et de l'information extrinsèque à chaque itération, en fonction des distributions de ces mêmes rapports à l'itération précédente (*cf.* formule (4.9)). Nous illustrons dans le paragraphe suivant comment évaluer ces distributions pour quelques codes composites.

4.4 Évaluation des performances asymptotiques des codes composites

Cette section est dédiée à l'évaluation des limites asymptotiques des codes composites introduits aux chapitres 2 et 3, avec décodage itératif, sur le canal AWGN. La formule de propagation (4.9) peut être utilisée directement dans quelques cas relativement simples, comme pour les codes LDPC ou les codes GLD blocs construits à partir de codes de Hamming "triviaux" (par exemple, le code $C_0(7,4)$). Cependant, pour la plupart des codes composites, appliquer directement cette formule s'avère trop coûteux en termes de calculs numériques. Nous proposons dans ce cas d'évaluer les distributions de probabilité requises en s'appuyant sur une méthode de type Monte-Carlo. Deux cas sont donc envisagés dans cette partie, selon que les distributions de probabilités des rapports logarithmiques peuvent ou non être déterminées de manière analytique. Puis nous donnons quelques résultats numériques pour divers ensembles de codes composites.

4.4.1 Codes LDPC

Considérons un code LDPC (N, j, ℓ) ; les degrés des bits et des codes (pce) sont respectivement $d_b = j$ et $d_c = \ell$. Tous les bits sont protégés de manière identique dans le graphe ; le code est donc isotrope : la distribution de l'APP partielle d'un bit codé à l'itération m ne dépend pas de la position de ce bit.

L'APP partielle $\mathcal{B}_{m+1}$ d'un bit codé c_i sur l'arête e reliant ce bit à un code C est la somme de d_b variables indépendantes : l'observation du canal et les $d_b - 1$ informations extrinsèques venant des codes autres que C reliés à ce bit. Nous avons, d'après [42], les relations suivantes :

$$\mathcal{A}_{m+1}(c_i = 1) \propto \Pr(y_i|x_i = 1)\ \Pr(c_i = 1)$$

avec

$$\Pr(c_i = 1) = \frac{1}{2} \prod_{p=1}^{d_b-1} \left[1 - \prod_{\ell_p=1}^{d_c-1} \left(1 - 2\mathcal{A}_m(c_{\ell_p} = 1)\right) \right] .$$

De même,

$$\mathcal{A}_{m+1}(c_i = 0) \propto \Pr(y_i|x_i = -1)\ \Pr(c_i = 0)$$

avec

$$\Pr(c_i = 0) = \frac{1}{2} \prod_{p=1}^{d_b-1} \left[1 + \prod_{\ell_p=1}^{d_c-1} \left(1 - 2\mathcal{A}_m(c_{\ell_p} = 1)\right) \right] .$$

Par conséquent, les relations (4.8) nous permettent d'écrire en omettant la position du bit c_i :

$$\begin{aligned}
\mathcal{B}_{m+1} &= appLR_0 + \sum_{p=1}^{d_b-1} (extLR_m)_p \\
&= appLR_0 + \sum_{p=1}^{d_b-1} \left(\log \frac{1 - \prod_{\ell_p=1}^{d_c-1} \frac{1-e^{\mathcal{B}_m(c_{\ell_p})}}{1+e^{\mathcal{B}_m(c_{\ell_p})}}}{1 + \prod_{\ell_p=1}^{d_c-1} \frac{1-e^{\mathcal{B}_m(c_{\ell_p})}}{1+e^{\mathcal{B}_m(c_{\ell_p})}}} \right)_p .
\end{aligned} \tag{4.12}$$

Les $d_b - 1$ informations extrinsèques ont la même distribution, d'où l'expression concise :

$$\mathcal{B}_{m+1} = appLR_0 + (d_b - 1) \otimes \log \frac{1 - \left(\frac{1-e^{\mathcal{B}_m}}{1+e^{\mathcal{B}_m}}\right)^{\otimes d_c - 1}}{1 + \left(\frac{1-e^{\mathcal{B}_m}}{1+e^{\mathcal{B}_m}}\right)^{\otimes d_c - 1}} .$$

Nous retrouvons l'équation de propagation proposée par Richardson et Urbanke [68] pour le calcul des seuils des codes LDPC. Il est alors possible de déterminer la distribution de la variable $\mathcal{B}$ à l'itération $m+1$ en fonction de la distribution de la variable $\mathcal{B}$ à l'itération m de manière analytique [68]. Étudions maintenant comment évaluer la distribution de probabilités de l'observation du canal $appLR_0$ et celle des informations extrinsèques $extLR_m$.

L'observation du canal $appLR_0$ a la même distribution quelle que soit l'itération m considérée. Si $y_i = x_i + z_i$ caractérise la sortie du canal AWGN à l'instant i, pour un bruit blanc z_i gaussien de moyenne nulle et de variance N_0, lorsque le bit $c_i = 0$ a été émis ($x_i = -1$), l'observation du canal est donnée par :

$$appLR_0(c_i) = 2\frac{y_i}{N_0} ,$$

où y_i est une variable aléatoire gaussienne de moyenne -1 et de variance N_0. Par conséquent la variable $appLR_0$ suit une loi gaussienne, de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$.

Pour déterminer la distribution des informations extrinsèques, nous appliquons la méthode proposée par Richardson et Urbanke [68]. Posons :

$$q = \frac{1 - e^{\mathcal{B}_m}}{1 + e^{\mathcal{B}_m}} .$$

La variable q prend ses valeurs dans l'intervalle $[-1, 1]$. De manière équivalente, nous avons la relation :

$$\mathcal{B}_m = \log \frac{1 - q}{1 + q} .$$

Soit u une variable aléatoire définie par

$$u = -\log |q| .$$

Par définition, si $p_{\mathcal{B}_m}$ et Q désignent les distributions de probabilité respectives des variables $\mathcal{B}_m$ et u, nous avons la relation :

$$p_{\mathcal{B}_m} |d\mathcal{B}_m| = Q|du| . \tag{4.13}$$

LA distribution Q est définie par Q^0 et Q^1, les distributions de u lorsque $q > 0$ et $q < 0$ respectivement :

$$Q = Q^0 + Q^1 .$$

En appliquant la relation (4.13), et en notant que

$$\left|\frac{d\mathcal{B}_m}{du}\right| = \frac{2e^u}{e^{2u} - 1} ,$$

nous obtenons, $\forall\ u \geq 0$,

$$Q^0(u) = \frac{2e^u}{e^{2u}-1} p_{\mathcal{B}_m} \left(\log \frac{e^u - 1}{e^u + 1} \right) \tag{4.14}$$

et

$$Q^1(u) = \frac{2e^u}{e^{2u}-1} p_{\mathcal{B}_m} \left(-\log \frac{e^u - 1}{e^u + 1} \right) \ . \tag{4.15}$$

Nous cherchons maintenant à déterminer la distribution du terme $q^{\otimes d_c - 1}$. Il s'agit du produit de $d_c - 1$ termes q_i indépendants et identiquement distribués par hypothèse. Nous considérons le logarithme de ce produit et introduisons la variable r telle que :

$$-\log|r| = -\log\left| \prod_{i=1}^{d_c-1} q_i \right| = -\sum_{i=1}^{d_c-1} \log|q_i| = -\sum_{i=1}^{d_c-1} u_i \ .$$

Soit R la distribution de $-\log|r|$; R est définie par R^0 et R^1, les distributions de $-\log|r|$ lorsque $r > 0$ et $r < 0$ respectivement :

$$R = R^0 + R^1 \ .$$

La distribution de $-\log|r|$ s'écrit comme la convolution de $d_c - 1$ distributions identiques. Pour simplifier les calculs, nous considérons le domaine de Fourier, dans lequel la transformée de cette distribution est égale au produit des transformées de Fourier des distributions des variables q_i.

Soient $\hat{Q}^0$, $\hat{Q}^1$, $\hat{R}^0$ et $\hat{R}^1$, les transformées de Laplace respectives des fonctions Q^0, Q^1, R^0 et R^1. Les transformées de Fourier bidimensionnelles des distributions R et Q, notées respectivement F_R et F_Q, sont définies par :

$$F_Q(f,0) = \hat{Q}^0(f) + \hat{Q}^1(f) \qquad F_Q(f,1) = \hat{Q}^0(f) - \hat{Q}^1(f)$$

et

$$F_R(f,0) = \hat{R}^0(f) + \hat{R}^1(f) \qquad F_R(f,1) = \hat{R}^0(f) - \hat{R}^1(f) \ ,$$

d'où les relations suivantes :

$$\hat{R}^0(f) + \hat{R}^1(f) = \left(\hat{Q}^0(f) + \hat{Q}^1(f) \right)^{d_c-1} \tag{4.16}$$

et

$$\hat{R}^0(f) - \hat{R}^1(f) = \left(\hat{Q}^0(f) - \hat{Q}^1(f) \right)^{d_c-1} \ . \tag{4.17}$$

Nous pouvons maintenant à l'aide de ces équations déterminer la distribution S de l'information extrinsèque. De nouveau, S est définie par les distributions S^0 et S^1 :

$$S = S^0 + S^1 \ ,$$

selon le signe de r. L'information extrinsèque s'écrit comme une fonction de $-\log|r|$. En appliquant la relation 4.13 aux distributions S et R, nous obtenons, $\forall\ x \geq 0$,

$$S^0(x) = \frac{2e^x}{1-e^{2x}} R^0 \left(\log \frac{1+e^x}{1-e^x} \right) \tag{4.18}$$

et

$$S^1(x) = \frac{2e^x}{e^{2x}-1} R^1\left(-\log\frac{1+e^x}{1-e^x}\right) . \tag{4.19}$$

En résumé, voici les différentes étapes nécessaires pour déterminer la distribution de l'APP partielle à l'itération $m+1$. La distribution de l'APP partielle $\mathcal{B}_m$ à l'itération précédente est supposée connue et égale à $p_{\mathcal{B}_m}$; à l'itération 0, il s'agit comme nous l'avons vu, d'une distribution gaussienne de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$.

1. Calcul des distributions Q^0 et Q^1 à l'aide des équations (4.14) et (4.15).
2. Évaluation des transformées de Fourier de R^0 et R^1 à l'aide des équations (4.16) et (4.17) ; nous déduisons de ces transformées de Fourier les distributions R^0 et R^1.
3. Calcul des distributions S^0 et S^1 grâce aux équations (4.18) et (4.19).
4. La distribution de l'APP partielle à l'itération $m+1$ est la convolution de d_b distributions de probabilité : la distribution de l'observation et d_b-1 distributions identiques de l'information extrinsèque S déterminées par les étapes précédentes.

La distribution de l'APP totale à l'itération $m+1$ est alors définie comme la convolution de la distribution de l'APP partielle $\mathcal{B}_m$ et de la distribution de l'information extrinsèque $extLR_m$ à l'itération m.

Parallèlement à cette méthode analytique, il est également possible de déterminer la distribution de l'information extrinsèque à l'itération $m+1$ par simulation, en utilisant une méthode de type Monte-Carlo. Nous supposons que la distribution de l'APP partielle à l'itération m est connue ; cette distribution est indépendante de la position du bit. L'information extrinsèque d'un bit codé s'écrit :

$$extLR_m = \log\frac{1-\left(\frac{1-e^{\mathcal{B}_m}}{1+e^{\mathcal{B}_m}}\right)^{\otimes d_c-1}}{1+\left(\frac{1-e^{\mathcal{B}_m}}{1+e^{\mathcal{B}_m}}\right)^{\otimes d_c-1}}$$

Il faut donc tirer d_c-1 variables aléatoires indépendantes suivant la distribution $p_{\mathcal{B}_m}$ de l'APP partielle $\mathcal{B}_m$ et calculer la valeur correspondante de l'information extrinsèque à l'aide de cette équation. Cette valeur vient remplir un histogramme utilisé pour décrire la distribution de $extLR_m$. L'opération est réitérée plusieurs fois afin d'obtenir une estimation précise de cette distribution. Elle est alors utilisée pour calculer la distribution de l'APP partielle $\mathcal{B}_{m+1}$, puis de l'APP totale $appLR_{m+1}$, par convolution comme décrit précédemment.

Une telle méthode peut être utilisée pour déterminer les distributions de l'information extrinsèque et de l'APP lorsqu'il est trop difficile de calculer de manière analytique ces distributions. C'est le cas notamment de la plupart des codes composites, hormis les codes LDPC. Pour les codes GLD blocs, lorsque le code constituant est relativement simple, il est aisé d'obtenir une équation simple de la propagation de l'APP partielle en appliquant la formule de propagation (4.9). Cependant, pour des codes constituants plus complexes, la distribution de l'information extrinsèque reste difficile à évaluer. Nous employons alors la méthode semi-analytique décrite ci-dessus [62]. Dans le cas de codes composites constitués de codes plus complexes, comme les codes convolutifs par exemple, nous proposons d'évaluer directement par méthode Monte-Carlo les distributions des informations extrinsèques et des APP.

4.4.2 Codes composites

Appliquer directement la formule de propagation (4.9) est trop coûteux en calculs numériques pour la plupart des codes composites. Ainsi, lorsque les codes constituants sont des codes convolutifs et l'algorithme de décodage utilisé est l'algorithme aller-retour, l'évaluation des distributions des informations extrinsèques et des APP se fait en observant les distributions des densités de probabilités α et β (*cf.* équations (2.14) et (2.15), chapitre 2). Même pour un code convolutif relativement simple dont le treillis est composé de 4 états, ceci signifie qu'à chaque itération 4 distributions pour α et 4 distributions pour β doivent être déterminées. Puis, à partir de ces distributions il faut extraire celles des informations souples qui nous intéressent. Nous avons opté pour une approche beaucoup plus pragmatique utilisant directement l'algorithme aller-retour pour évaluer les distributions des informations extrinsèques et des APP. A partir de ces distributions, nous déterminons la probabilité d'erreur à l'itération m en appliquant la formule (4.11).

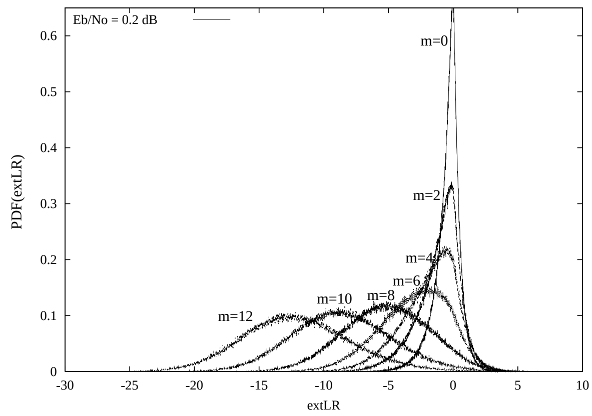
Soit $C(n, k)$ un code constituant du code composite $\mathcal{C}(N, K)$; ce code est par hypothèse isotrope. Nous supposons que les distributions de l'information extrinsèque et de l'APP partielle des bits codés sont connues à l'itération $m-1$. La distribution de l'observation du canal est une distribution gaussienne de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$. Si W est la fenêtre de dépendance autour de chaque bit codé, nous tirons W échantillons selon la loi de l'observation et W informations *a priori*. Ces informations a priori sont nulles s'il s'agit de bits de parité (degré égal à 1), ou distribuées selon la loi de l'information extrinsèque à l'itération $m-1$ s'il s'agit de bits d'informations. Nous appliquons ensuite l'algorithme aller-retour sur ces échantillons et construisons deux histogrammes utilisés pour décrire les distributions de l'information extrinsèque et de l'APP à l'itération m. Cette opération est répétée un nombre suffisant de fois pour obtenir une estimation fiable de ces distributions. Le premier histogramme permet d'extraire la loi des informations *a priori* à l'itération m, tandis que le second histogramme sur les APP permet de calculer la probabilité d'erreur à l'itération m.

La figure 4.4 représente l'évolution des distributions de l'information extrinsèque et de l'APP totale au fil des itérations, pour le turbo-code parallèle de rendement $R = \frac{1}{3}$, construit à partir de deux codes RSC $(37, 21)$, et à un rapport signal-à-bruit $\frac{E_b}{N_0} = 0.2$ dB. Les itérations $m = 0, 2, 4, 6, 8, 10, 12$ sont tracées. Nous constatons sur cette figure que les deux distributions sont très proches d'une distribution gaussienne, notamment lorsque le nombre d'itérations augmente. Par ailleurs, la convergence de ces distributions vers $-\infty$ à ce rapport signal-à-bruit est très rapide[1] : il faut, en effet, peu d'itérations pour obtenir une probabilité d'erreur nulle, c'est-à-dire pour que la «participation» de la distribution de probabilité de l'APP dans les valeurs positives tende vers 0 ($m = 12$ sur l'exemple représenté).

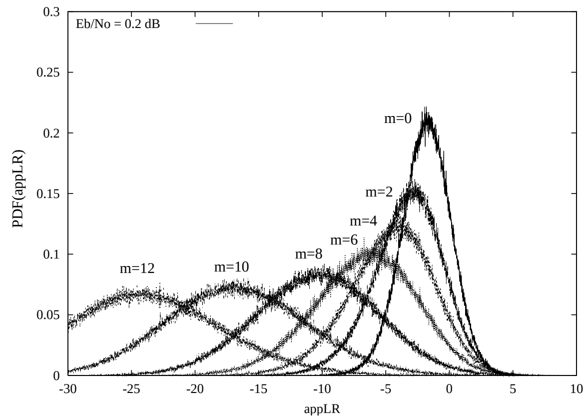
4.4.3 Résultats numériques

Nous avons appliqué les méthodes précédemment décrites aux différents codes composites présentés aux chapitres 2 et 3. Pour trouver les performances limites du décodage itératif appliqué à ces codes, il faut déterminer la valeur du rapport signal-à-bruit $\frac{E_b}{N_0}$ minimal (en dB) ou de la

[1] Rappelons que la séquence modulée $(-1, -1, \ldots, -1)$ a été émise sur le canal



(a) *Information Extrinsèque*



(b) *Probabilité a posteriori*

FIG. 4.4 – Évolution des distributions de $extLR$ et de $appLR$ en fonction de l'itération m.

variance de bruit N_0 maximale que le système peut accepter pour maintenir une probabilité d'erreur par bit qui tend vers zéro. Pour un code de rendement R, la relation entre ces deux paramètres est donnée par

$$(N_0)_{\max} = \frac{1}{2R}\left(\frac{E_b}{N_0}\right)^{-1}_{\min}.$$

Cette limite est atteinte avec un nombre infini d'itérations de décodage. Elle peut être déterminée en traçant la courbe représentant le nombre d'itérations minimal nécessaire pour atteindre une probabilité d'erreur arbitrairement petite en fonction du rapport signal-à-bruit $\frac{E_b}{N_0}$. La valeur du rapport signal-à-bruit minimal est alors donnée par l'asymptote verticale à cette courbe. Nous avons représenté cette courbe et son asymptote sur la figure 4.5 pour un turbo-code parallèle de rendement $R = \frac{1}{2}$ construit à partir de deux codes RSC de polynômes générateurs $(37, 21)$. Dans ce cas, l'asymptote se trouve en $\frac{E_b}{N_0} = 0.50$ dB (les valeurs sont estimées avec une précision de 0.01 dB).

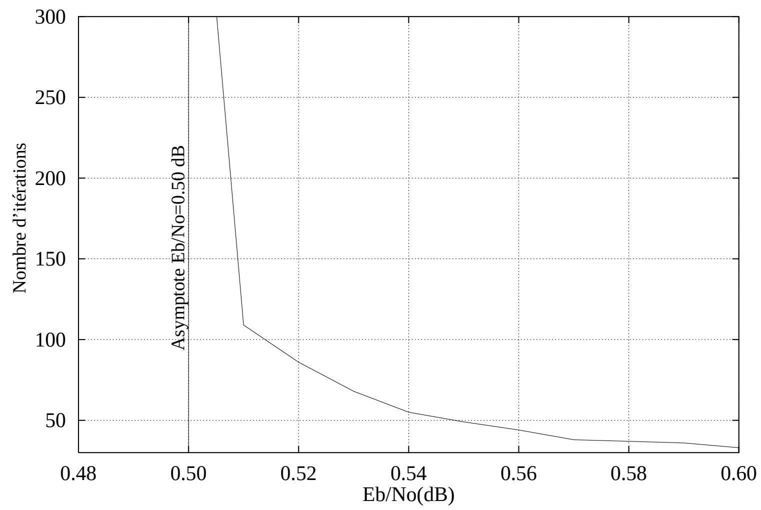


Fig. 4.5 – Nombre d'itérations en fonction du rapport $\frac{E_b}{N_0}$

Notons que pour les codes LDPC, la preuve de l'existence d'un seuil sur la variance du bruit a été apportée par [68] : pour un bruit dont la variance se situe au-delà de ce seuil, il a été démontré que la probabilité d'erreur ne peut plus tendre vers 0 quel que soit le nombre d'itérations considéré. Inversement, en-deçà de ce seuil, une probabilité d'erreur nulle peut être atteinte moyennant un nombre d'itérations de décodage suffisant. Une telle démonstration n'est pas connue pour les autres codes composites, tels les turbo-codes ou les codes GLD : l'existence d'un seuil n'est pas garantie, et il peut y avoir quelques erreurs résiduelles lorsque la variance du bruit est en-deçà des limites que nous avons évaluées. Cependant, les systèmes considérés en pratique ne requièrent pas de probabilité d'erreur nulle. Il est ainsi suffisant d'étudier les limites des codes composites pour atteindre des probabilités d'erreur relativement faibles par rapport aux valeurs souhaitées à défaut de garantir une probabilité d'erreur nulle.

Codes LDPC

Les deux méthodes présentées pour évaluer les limites des codes LDPC (analytique et semi-analytique) donnent des résultats identiques. La table 4.1 consigne les seuils des performances obtenues avec des codes LDPC pour différents degrés d_c et d_b avec un décodage APP itératif. Pour chaque code, ce tableau mentionne le rendement du code LDPC correspondant, et la valeur donnée par le calcul de la capacité du canal AWGN, pour une modulation BPSK et un code de même rendement décodé en optimisant le critère du maximum de vraisemblance. Les limites asymptotiques de ces différents codes sont données en termes de rapport signal-à-bruit minimal et de variance de bruit maximale.

Nous retrouvons dans ce tableau les résultats publiés par Richardson et Urbanke [68]. Ces différentes valeurs sont également à rapprocher des seuils calculés par Gallager [42] : les estimations de ces limites ont pu être nettement améliorées grâce à des moyens de calcul plus efficaces [57].

LDPC (d_b, d_c)	R	Capacité (ML)	$\left(\frac{E_b}{N_0}\right)_{\text{min}}$ (itératif)	$(N_0)_{\max}$
(3,6)	1/2	0.18 dB	1.09 dB	0.77804
(3,5)	2/5	-0.24 dB	0.88 dB	1.02072
(4,6)	1/3	-0.50 dB	1.65 dB	1.02586
(3,4)	1/4	-0.80 dB	0.94 dB	1.61075

Tab. 4.1 – Limites asymptotiques de codes LDPC avec décodage itératif sur canal AWGN

Codes à faible densité généralisés

Les tables 4.2 et 4.3 donnent les performances asymptotiques pour différents codes GLD ($J = 2$), construits à partir de codes en blocs ou de codes convolutifs. Les codes constituants sont identiques à une permutation près. Dans les deux cas, les limites ont été déterminées par simulation de l'algorithme aller-retour. Pour les codes en blocs, la fenêtre de dépendance W sur laquelle appliquer l'algorithme est égale à la longueur n du code constituant. Pour les codes convolutifs, nous avons considéré une fenêtre de dépendance de longueur W supérieure ou égale à $(32 \times \nu + 1)$ branches dans le treillis, où ν est la mémoire du code convolutif. Cette valeur a été obtenue par simulation en calculant le taux d'erreur du bit situé au milieu de la fenêtre. Elle correspond au nombre minimal de branches dans le treillis au-delà duquel le taux d'erreur binaire mesuré reste constant. La figure 4.6 s'intéresse au choix de cette fenêtre pour le code RSC de polynômes générateurs $(23, 35)$ et de rendement $r = \frac{1}{2}$.

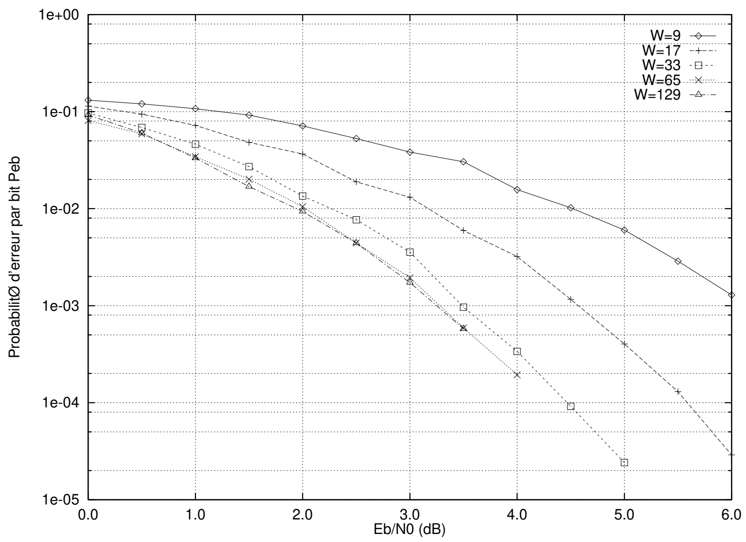


Fig. 4.6 – Fenêtre de dépendance pour un code de longueur de contrainte 5

Code constituant C	R	Capacité (ML)	$\left(\frac{E_b}{N_0}\right)_{\text{min}}$ (itératif)	$(N_0)_{\max}$
Hamming $(7, 4, 3)$	1/7	-1.15 dB	1.28 dB	2.60656
Hamming $(15, 11, 3)$	7/15	0.03 dB	0.83 dB	0.88504

Tab. 4.2 – Limites asymptotiques de codes GLD blocs avec décodage itératif

Code constituant C	R	Capacité (ML)	$\left(\frac{E_b}{N_0}\right)_{\min}$ (itératif)	$(N_0)_{\max}$
$RSC(15,16,13,17)$	1/2	0.18 dB	1.41 dB	0.72277
$NRNS(13,15,2,14)_D$	1/2	0.18 dB	0.85 dB	0.82224

TAB. 4.3 – Limites asymptotiques de codes GLD convolutif avec décodage itératif

La table 4.3 permet de valider le comportement des codes GLD convolutifs étudiés au chapitre 3 : en effet, nous avons vu au paragraphe 3.5 que le code GLD construit à partir du code NRNS de polynômes générateurs $(13, 15, 2, 14)_D$ permet, lorsque la longueur du code tend vers l'infini, de se rapprocher des limites données par la capacité du canal, alors que le code GLD construit à partir du code RSC de polynômes générateurs $(15, 16, 13, 17)$ ne le permet pas. Il s'agit comme le montre la table 4.3 de limites *imposées par le décodage itératif* et non par le décodage à maximum de vraisemblance, puisque la capacité du canal est indépendante du choix des polynômes générateurs des codes constituants.

Turbo-codes

Les tables 4.4 et 4.5 donnent les limites asymptotiques de différents turbo-codes construits à partir de $J = 2$ codes convolutifs. Les concaténations envisagées sont parallèle ou série. Les polynômes générateurs des codes constituants sont indiqués en octal ; le premier polynôme est le polynôme de rétroaction. Dans les deux cas, nous avons utilisé la méthode par simulation Monte-Carlo pour déterminer ces seuils.

Pour les turbo-codes parallèle, les codes constituants sont identiques à une permutation près. Il s'agit de codes convolutifs récursifs systématiques à 4, 8 ou 16 états. La lettre (P) à côté des polynômes générateurs des codes constituants de rendement $r = \frac{1}{2}$ indique que le turbo-code a été obtenu en perforant les bits en sortie de la concaténation. Les autres turbo-codes de rendement $R = \frac{1}{2}$ ont été construits à partir de codes constituants de rendement $r = \frac{2}{3}$.

Différents commentaires peuvent être faits en observant les valeurs consignées dans la table 4.4. Pour un rendement donné, le choix des polynômes générateurs du code constituant est déterminant pour les performances du décodage itératif. Ainsi, si on considère un treillis à 16 états, il est possible de gagner de l'ordre de $0.1 - 0.2$ dB selon les polynômes envisagés. Nous avons contrôlé par simulation que ce gain se maintient pour des longueurs d'entrelacement plus faibles, par exemple 1024 bits. Cependant, les codes ainsi choisis ne sont pas ceux qui mènent à une distance minimale du turbo-code la plus grande. Ceci peut être pris en compte en utilisant un entrelaceur approprié.
Nous remarquons également dans cette table que les turbo-codes de rendement $\frac{1}{2}$ obtenus à partir de codes de rendement $\frac{2}{3}$ se comportent «moins bien» face au décodage itératif que les turbo-codes obtenus à partir de codes de rendement $\frac{1}{2}$ et perforés.

Pour les turbo-codes série, nous avons choisi un code extérieur C^1 non récursif non systématique et un code intérieur C^2 récursif systématique (*cf.* chapitre 2). Le premier turbo-code série de la table 4.5 a pour codes constituants deux codes de rendement $r = 1/2$. Le second turbo-code série a un code externe de rendement $2/3$ (code NRNS) et un code interne de rendement $r = 3/4$

Code constituant C	R	Capacité (ML)	$\left(\frac{E_b}{N_0}\right)_{\min}$ (itératif)	$(N_0)_{\max}$
$RSC(31,27,23,37)$ $(r=1/4)$	1/7	-1.15 dB	-0.66 dB	4.07444
$RSC(37,21,31,27)$ $(r=1/4)$	1/7	-1.15 dB	-0.78 dB	4.18859
$RSC(7,5)$ $(r=1/2)$	1/3	-0.50 dB	0.02 dB	1.49311
$RSC(13,15)$ $(r=1/2)$	1/3	-0.50 dB	-0.07 dB	1.52437
$RSC(23,35)$ $(r=1/2)$	1/3	-0.50 dB	-0.01 dB	1.50346
$RSC(37,21)$ $(r=1/2)$	1/3	-0.50 dB	-0.15 dB	1.55271
$RSC(25,35)$ $(r=1/2)$	1/3	-0.50 dB	-0.22 dB	1.57794
$RSC(25,37)$ $(r=1/2)$	1/3	-0.50 dB	-0.24 dB	1.58523
$RSC(7,5)$ $(r=1/2)$ (P)	1/2	0.18 dB	0.69 dB	0.85310
$RSC(13,15)$ $(r=1/2)$ (P)	1/2	0.18 dB	0.59 dB	0.87297
$RSC(23,35)$ $(r=1/2)$ (P)	1/2	0.18 dB	0.62 dB	0.86696
$RSC(37,21)$ $(r=1/2)$ (P)	1/2	0.18 dB	0.50 dB	0.89125
$RSC(31,21,37)$ $(r=2/3)$	1/2	0.18 dB	0.64 dB	0.86298
$RSC(37,21,27)$ $(r=2/3)$	1/2	0.18 dB	0.60 dB	0.87096
$RSC(25,37,35)$ $(r=2/3)$	1/2	0.18 dB	0.55 dB	0.88105

TAB. 4.4 – Limites asymptotiques de turbo-codes parallèle avec décodage itératif

(code RSC). Notons que les turbo-codes série construits à partir de codes constituants différents sont des codes composites irréguliers. Il suffit alors d'observer successivement la propagation des densités de probabilité à travers les différents codes constituants.

Codes constituants (C^1, C^2)	R	Capacité (ML)	$\left(\frac{E_b}{N_0}\right)_{\min}$ (itératif)	$(N_0)_{\max}$
$C^1(7,5)$, $C^2(7,5)$	1/4	-0.80 dB	0.14 dB	1.93655
$C^1(17,6,15)$, $C^2(31,25,33,37)$	1/2	0.18 dB	0.88 dB	0.81658

TAB. 4.5 – Limites asymptotiques de turbo-codes série avec décodage itératif

La comparaison des résultats donnés par les tables 4.2, 4.3, 4.4 et 4.5 semble indiquer que, pour un rendement $R = \frac{1}{2}$, les turbo-codes parallèle réagissent mieux au décodage itératif que les turbo-codes série ou les codes GLD. Il faut cependant noter que l'étude des limites du décodage itératif s'intéresse à des rapports signal-à-bruit très faibles et ne traduisent en rien l'impact de la distance minimale du code pour un rapport signal-à-bruit plus élevé. Il a été en effet montré aux chapitres 2 et 3 que la distance minimale des turbo-codes série ou celle des codes GLD est plus grande que celle des turbo-codes parallèle.

Nous avons présenté une méthode pour estimer les limites, en termes de rapport signal-à-bruit ou de variance du bruit, d'un ensemble de codes composites décodés selon un processus itératif. Cette méthode est fondée sur l'analyse de l'évolution des distributions de probabilité *exactes* de l'information extrinsèque et de l'APP des bits codés, au cours des itérations. Aussi nous l'appelons par la suite *méthode de l'évolution de la densité*. Nous présentons dans la section suivante une méthode plus qualitative permettant d'analyser la convergence et le comportement du décodage itératif d'un code composite.

4.5 Méthode de la fonction de transfert

Par opposition à la méthode de l'évolution de la densité, la méthode présentée dans ce paragraphe repose sur l'introduction d'une approximation concernant les densités de probabilité de l'information extrinsèque et de l'APP. En effet, nous avons vu au paragraphe 4.4.2 que les informations circulant dans le graphe de Tanner d'un code composite $\mathcal{C}$ sont très proches de variables aléatoires gaussiennes. Cette approximation permet de donner une expression analytique de la probabilité d'erreur à l'itération m, en entrée et en sortie de chaque sommet du graphe. A l'aide de ces expressions, nous déterminons les performances asymptotiques du code $\mathcal{C}$. La perte occasionnée par l'approximation sur les résultats est négligeable. Nous utilisons ensuite cette méthode pour étudier le décodage itératif d'un code composite dont l'entrelacement se fait symbole par symbole, comme décrit au chapitre 3 pour les codes GLD convolutifs.

4.5.1 Approximation gaussienne

Le canal est supposé AWGN ; le bruit est une variable aléatoire gaussienne de moyenne nulle et de variance N_0. La séquence émise sur le canal est $\mathbf{x} = (-1, -1, \ldots, -1)$. A chaque instant, la sortie du canal est ainsi une variable aléatoire gaussienne de moyenne -1 et de variance N_0. Le rapport logarithmique des observations du canal est donc une variable aléatoire gaussienne de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$.

La figure 4.4 montre que le rapport logarithmique des informations extrinsèques peut être assimilé à une variable aléatoire gaussienne, de moyenne m_{ap} (négative) et de variance σ_{ap}^2. Cette approximation est d'autant plus valide que le nombre d'itérations m de décodage augmente. Il en est de même pour l'information extrinsèque exprimée sous forme de probabilité, qui est proche d'une variable aléatoire gaussienne de moyenne -1 et de variance $N_0(ap)$ (par analogie avec la variance N_0 du bruit). La moyenne et la variance du rapport logarithmique de l'information extrinsèque sont alors données par :

$$m_{ap} = -\frac{2}{N_0(ap)} \quad \text{et} \quad \sigma_{ap}^2 = \frac{4}{N_0(ap)} \ .$$

La valeur du paramètre $N_0(ap)$ varie au fil des itérations avec l'évolution des valeurs de l'information extrinsèque. Ainsi, lorsque la valeur de la probabilité extrinsèque d'un bit est égale à $\frac{1}{2}$, la variance $N_0(ap)$ tend vers l'infini. Aucune aide n'est réellement apportée par l'information extrinsèque lors du décodage itératif. Inversement, lorsque la valeur de la probabilité extrinsèque est voisine de 0 ou de 1, la variance $N_0(ap)$ tend vers de très faibles valeurs. L'information extrinsèque influence alors grandement les décisions prises sur le bit.

4.5.2 Probabilité d'erreur évaluée à partir de l'APP partielle

A partir de l'approximation précédente sur la distribution des informations extrinsèques, nous cherchons maintenant à déterminer simplement l'évolution de la probabilité d'erreur par bit Pe_m dans le graphe. En observant cette évolution, nous pourrons déterminer quelle est la variance du bruit maximale pour laquelle Pe_m tend vers zéro. Bien que Pe_m soit calculée à partir des décisions sur les bits, *i.e.*, à partir des APP totales, le théorème 4.5.1 et le lemme 2 indiquent que l'on peut aussi considérer la probabilité d'erreur évaluée à partir de l'APP partielle des bits pour déterminer les limites asymptotiques des codes composites avec décodage itératif. Il est en effet montré, que si la probabilité d'erreur évaluée à partir de la distribution de l'APP partielle $\mathcal{B}_m$ tend vers zéro pour un nombre d'itérations $m > m_0$, la probabilité d'erreur par bit $P\mathrm{e}_m$ tend également vers zéro.

Théorème 4.5.1
Soient x, y_m, z_m trois variables aléatoires réelles telles que

$$z_m = x + y_m \ ,$$

où m est un entier naturel. Nous supposons que la variable aléatoire x vérifie les deux hypothèses suivantes :
- $\forall\ \epsilon > 0,\ \exists\ A \in \mathbb{R},\ Pr(x \geq A) < \epsilon$
- $\forall\ \epsilon > 0,\ \exists\ A \in \mathbb{R},\ Pr(x \leq A) < \epsilon$

Les deux propositions ci-dessous sont alors équivalentes :
1. $\forall\ \alpha \in \mathbb{R},\ \forall\ \epsilon > 0,\ \exists\ m_0,\ \forall m > m_0\ Pr(y_m \geq \alpha) < \epsilon$
2. $\forall\ \beta \in \mathbb{R},\ \forall\ \epsilon > 0,\ \exists\ m_0,\ \forall m > m_0\ Pr(z_m \geq \beta) < \epsilon$

Démonstration :
1. Supposons que la propriété (1) est vérifiée :
$$\forall\ \alpha \in \mathbb{R},\ \forall\ \epsilon > 0,\ \exists\ m_0,\ \forall m > m_0\ \Pr(y_m \geq \alpha) < \epsilon \ .$$
Posons $\epsilon' = f(\epsilon)$. D'après les hypothèses formulées sur la variable x :
$$\exists\ A_{\epsilon'} \in \mathbb{R},\ \Pr(x \geq A_{\epsilon'}) < \epsilon' \ .$$
Soient β et ϵ deux réels strictement positifs. Nous avons
$$\Pr(z_m \geq \beta) = \Pr(z_m \geq \beta | x \leq A_{\epsilon'})\Pr(x \leq A_{\epsilon'}) + \Pr(z_m \geq \beta | x \geq A_{\epsilon'})\Pr(x \geq A_{\epsilon'}) \ .$$
Cette probabilité peut être majorée par :
$$\Pr(z_m \geq \beta) \leq \Pr(z_m \geq \beta | x \leq A_{\epsilon'}) + \epsilon'$$
(les autres probabilités ont été majorées par 1). Or,
$$x \leq A_{\epsilon'} \Leftrightarrow \beta - x \geq \beta - A_{\epsilon'}$$
donc
$$\Pr(z_m \geq \beta) \leq \Pr(y_m \geq \beta - A_{\epsilon'}) + \epsilon' \ .$$
En posant $\epsilon' = \frac{\epsilon}{2}$ et $\alpha = \beta - A_{\epsilon'}$, nous obtenons la proposition (2).
2. Supposons que la propriété (2) soit vérifiée :
$$\forall\ \beta \in \mathbb{R},\ \forall\ \epsilon > 0,\ \exists\ m_0,\ \forall m > m_0\ \Pr(z_m \geq \beta) < \epsilon \ .$$

Posons $\epsilon' = f(\epsilon)$. D'après les hypothèses formulées sur la variable x,

$$\exists\ A_{\epsilon'} \in \mathbb{R},\ Pr(x \leq A_{\epsilon'}) < \epsilon' \ .$$

Soient α et ϵ deux réels strictement positifs. Nous avons :

$$Pr(y_m \geq \alpha) = Pr(y_m \geq \alpha | x \leq A_{\epsilon'}) Pr(x \leq A_{\epsilon'}) + Pr(y_m \geq \alpha | x \geq A_{\epsilon'}) Pr(x \geq A_{\epsilon'}) \ .$$

Cette probabilité peut être majorée par :

$$Pr(y_m \geq \alpha) \leq \epsilon' + Pr(y_m \geq \alpha | x \geq A_{\epsilon'})$$

(les autres probabilités ont été majorées par 1). Or

$$x \geq A_{\epsilon'} \Leftrightarrow \alpha + x \geq \alpha + A_{\epsilon'}$$

donc

$$Pr(y_m \geq \alpha) \leq Pr(y_m \geq \alpha + A_{\epsilon'}) + \epsilon' \ .$$

En posant $\epsilon' = \frac{\epsilon}{2}$ et $\beta = \alpha + A_{\epsilon'}$, nous obtenons la proposition (1).
c.q.f.d.

Lemme 2
La séquence $\mathbf{x} = (-1, \ldots, -1)$ *est émise sur un canal AWGN. Considérons les deux propositions suivantes :*

1. $\forall \epsilon > 0,\ \exists\ \ m_0$ *tel que* $\forall m > m_0,\ Pr(\mathcal{B}_m > 0) < \epsilon$
2. $\forall \epsilon > 0,\ \exists\ \ m_0$ *tel que* $\forall m > m_0,\ Pr(appLR_m > 0) < \epsilon$

Alors la proposition (1) *implique la proposition* (2).

Démonstration :
Considérons le cas simple où :

$$appLR_m = \mathcal{B}_m + extLR_m \ ,$$

avec

$$\mathcal{B}_m = obsLR + extLR_{m-1} \ .$$

Le cas où $\mathcal{B}_m$ est obtenu à partir de $d_b - 1$ informations extrinsèques indépendantes et identiquement distribuées peut être directement déduit de cette démonstration. Posons alors :

$$z_m = appLR_m \qquad y_m = extLR_m + extLR_{m-1} \qquad x = obsLR \ .$$

La variable x vérifie les hypothèses du théorème 4.5.1 puisque x est une variable aléatoire gaussienne. Par conséquent, il existe un réel A positif tel que

$$\forall\ \epsilon' > 0,\ Pr(|x| \geq A) < \epsilon' \ .$$

Supposons que la proposition (1) est vérifiée. D'après le théorème 4.5.1 appliqué une première fois aux variables $(\mathcal{B}_m, obsLR, extLR_{m-1})$, nous avons :

$$\forall\ \alpha \in \mathbb{R},\ \forall\ \epsilon' > 0,\ \exists\ m_0',\ \forall m > m_0'\ Pr(extLR_{m-1} \geq \alpha) < \epsilon' \ . \tag{4.20}$$

Soient ϵ fixé, et $m_0'' = \max(m_0', m_0)$. Alors $\forall\ m > m_0''$:

$$\begin{aligned}\Pr(y_m > 0) &= \Pr(extLR_m + extLR_{m-1} > 0 | extLR_{m-1} \leq \alpha)\Pr(extLR_{m-1} \leq \alpha) \\ &+ \Pr(extLR_m + extLR_{m-1} > 0 | extLR_{m-1} \geq \alpha)\Pr(extLR_{m-1} \geq \alpha)\ .\end{aligned}$$

Cette probabilité peut être majorée par :

$$\Pr(y_m > 0) < \Pr(extLR_m + extLR_{m-1} > 0 | extLR_{m-1} \leq \alpha) + \epsilon'\ ,$$

ou encore

$$\Pr(y_m > 0) < \Pr(extLR_m > -\alpha) + \epsilon'\ .$$

D'après l'équation (4.20), $\exists\ m_0^{'''}$, $\forall\ m > m_0^{'''}$:

$$\Pr(extLR_m > -\alpha) < \epsilon',$$

donc en posant $\epsilon' = \frac{\epsilon}{2}$, nous avons :

$$\Pr(y_m > 0) < \epsilon\ .$$

En appliquant de nouveau le théorème 4.5.1 aux variables (z_m, x, y_m), nous obtenons la proposition (2).
c.q.f.d.

Signalons que l'approximation des informations extrinsèques par des variables aléatoires gaussiennes permet de montrer que les propositions (1) et (2) sont équivalentes.

4.5.3 Estimation de la probabilité d'erreur à chaque itération

Soit Pe_m' la probabilité d'erreur évaluée en prenant une décision sur les bits à partir de l'APP partielle. D'après le lemme 2, il est suffisant d'étudier la propagation de cette probabilité d'erreur pour s'assurer que la probabilité d'erreur Pe_m tend vers zéro ou non. Cette probabilité d'erreur est modifiée lors de son passage par chaque sommet du graphe de Tanner, comme l'indique le schéma de la figure 4.7. Elle se calcule donc à chaque itération en deux étapes : à la sortie d'un code, puis à la sortie d'un bit. Nous notons par la suite Pe'_{out} (respectivement Pe'_{in}) la probabilité d'erreur Pe_m' en sortie (respectivement en entrée) de chaque sommet.

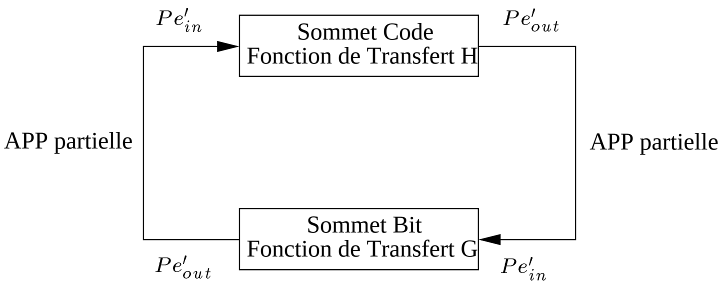


FIG. 4.7 – Évolution de Pe_m' dans le graphe

La probabilité d'erreur Pe'_{out} en sortie d'un sommet est une fonction de la probabilité d'erreur Pe'_{in} en entrée de ce sommet ; nous appelons H, respectivement G, la fonction appliquée sur la probabilité d'erreur au niveau d'un code, respectivement au niveau d'un bit. La propagation de l'information dans le graphe s'arrête lorsque la probabilité d'erreur à l'entrée du code est identique à la probabilité d'erreur à la sortie du bit pour la même itération m. Ceci équivaut à déterminer le point fixe, Pe', tel que

$$H(Pe') = G^{-1}(Pe')$$

Si le point fixe Pe' est nul, cela signifie que la probabilité d'erreur Pe'_m converge vers zéro après un certain nombre d'itérations m_0. A l'inverse, si le point fixe Pe' n'est pas nul, alors quel que soit le nombre d'itérations considéré, y compris un nombre infini d'itérations, la probabilité d'erreur Pe'_m ne tendra jamais vers zéro. Estimer les limites d'un code composite décodé itérativement revient donc à évaluer le rapport signal-à-bruit minimal pour lequel le point fixe Pe' est nul.

Pour cela, il est nécessaire de déterminer les fonctions de transfert H et G des sommets du graphe. Nous supposons que le code constituant est isotrope et que le graphe de Tanner du code composite est régulier. Ce graphe est, par exemple, celui représenté sur la figure 4.3. Les informations extrinsèques sont des variables aléatoires gaussiennes indépendantes et identiquement distribuées. Un code est de degré d_c et un bit de degré d_b.

Fonction de transfert d'un code

L'APP partielle, à l'entrée de chaque code, est la somme de l'observation du canal et d'une information *a priori* issue d'un bit. Cette information *a priori* est elle-même la somme de $d_b - 1$ variables indépendantes, identiquement distribuées selon une loi gaussienne de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$. L'APP partielle est donc également une variable aléatoire gaussienne de moyenne m_{code} et de variance σ^2_{code} définies par :

$$m_{code} = -\frac{2}{N_0} + \sum_{i=1}^{d_b-1} m_{ap} \quad \text{et} \quad \sigma^2_{code} = \frac{4}{N_0} + \sum_{i=1}^{d_b-1} \sigma^2_{ap},$$

soit

$$m_{code} = -\frac{2}{N_0} - \frac{2(d_b-1)}{N_0(ap)} \quad \text{et} \quad \sigma^2_{code} = \frac{4}{N_0} + \frac{4(d_b-1)}{N_0(ap)} \ .$$

Pour une variable aléatoire gaussienne de moyenne m_g et de variance σ_g^2, la probabilité d'erreur P_{err} s'écrit à l'aide de la fonction d'erreur Q :

$$P_{\text{err}} = Q\left(\frac{m_g}{\sigma_g}\right) \ .$$

Par conséquent, la probabilité d'erreur à l'entrée d'un code est :

$$Pe'_{\text{in}}(\text{code}) = Q\left(\sqrt{\frac{1}{N_0} + \frac{(d_b-1)}{N_0(ap)}}\right) \ .$$

A la sortie du code, nous évaluons la probabilité d'erreur Pe'_{out} par simulation, en déterminant la distribution de l'APP partielle $\mathcal{B}$ après son passage dans le décodeur SISO élémentaire, et en appliquant la formule suivante :

$$Pe'_{\text{out}}(\text{code}) = \Pr(\mathcal{B} > 0).$$

La figure 4.8 représente la probabilité d'erreur Pe'_{out} en fonction de la probabilité d'erreur Pe'_{in}, pour un code RSC de rendement $r = \frac{1}{2}$, de polynômes générateurs $(37, 21)$ et pour un rapport signal-à-bruit $\frac{E_b}{N_0} = 0.00$ dB (ce rapport fixe la variance du bruit N_0).

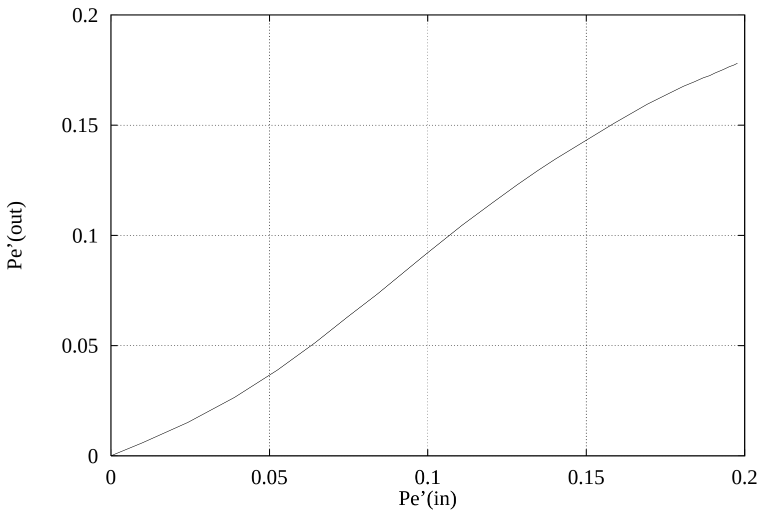


FIG. 4.8 – Fonction de transfert H d'un code

Fonction de transfert d'un bit

A l'entrée d'un bit, chaque APP partielle est constituée d'une information extrinsèque délivrée par un code et de l'observation du canal. Par conséquent, l'APP partielle à l'entrée d'un bit est une variable aléatoire gaussienne dont la moyenne m_{bit} et la variance σ^2_{bit} sont définies par :

$$m_{bit} = -\frac{2}{N_0} - \frac{2}{N_0(ap)} \quad \text{et} \quad \sigma^2_{bit} = \frac{4}{N_0} + \frac{4}{N_0(ap)} .$$

La probabilité d'erreur à l'entrée d'un bit s'écrit donc :

$$Pe'_{\text{in}}(\text{bit}) = Q\left(\sqrt{\frac{1}{N_0} + \frac{1}{N_0(ap)}}\right) .$$

Chaque bit combine à l'observation du canal $d_b - 1$ informations extrinsèques générées par les codes lui étant reliés. L'APP partielle à la sortie d'un bit est donc la somme de l'observation du canal et de $d_b - 1$ informations extrinsèques. Par conséquent, cette APP partielle est une variable gaussienne de moyenne m_{bit} et de variance σ^2_{bit} données par :

$$m_{bit} = -\frac{2}{N_0} - \frac{2(d_b - 1)}{N_0(ap)} \quad \text{et} \quad \sigma^2_{bit} = \frac{4}{N_0} + \frac{4(d_b - 1)}{N_0(ap)} .$$

La probabilité d'erreur à la sortie d'un bit s'écrit donc :

$$Pe'_{\text{out}}(\text{bit}) = Q\left(\sqrt{\frac{1}{N_0} + \frac{(d_b - 1)}{N_0(ap)}}\right) .$$

Lorsque le nombre de codes constituants du code composite est égal à $J = d_b = 2$, la fonction de transfert d'un bit est la fonction identité, *i.e.*, graphiquement il s'agit de la première bissectrice $Pe'_{\text{out}} = Pe'_{\text{in}}$. C'est le cas, par exemple, du turbo-code parallèle décrit originalement par [11]. La figure 4.9 représente la probabilité d'erreur Pe'_{out} en fonction de la probabilité d'erreur Pe'_{in} au niveau d'un bit, lorsque le code composite est formé de $J = 3$ codes, pour un rapport signal-à-bruit $\frac{E_b}{N_0} = 0.00$ dB.

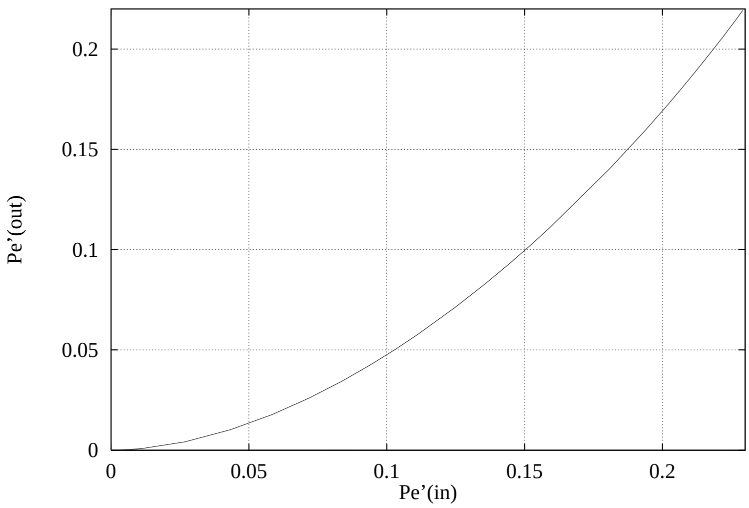


FIG. 4.9 – Fonction de transfert G d'un bit de degré 3

4.5.4 Détermination de la limite du décodage itératif d'un code composite

A l'aide des fonctions de transfert que nous venons d'évaluer pour chaque sommet du graphe, nous pouvons déterminer la limite du décodage itératif d'un code composite quelconque. Il suffit, en effet, de représenter les fonctions de transfert des deux types de sommets sur le même diagramme, en respectant le schéma de la figure 4.7. Ainsi, en abscisse, nous considérons la probabilité d'erreur en entrée du code $Pe'_{\text{in}}(\text{code})$ et la probabilité d'erreur en sortie d'un bit $Pe'_{\text{out}}(\text{bit})$, tandis qu'en ordonnée, se trouvent la probabilité d'erreur à la sortie d'un code $Pe'_{\text{out}}(\text{code})$ et la probabilité d'erreur en entrée d'un bit $Pe'_{\text{in}}(\text{bit})$.

La figure 4.10(a) représente le diagramme d'un turbo-code de rendement $R = \frac{1}{3}$, construits à partir de deux codes RSC de rendement $r = \frac{1}{2}$ et de polynômes générateurs $(37, 21)$. Le rapport signal-à-bruit est égal à 0.00 dB. La propagation de l'information et l'évolution de la probabilité d'erreur au fil des itérations se font, comme l'indiquent les flèches sur la figure, en «zigzaguant» entre les deux courbes de fonctions de transfert jusqu'au point d'intersection de ces deux courbes. Le point de départ de la propagation est donné par la probabilité d'erreur lorsque les probabilités extrinsèques sont toutes égales à $\frac{1}{2}$. Cette probabilité d'erreur initiale Pe'_0 est égale à la probabilité d'erreur en présence d'un bruit gaussien de moyenne nulle et de variance N_0 :

$$Pe'_0 = Q\left(\frac{1}{N_0}\right) .$$

Deux types de convergence peuvent être identifiés sur la figure 4.10(a) :

- une convergence lente et la présence d'un point critique aux alentours de $Pe'_{\text{in}} = 0.12$. Les deux fonctions de transfert se rapprochent dans cette zone, qui constitue une sorte de goulet d'étranglement pour la propagation de l'information ; de nombreuses itérations sont alors nécessaires pour aller au-delà de ce point.
- une convergence rapide avant et après ce point critique jusqu'à l'obtention d'une probabilité d'erreur nulle.

Le point critique joue un rôle important dans le sens où si l'on diminue légèrement le rapport signal-à-bruit, les deux fonctions de transfert vont se rapprocher de plus en plus jusqu'à se rencontrer. La convergence de l'information sera alors interrompue en ce point : le point fixe ne sera plus égal à zéro, et la probabilité d'erreur du système avec décodage itératif ne tendra plus vers zéro, quel que soit le nombre d'itérations considérées.

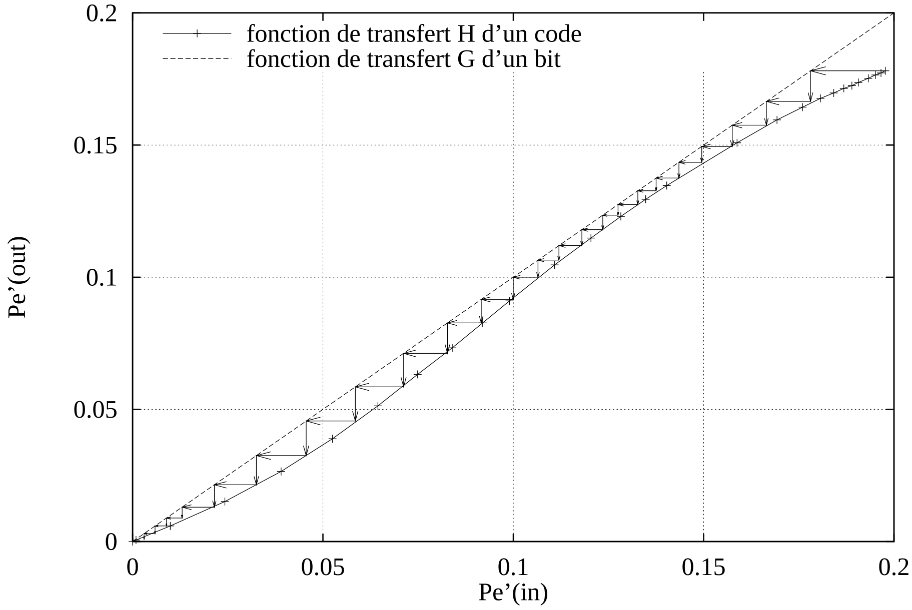
La figure 4.10(b) montre l'évolution de la fonction de transfert du code $(37, 21)$ avec la variance du bruit pour $\frac{E_b}{N_0} = 0.50$ dB et $\frac{E_b}{N_0} = -0.20$ dB. Rappelons que lorsque $J = 2$, la fonction de transfert d'un bit est égale à la fonction identité et ne varie donc pas en fonction du rapport $\frac{E_b}{N_0}$. Sur les deux diagrammes représentés à la figure 4.10(b) nous pouvons deviner le point critique. Pour le premier diagramme, la convergence reste rapide tout le long de la propagation, malgré un très léger ralentissement au niveau du point critique. En revanche, sur le second diagramme, la convergence vers une probabilité d'erreur nulle n'est plus possible en raison de la variance du bruit qui est trop élevée.

Le tableau 4.6 s'intéresse à l'influence de l'approximation gaussienne sur la précision des valeurs estimées pour les limites d'un code composite. Les limites obtenues avec la méthode de l'évolution de la densité sont comparées aux valeurs données par la méthode de la fonction de transfert pour différents turbo-codes de rendement $R = \frac{1}{3}$. Les écarts peuvent légèrement varier

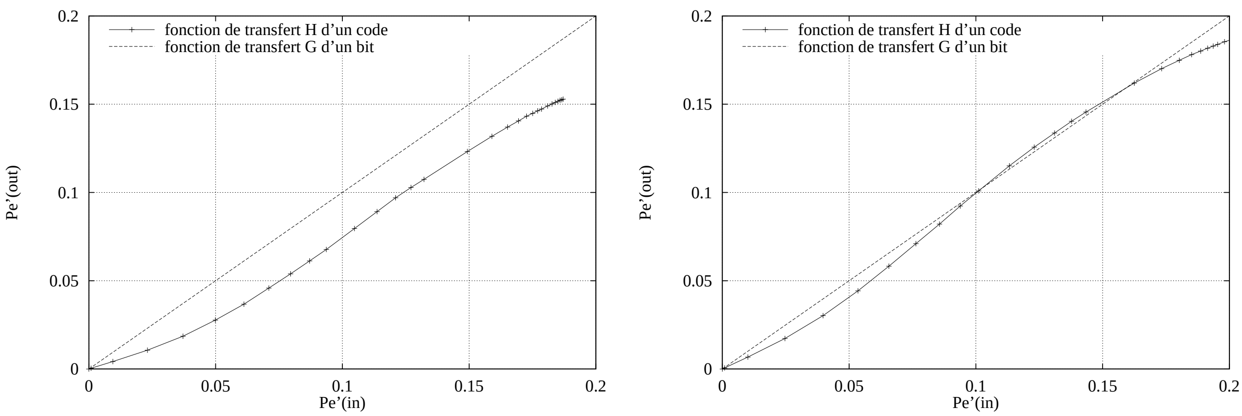
Code constituant	Méthode de l'évolution de la densité	Méthode de la fonction de transfert
RSC $(37, 21)$	-0.15 dB	-0.13 dB
RSC $(13, 15)$	-0.07 dB	-0.02 dB
RSC $(7, 5)$	0.02 dB	0.05 dB

Tab. 4.6 – Influence de l'approximation gaussienne en termes de précision

selon le nombre d'états des codes considérés ; cependant ils restent négligeables (≤ 0.05 dB).



(a) *Propagation de l'information dans un turbo-code parallèle*



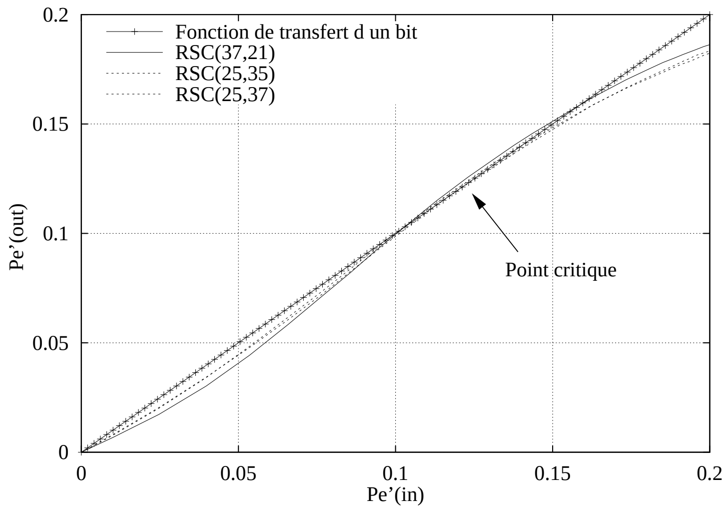
$\frac{E_b}{N_0} = 0.50 \;\; dB$ $\frac{E_b}{N_0} = -0.20 \;\; dB$

(b) *Évolution de la fonction de transfert du code en fonction du rapport* $\frac{E_b}{N_0}$

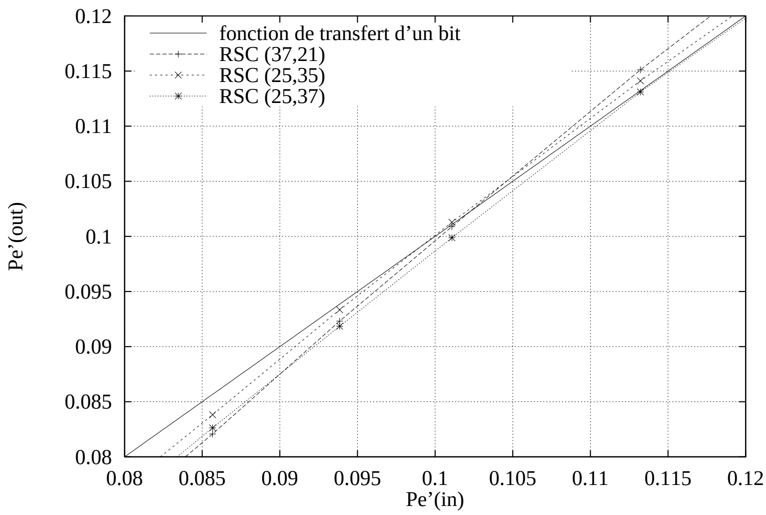
FIG. 4.10 – Fonctions de transfert d'un turbo-code parallèle

Choix des codes constituants d'un code composite

Pour un ensemble de codes composites donné, la méthode de la fonction de transfert permet de déterminer quels sont les codes constituants les plus adaptés au décodage itératif, c'est-à-dire les codes menant au rapport $\left(\frac{E_b}{N_0}\right)_{\min}$ le plus faible. Ainsi, la figure 4.11 met en évidence comment optimiser le choix du code constituant d'un turbo-code parallèle de rendement $R = \frac{1}{3}$, construit à partir de deux codes RSC de rendement $r = \frac{1}{2}$ à 16 états. Nous n'avons représenté sur cette figure que les codes permettant à la probabilité d'erreur de converger vers zéro pour un rapport signal-à-bruit inférieur ou égal à -0.10 dB. Le rapport signal-à-bruit sur la figure 4.11 est égal à -0.20 dB. Le meilleur code constituant pour ce turbo-code est le code de polynômes générateurs $(25, 37)$: on voit en effet sur la figure agrandie autour du point critique, que seul le code $(25, 37)$ permet encore à la probabilité d'erreur de converger vers zéro à ce rapport signal-à-bruit. Rappelons que le tableau 4.4 donnait comme limites des turbo-codes construits à partir des codes RSC $(37, 21)$, $(25, 35)$ et $(25, 37)$ et évaluées par la méthode de l'évolution de la densité, les valeurs respectives de -0.15 dB, -0.22 dB et -0.24 dB.



(a) *Fonctions de transfert de différents codes RSC à* 16 *états*



(b) *Agrandissement autour du point critique*

Fig. 4.11 – Optimisation du choix des codes constituants d'un turbo-code parallèle

Bien que cette optimisation soit faite en considérant que la taille de l'entrelaceur est infinie, la figure 4.12 montre que la hiérarchie entre les codes est respectée à faibles rapports signal-à-bruit pour une longueur de code $N = 2048$. Pour des rapports signal-à-bruit plus élevés (supérieurs à 0.6 dB), la distance minimale est le facteur dominant pour les performances, ce qui explique l'inversion des comportements sur la figure.

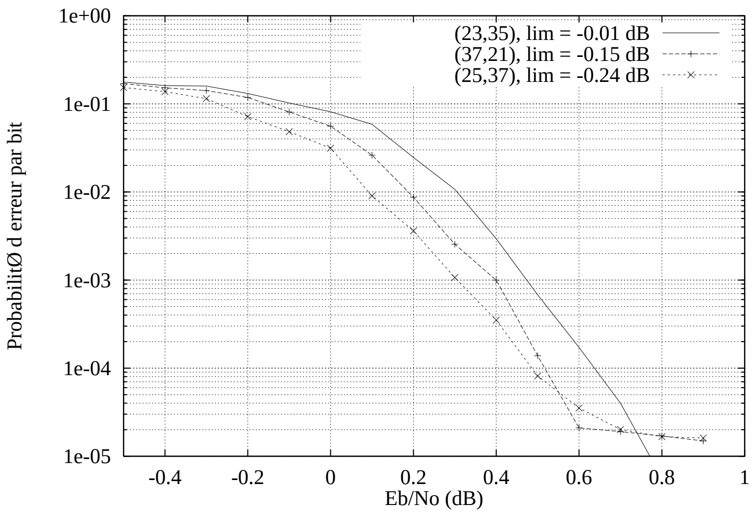


Fig. 4.12 – Turbo-code de longueur $N = 2048$ et de rendement $R = \frac{1}{3}$.

Le choix des polynômes des codes constituants d'un turbo-code n'est qu'un exemple d'application de la méthode de la fonction de transfert. Cette méthode n'est bien entendu pas limitée aux turbo-codes parallèle, et peut être utilisée pour étudier le décodage itératif d'un code composite quelconque, à la seule condition d'évaluer les fonctions de transfert des sommets du graphe. Dans le paragraphe suivant, nous adaptons cette méthode à l'étude du décodage itératif d'un code composite dont l'entrelacement se fait symbole par symbole (quaternaire), comme celui présenté au chapitre 3 pour les codes GLD convolutifs. Par abus de langage, nous désignons un tel entrelacement par "entrelacement quaternaire" et le processus de décodage par "décodage itératif quaternaire".

4.5.5 Application au décodage itératif quaternaire

Soit $\mathcal{C}(N, K)$ un code composite construit à partir de J codes constituants isotropes et identiques, à une permutation près. Les $J - 1$ entrelaceurs du code composite permutent les bits deux à deux, c'est-à-dire symbole quaternaire par symbole quaternaire.

Lorsque l'on considère de tels entrelaceurs, la représentation graphique du code composite $\mathcal{C}(N, K)$ reste quasi-identique à celle d'un code composite construit à partir d'entrelaceurs bit à bit, si ce n'est que la nature des sommets du graphe et leurs degrés changent. En effet, le graphe de Tanner du code $\mathcal{C}(N, K)$ contient toujours deux types de sommets, dont les codes constituants, mais les bits sont maintenant transformés en symboles (quaternaires au lieu de binaires). Le degré de chaque code d_c est donc divisé par deux par rapport au graphe du code composite «binaire», tandis que le degré de chaque symbole, noté d_s, reste égal au nombre de codes constituants J. Une fenêtre de dépendance peut être considérée autour de chaque symbole pour déterminer le degré de chaque code. L'algorithme de décodage est l'algorithme aller-retour adapté pour l'entrelacement quaternaire et présenté au chapitre 3.

Afin d'évaluer les limites asymptotiques d'un code composite construit à partir d'un entrelaceur quaternaire, nous pouvons appliquer les deux méthodes précédemment décrites. La méthode de l'évolution de la densité, bien que donnant la propagation exacte de l'information et l'évolution de la probabilité d'erreur au cours des itérations, devient relativement complexe lorsque des symboles quaternaires sont considérés. En effet, il est maintenant nécessaire de construire trois histogrammes pour chaque type d'information, au lieu d'un seul dans le cas binaire ; le nombre de tirages utiles pour obtenir ces histogrammes avec précision augmente ainsi de manière non négligeable. Nous proposons donc d'utiliser la méthode de la fonction de transfert pour déterminer les limites du décodage itératif quaternaire appliqué à un ensemble de codes composites. Cette étude a été menée en collaboration avec Sawaya, Boutros et Zémor [91].

Soit $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ un mot du code composite $\mathcal{C}$, par exemple le mot $(0, \ldots, 0)$. Les symboles codés sont modulés par une modulation BPSK et émis sur un canal AWGN, dont le bruit est de moyenne nulle et de variance N_0. La séquence des symboles modulés est notée $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ et $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ est le vecteur réel correspondant, disponible à la sortie du canal. Nous supposons que tous les bits codés sont entrelacés deux à deux. Il s'agit, par exemple, d'un code GLD convolutif ou d'un code GLD blocs. Le cas où seuls les bits d'information sont entrelacés, comme pour un turbo-code parallèle, ne sera pas envisagé mais peut cependant être très facilement déduit de cette étude [91].

Lorsque deux bits codés c_{2i-1} et c_{2i} sont combinés pour former le symbole quaternaire S_i, le vecteur des observations du canal $\mathbf{y}$ peut être réécrit sous la forme $\mathbf{Y} = (Y_1, \ldots, Y_{N/2})$ avec $Y_i = (y_{2i-1}\ y_{2i})$. Nous supposons que l'étiquetage entre les entiers $q = 0, 1, 2, 3$ et les couples $(q_1\ q_2)$ désignant un symbole, est donné par :

$$\begin{array}{ccc} 0 & \rightarrow & 0\,0 \\ 1 & \rightarrow & 0\,1 \\ 2 & \rightarrow & 1\,0 \\ 3 & \rightarrow & 1\,1 \end{array}$$

Par analogie avec le cas binaire, nous définissons, pour un symbole quaternaire S_i ($S_i \in 0, 1, 2, 3$), les rapports logarithmiques de probabilités suivants :

– **Vecteur des rapports logarithmiques des APP partielles :**
$\boldsymbol{\mathcal{B}}(S_i) = (\mathcal{B}^1(S_i), \mathcal{B}^2(S_i), \mathcal{B}^3(S_i))$ avec

$$\mathcal{B}^q(S_i) = \log \frac{\mathcal{A}(S_i = q)}{\mathcal{A}(S_i = 0)}, \quad q = 1, 2, 3.$$

– **Vecteur des rapports logarithmiques des informations extrinsèques :**

$\mathbf{extLR}(S_i) = (extLR^1(S_i), extLR^2(S_i), extLR^3(S_i))$ avec

$$extLR^q(S_i) = \log \frac{Ext(S_i = q)}{Ext(S_i = 0)}, \quad q = 1, 2, 3.$$

- **Vecteur des rapports logarithmiques des APP totales :**
  $\mathbf{LR}(S_i) = \boldsymbol{\mathcal{B}}(S_i) + \mathbf{extLR}(S_i)$ avec

$$LR^q(S_i) = \log \frac{APP(S_i = q)}{APP(S_i = 0)} = \mathcal{B}^q(S_i) + extLR^q(S_i), \quad q = 1, 2, 3.$$

- **Vecteur des rapports logarithmiques des observations du canal :**
  $\mathbf{LR_0}(S_i) = (LR_0^1(S_i), LR_0^2(S_i), LR_0^3(S_i))$ avec

$$LR_0^q(S_i) = \log \frac{Pr(Y_i/S_i = q)}{Pr(Y_i/S_i = 0)}, \quad q = 1, 2, 3.$$

A la sortie du canal, les observations sur chaque symbole binaire x_i sont des variables indépendantes. L'observation du canal pour le symbole $S_i = (c_{2i-1}\ c_{2i})$ s'écrit alors, pour $q = 1,\ 2,\ 3$:

$$LR_0^q(S_i) = \log \frac{Pr(y_{2i-1}|q_1)}{Pr(y_{2i-1}|0)} + \log \frac{Pr(y_{2i}|q_2)}{Pr(y_{2i}|0)}\ . \tag{4.21}$$

où le couple $(q_1\ q_2)$ désigne les bits composant le symbole q. Ainsi, les rapports logarithmiques LR_0^1 et LR_0^2 correspondant respectivement à $q_1 = 0$ et $q_2 = 0$ sont des variables aléatoires gaussiennes indépendantes de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$. La variable aléatoire LR_0^3 est, quant à elle, égale à la somme de LR_0^1 et de LR_0^2. Les variables LR_0^1, LR_0^2 et LR_0^3 ne sont donc pas indépendantes entre elles. De même, les informations *a priori* fournies par les différents codes sur les symboles quaternaires sont supposées gaussiennes et indépendantes du fait de l'entrelaceur infini, mais les variables aléatoires $extLR_m^1$, $extLR_m^2$ et $extLR_m^3$ représentatives de ces informations ne sont cependant pas décorrélées.

Comme pour le cas binaire, nous considérons la probabilité d'erreur évaluée à partir des APP partielles des symboles pour déterminer les limites du décodage itératif d'un code composite. Évaluons maintenant sous ces hypothèses les fonctions de transfert d'un symbole et d'un code.

Fonction de transfert d'un code

A l'entrée d'un code, la probabilité d'erreur est définie par

$$\begin{aligned} Pe'_{\text{in}}(\text{code}) &= \Pr\left(\mathcal{B}^1 > 0 \ ou\ \mathcal{B}^2 > 0 \ ou\ \mathcal{B}^3 > 0\right) \\ &= 1 - \Pr\left(\mathcal{B}^1 < 0 \ et\ \mathcal{B}^2 < 0 \ et\ \mathcal{B}^3 < 0\right)\ . \end{aligned}$$

Les APP partielles $\mathcal{B}^q$, $q = 1, 2, 3$, sont des variables aléatoires gaussiennes, mais ne sont pas indépendantes comme évoqué ci-dessus. Le vecteur des APP partielles $\boldsymbol{\mathcal{B}}$ est donc caractérisé par son vecteur moyenne $\mathbf{m_v} = (m_1, m_2, m_3)$ et sa matrice de covariance Γ donnée par :

$$\Gamma = \begin{bmatrix} E[\mathcal{B}_c^1\mathcal{B}_c^1] & E[\mathcal{B}_c^1\mathcal{B}_c^2] & E[\mathcal{B}_c^1\mathcal{B}_c^3] \\ E[\mathcal{B}_c^2\mathcal{B}_c^1] & E[\mathcal{B}_c^2\mathcal{B}_c^2] & E[\mathcal{B}_c^2\mathcal{B}_c^3] \\ E[\mathcal{B}_c^3\mathcal{B}_c^1] & E[\mathcal{B}_c^3\mathcal{B}_c^2] & E[\mathcal{B}_c^3\mathcal{B}_c^3] \end{bmatrix},$$

où $E[\ ]$ désigne l'espérance mathématique, et $\mathcal{B}_c^q$, $q = 1,\ 2,\ 3$, est l'APP partielle du symbole centrée autour de sa valeur moyenne. La densité de probabilité du vecteur $\boldsymbol{\mathcal{B}}$ est :

$$p(\boldsymbol{\mathcal{B}}) = \frac{1}{\sqrt{2\pi}^3 \sqrt{det(\Gamma)}} \exp\left(-\frac{1}{2}\left(\boldsymbol{\mathcal{B}} - \mathbf{m}_v\right)\Gamma^{-1}\left(\boldsymbol{\mathcal{B}} - \mathbf{m}_v\right)^t\right) .$$

La probabilité d'erreur en entrée d'un code s'écrit alors sous la forme :

$$Pe'_{\text{in}}(\text{code}) = 1 - \int_{\mathcal{B}^1<0 \ \ et \ \ \mathcal{B}^2<0 \ \ et \ \ \mathcal{B}^3<0} p(\boldsymbol{\mathcal{B}}) d\mathcal{B}^1 d\mathcal{B}^2 d\mathcal{B}^3 .$$

Cette expression est particulièrement difficile à évaluer. Il est donc nécessaire d'introduire quelques hypothèses concernant l'indépendance des APP partielles $\mathcal{B}^q$, $q = 1, 2, 3$.

Pour une modulation BPSK, les observations $\mathcal{B}_0^1$ et $\mathcal{B}_0^2$ sont des variables aléatoires gaussiennes indépendantes de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$. Nous supposons qu'une hypothèse similaire peut être faite pour les APP partielles $\mathcal{B}^1$ et $\mathcal{B}^2$ à l'itération m : si $N_0(ap)$ désigne la variance de la probabilité extrinsèque d'un bit codé (et non d'un symbole quaternaire), les APP partielles $\mathcal{B}^1$ et $\mathcal{B}^2$ sont assimilées à des variables aléatoires gaussiennes indépendantes de moyenne $-\frac{2}{N_0} - \frac{2(d_s-1)}{N_0(ap)}$ et de variance $\frac{4}{N_0} + \frac{4(d_s-1)}{N_0(ap)}$. L'APP partielle $\mathcal{B}_m^3$ s'écrit alors comme la somme de deux variables aléatoires gaussiennes indépendantes :

$$\mathcal{B}^3 = \mathcal{B}^1 + \mathcal{B}^2 .$$

La probabilité d'erreur à l'entrée du code se simplifie en :

$$\begin{aligned} Pe'_{\text{in}}(\text{code}) &= 1 - \Pr(\mathcal{B}^1 < 0)\Pr(\mathcal{B}^2 < 0) \\ &= 1 - \left(1 - Q\left(\sqrt{\frac{1}{N_0} + \frac{d_s - 1}{N_0(ap)}}\right)\right)^2 \\ &\approx 2\,Q\left(\sqrt{\frac{1}{N_0} + \frac{d_s - 1}{N_0(ap)}}\right) . \end{aligned} \tag{4.22}$$

A la sortie du code, la probabilité d'erreur est donnée par :

$$Pe'_{\text{out}}(\text{code}) = \int_{\mathcal{B}^1>0 \ \ ou \ \ \mathcal{B}^2>0 \ \ ou \ \ \mathcal{B}^3>0} p(\boldsymbol{\mathcal{B}}) d\boldsymbol{\mathcal{B}} ,$$

où la distribution de $\boldsymbol{\mathcal{B}}$ est obtenue par simulation de l'algorithme aller-retour décrit au chapitre 3.

Fonction de transfert d'un symbole

Les APP partielles disponibles sur chacune des $(d_s - 1)$ arêtes à l'entrée d'un symbole sont définies comme la somme de deux variables gaussiennes indépendantes : l'observation du canal, de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$, et une information *a priori* délivrée par chaque code, de moyenne m_q et de variance σ_q^2 pour $q = 1,\ 2,\ 3$. Nous faisons les mêmes hypothèses que pour la fonction de transfert du code, c'est-à-dire, que les APP partielles $\mathcal{B}^1$ et $\mathcal{B}^2$ sont deux variables aléatoires indépendantes et identiquement distribuées selon une loi gaussienne de moyenne $-\frac{2}{N_0}-\frac{2}{N_0(ap)}$ et de variance $\frac{4}{N_0}+\frac{4}{N_0(ap)}$. La probabilité d'erreur en entrée d'un symbole s'écrit d'après l'approximation (4.22) :

$$Pe'_{\text{in}}(\text{symbole}) = 2\,Q\left(\sqrt{\frac{1}{N_0}+\frac{1}{N_0(ap)}}\right)\ .$$

Le symbole calcule une nouvelle APP partielle en sommant l'observation du canal et les (d_s-1) informations *a priori* indépendantes fournies par les codes. Les APP partielles $\mathcal{B}^1$ et $\mathcal{B}^2$ en sortie d'un symbole sont ainsi des variables aléatoires gaussiennes de moyenne $-\frac{2}{N_0}-\frac{2(d_s-1)}{N_0(ap)}$ et de variance $\frac{4}{N_0}+\frac{4(d_s-1)}{N_0(ap)}$. La probabilité d'erreur à la sortie d'un symbole est donc :

$$Pe'_{\text{out}}(\text{symbole}) = 2\,Q\left(\sqrt{\frac{1}{N_0}+\frac{d_s-1}{N_0(ap)}}\right)\ .$$

Notons que pour $d_s=2$, la fonction de transfert d'un symbole est définie par :

$$Pe'_{\text{in}}(\text{symbole}) = Pe'_{\text{out}}(\text{symbole})\ .$$

Pour étudier le comportement du décodage itératif, les fonctions de transfert des symboles et des codes sont tracées sur le même diagramme. La figure 4.13 représente le diagramme obtenu pour un code GLD de rendement $R=\frac{1}{2}$, construit à partir de deux codes convolutifs isotropes, lorsque le rapport signal-à-bruit est 1.2 dB. Ces codes sont de rendement $r=\frac{3}{4}$ et sont définis par les polynômes générateurs $(13,15,16,17)$. L'évolution de la probabilité d'erreur s'obtient de nouveau en zigzaguant entre les deux courbes représentant les fonctions de transfert du code et du symbole. Nous observons sur cette figure la présence d'un point critique dans la convergence de la probabilité d'erreur, aux alentours de $Pe'_{\text{in}}=0.18$: la propagation de l'information est alors beaucoup plus lente et nécessite de nombreuses itérations. Au-delà de ce goulet d'étranglement la convergence est nettement plus rapide, et très peu d'itérations sont requises pour atteindre une probabilité d'erreur nulle.

La méthode de la fonction de transfert nous permet de déterminer les limites de différents codes composites construits à partir d'entrelaceurs symbole par symbole, en observant l'évolution du point fixe Pe'. Pour des raisons de complexité, nous n'avons considéré que l'entrelacement par symbole quaternaire. Le tableau 4.7 donne les limites obtenues, en termes de rapport signal-à-bruit minimal, pour un turbo-code parallèle et un code GLD convolutif de rendements $\frac{1}{2}$, construits à partir de deux codes RSC. Nous observons dans ce tableau qu'un gain de l'ordre

Code Composite	Code constituant $C(n,k)$	R	Capacité (ML)	$\left(\frac{E_b}{N_0}\right)_{\min}$ (itératif)
Turbo-code	$RSC(31,21,37)$ $(r=2/3)$	1/2	0.18 dB	0.51 dB
Turbo-code	$RSC(37,21,27)$ $(r=2/3)$	1/2	0.18 dB	0.46 dB
Turbo-code	$RSC(25,37,35)$ $(r=2/3)$	1/2	0.18 dB	0.40 dB
GLD convolutif	$RSC(13,15,16,17)$ $(r=2/3)$	1/2	0.18 dB	0.80 dB

TAB. 4.7 – Limites asymptotiques de codes composites “quaternaires”

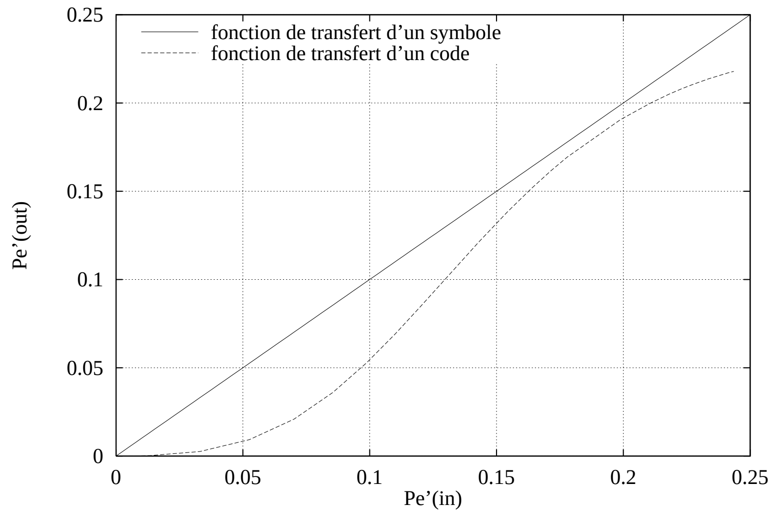


Fig. 4.13 – Propagation de l'information pour un décodage "quaternaire" d'un code GLD convolutif

de 0.2 dB peut être réalisé avec un entrelacement quaternaire pour les turbo-codes parallèle, et de l'ordre de 0.60 dB pour les codes GLD convolutifs. Ceci confirme les résultats de simulations obtenus au chapitre 3 avec un tel entrelaceur.

Signalons que dans [91], les estimations des limites des codes composites avec entrelacement quaternaire données par la méthode de la fonction de transfert ont été comparées aux valeurs données par la méthode de l'évolution de la densité. Les écarts trouvés entre les estimations sont de l'ordre de ceux signalés dans le cas binaire, c'est-à-dire de l'ordre de quelques centièmes de décibel, ce qui permet de valider les différentes approximations faites au cours de cette étude.

Nous avons présenté dans cette section une méthode graphique permettant d'évaluer les performances ultimes d'un ensemble de codes composites décodés itérativement. Cette technique constitue une alternative simple à la méthode de l'évolution de la densité. Dans l'ensemble des exemples considérés, nous avons supposé que les codes constituants du code composite étaient identiques à une permutation près. Il est possible d'étendre cette méthode à des codes constituants différents, en traçant les fonctions de transfert de chaque code et la fonction de transfert d'un bit sur la même figure. La propagation de la probabilité d'erreur au cours des itérations s'obtient alors en évoluant entre les différentes fonctions de transfert représentées. Lorsque le nombre de codes constituants est $J = 2$, la fonction de transfert du bit laisse invariante la probabilité d'erreur et peut ainsi être omise. La propagation de la probabilité d'erreur s'obtient alors en évoluant entre les fonctions de transfert des codes uniquement.

L'introduction de codes constituants différents dans la construction d'un code composite crée une irrégularité au niveau des codes, qui n'ont plus tous le même degré. Les turbo-codes série sont un exemple de tels codes. Nous proposons dans la section suivante d'introduire une irrégularité dans le graphe d'un code composite au niveau des bits, à travers un exemple de turbo-code irrégulier.

4.6 Turbo-codes Irréguliers

En observant la figure 4.10, qui analyse la propagation de la probabilité d'erreur au cours des itérations pour un turbo-code parallèle régulier ($J = 2$), nous constatons que pour améliorer les limites asymptotiques de ce code, il faut augmenter la distance entre les fonctions de transfert au voisinage du point critique. Pour cela, deux possibilités peuvent être envisagées :

1. changer la fonction de transfert du code, *i.e.*, changer le code constituant du code composite ou introduire des codes constituants différents
2. changer la fonction de transfert du bit, *i.e.*, changer le degré des bits en introduisant éventuellement différents degrés

La première éventualité a été évoquée lors de l'optimisation des codes constituants, dans la section 4.5.4, ou par le biais des turbo-codes série. Nous nous concentrons, dans cette section, sur la mise en œuvre de la seconde alternative. Cette dernière correspond à la définition des turbo-codes irréguliers donnée par [41].

Dans un premier temps, nous introduisons brièvement le schéma de codage des turbo-codes irréguliers et le principe du décodage itératif appliqué à ces codes. Puis nous montrons comment adapter la méthode de l'évolution de la densité et la méthode de la fonction de transfert afin de déterminer les limites du décodage itératif d'un turbo-code irrégulier. Ce turbo-code est caractérisé par un profil de degrés donnant la fraction de bits de degré d pour différentes valeurs de d. L'étude de la propagation de la probabilité d'erreur est ensuite utilisée pour optimiser le profil de degrés du code.

4.6.1 Définition et décodage itératif des turbo-codes irréguliers

Schéma de codage

Soit $\mathcal{C}_R(N_R, K_R)$ un turbo-code parallèle régulier construit à partir de J codes constituants identiques isotropes C . La représentation la plus usuelle de ce turbo-code est donnée sur la figure 4.14.

L'ordre des bits systématiques importe peu ; par conséquent, il est possible d'ajouter un entrelaceur à l'entrée du premier codeur. Le turbo-code $\mathcal{C}_R$ peut alors être vu comme la concaténation d'un code à répétition de rendement $\frac{1}{J}$, d'un entrelaceur agissant sur l'ensemble des bits répétés, et du code convolutif C, comme le montre la figure 4.15. Ainsi, pour le cas particulier d'un turbo-code parallèle construit à partir de deux codes RSC identiques, les bits d'information sont dupliqués, entrelacés puis envoyés à l'entrée du code convolutif RSC. Les performances de ce schéma de codage sont identiques aux performances du schéma classique de la figure 4.14 pour des tailles d'entrelaceur garantissant l'indépendance des répliques.

Dans un turbo-code régulier, tous les bits d'information sont répétés un nombre identique de fois égal au nombre de codes constituants, c'est-à-dire au degré d'un bit d'information $d_b = J$. Pour rendre ce turbo-code irrégulier, il suffit, comme pour les codes LDPC [57][66], de changer le degré de certains bits. Nous définissons ainsi un profil de degrés $\{f_d\}_{d=1,\dots,D}$ donnant la fraction

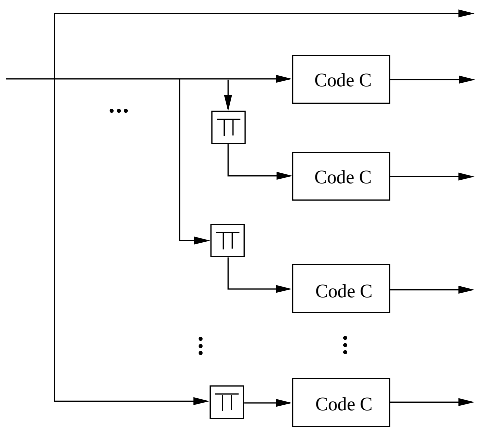


Fig. 4.14 – Représentation “classique” d'un turbo-code parallèle construit à partir de J codes

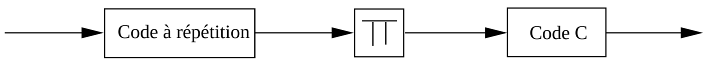


Fig. 4.15 – Représentation “condensée” d'un turbo-code parallèle construits à partir de J codes

de bits codés de degré d si D est le degré maximal des bits codés. Ces fractions vérifient l'égalité suivante :

$$\sum_{d=1}^{D} f_d = 1 \ .$$

Pour un turbo-code irrégulier de rendement R_I, la fraction de bits de parité, *i.e.*, de bits de degré 1, est :

$$f_1 = 1 - R_I \ .$$

La sortie du turbo-code irrégulier $\mathcal{C}_I(N_I, K_I)$, défini par ce profil de degrés, est perforée de manière à ne garder qu'une seule réplique des bits d'information. Soit $\bar{d}$ le degré moyen des bits codés, défini par :

$$\bar{d} = \sum_{d=1}^{D} d \ f_d \ .$$

Si r est le rendement du code constituant C, alors le rendement R_I du turbo-code irrégulier $\mathcal{C}_I$ est obtenu à l'aide de l'égalité ci-dessous :

$$\bar{d}(1 - r) = (1 - R_I) \ .$$

Cette relation montre que pour un degré moyen $\bar{d}$ et un rendement R_I du turbo-code fixés, il faut poinçonner le code constituant afin d'augmenter son rendement r. La perforation se fait sur les bits de degré 1. En pratique, il est nécessaire de faire un compromis entre le degré moyen (c'est-à-dire le profil des degrés) et le rendement total du turbo-code : en effet, si le degré moyen est trop élevé alors le code constituant doit être perforé de manière importante et risque ainsi de perdre de son efficacité. Ceci encourage l'utilisation des turbo-codes irréguliers pour de faibles rendements.

Exemple 2
Considérons un turbo-code irrégulier de rendement $R_I = \frac{1}{3}$ *construit à partir d'un code convolutif de rendement initial* $r = \frac{1}{2}$. *Le profil de degrés du turbo-code irrégulier est supposé être le suivant :*

$$\begin{aligned} d = 1 \quad &: \quad f_1 = \frac{2}{3} \\ d = 2 \quad &: \quad f_2 = \frac{91}{300} \\ d = 10 \quad &: \quad f_{10} = \frac{3}{100} \end{aligned}$$

Afin de garantir un rendement $R_I = \frac{1}{3}$ *pour le turbo-code irrégulier, le rendement du code convolutif doit être égal à* $r \approx 0.576$ *au lieu de* $\frac{1}{2}$ *initialement.* 26.5% *des bits de parité doivent donc être éliminés, ce qui reste raisonnable. En revanche, si l'on considère le profil de degrés ci-dessous :*

$$\begin{aligned} d = 1 \quad &: \quad f_1 = \frac{2}{3} \\ d = 2 \quad &: \quad f_2 = \frac{41}{150} \\ d = 10 \quad &: \quad f_{10} = \frac{3}{100} \\ d = 20 \quad &: \quad f_{20} = \frac{3}{100} \end{aligned}$$

le rendement du code convolutif doit être égal à $r \approx 0.685$ *pour maintenir le rendement du turbo-code. Ceci signifie que* 53.9% *des bits de parité sont poinçonnés en sortie du code convolutif, ce qui risque de limiter l'efficacité du code convolutif. Nous verrons dans la partie 4.6.4 comment optimiser le compromis entre le profil de degrés et le rendement du code constituant afin de garantir les meilleures performances.*

La représentation graphique d'un turbo-code irrégulier est relativement similaire à celle d'un turbo-code régulier. L'entrelacement présent dans le turbo-code irrégulier est supposé de longueur infinie : les répliques d'un bit d'information sont ainsi suffisamment éloignées par rapport à la fenêtre de dépendance du code constituant pour considérer qu'elles appartiennent à des codes différents. Plusieurs voisinages de bits (autant que de degrés) coexistent maintenant dans le graphe, selon le degré des bits : un bit de degré d est ainsi relié à d codes indépendants. Les bits de degré 1 (bits de parité) ne propagent aucune information dans le graphe. Les codes ont quant à eux tous le même voisinage ; une arête reliée à un code provient d'un bit de degré d, $d = 2, \ldots, D$, avec une probabilité μ_d telle que :

$$\mu_d = \frac{d\ f_d}{\sum_{d=2}^{D} d\ f_d}\ . \tag{4.23}$$

Le turbo-code irrégulier n'est pas isotrope. Il faut, par conséquent, étudier la distribution de probabilité de l'APP partielle et de l'APP totale de chaque bit. En fait, il suffit de considérer un nombre de distributions de probabilité égal au nombre de voisinages différents présents dans le graphe, c'est-à-dire au plus $D - 1$ distributions.

Décodage itératif

Soit $\mathcal{C}_I(N, K)$ un turbo-code irrégulier construit à partir du code convolutif récursif systématique C et dont le profil de degrés est $\{f_d\}_{d=1,\dots,D}$. Un mot de code de $\mathcal{C}_I(N, K)$ est noté $\mathbf{c} = (c_1, \dots, c_N)$, et $\mathbf{x} = (x_1, \dots, x_N)$ est la séquence modulée correspondante (modulation BPSK). Nous considérons un canal AWGN, dont le bruit est de moyenne nulle et de variance N_0 ; la sortie du canal est notée $\mathbf{y} = (y_1, \dots, y_N)$. Le vecteur $\mathbf{y}$ est fourni à un décodeur SISO utilisant l'algorithme aller-retour.

A partir du vecteur $\mathbf{y}$, le décodeur calcule les observations du canal définies par les probabilités $\Pr(y_i|x_i)$, $i = 1, \dots, N$. Pour un bit codé c_i de degré d, il copie l'observation $\Pr(y_i|x_i)$ dans les probabilités $P^j(y_i|x_i)$, $j = 1, \dots, d$. A la première itération, les informations *a priori* $P(c_i), i = 1, \dots, N$, sont initialisées à 0.5 et sont copiées dans les probabilités $P^j(c_i)$. Les répliques des observations du canal et des probabilités *a priori* sont utilisées par le décodeur aux emplacements des bits d'information poinçonnés dans la séquence codée. En effet, contrairement au schéma de codage de la figure 4.14, le turbo-code irrégulier introduit par [41] est composé d'un seul code RSC. Par conséquent, un seul décodeur SISO élémentaire est considéré en réception pour le décodage itératif ; l'algorithme aller-retour est alors appliqué sur la séquence codée entière.

Les répliques d'un bit d'information c_i sont supposées être suffisamment éloignées par l'entrelaceur dans la séquence codée pour garantir l'indépendance des probabilités extrinsèques $Ext^j(c_i)$, $j = 1, \dots, d$, générées sur ces répliques par le décodeur SISO. Ces informations extrinsèques sont combinées à l'observation du canal pour former la probabilité *a posteriori* du bit c_i à l'itération m :

$$APP(c_i) = \Pr(y_i|x_i) \prod_{j=1}^{d} Ext^j(c_i) \ . \tag{4.24}$$

L'information *a priori* fournie à l'itération $m + 1$ pour la j-ième réplique du bit c_i est donnée par :

$$P^j(c_i) = \prod_{\substack{\ell=1 \\ \ell \neq j}}^{d} Ext^\ell(c_i) \ . \tag{4.25}$$

Chaque itération de décodage se déroule selon les étapes suivantes :

1. Calcul des APP partielles des bits d'information et de leurs répliques en combinant l'observation du canal avec la probabilité *a priori* du bit évaluée selon l'équation (4.25),
2. Permutation de ces informations souples pour les fournir au décodeur SISO,
3. Décodage par l'algorithme aller-retour,
4. Permutation des sorties souples de l'algorithme aller-retour,
5. Évaluation de l'APP de chaque bit d'information en prenant en compte toutes ses répliques, selon l'équation (4.24) .

Comme pour les turbo-codes réguliers, il est possible d'étudier les performances asymptotiques d'un ensemble de turbo-codes irréguliers de même rendement et de même profil de degrés, décodés itérativement. Nous expliquons dans les sections suivantes comment appliquer les méthodes présentées aux sections 4.3 et 4.5 lorsque le graphe de Tanner du code composite est irrégulier. Par la suite, les informations souples se propageant dans le graphe sont des rapports logarithmiques de probabilités.

4.6.2 Méthode de l'évolution de la densité pour les turbo-codes irréguliers

Nous considérons un ensemble de turbo-codes irréguliers de rendement R, construits à partir d'un code RSC noté $C(n,k)$, de polynômes générateurs donnés et d'un profil de degrés $\{f_d\}_{d=1,\dots,D}$, où D est le degré maximal des bits codés. Le code constituant C est supposé isotrope. Nous faisons l'hypothèse que le mot $\mathbf{c} = (0,\dots,0)$ est modulé en $\mathbf{x} = (-1,\dots,-1)$ puis émis sur un canal gaussien, dont le bruit est de variance N_0.

Une formule de propagation similaire à la formule (4.9) peut être obtenue en incluant le profil de degrés du turbo-code irrégulier. Comme pour les codes composites réguliers, cette formule est trop complexe pour être appliquée directement ; par conséquent, nous cherchons les performances limites de l'ensemble des turbo-codes irréguliers en déterminant par simulation (méthode Monte-Carlo) les différentes distributions des informations extrinsèques et des probabilités *a posteriori*.

Le turbo-code irrégulier n'est pas isotrope ; soit δ_a son degré d'anisotropie. Comme le code constituant C est isotrope, δ_a est égal au nombre de degrés distincts dans le profil du turbo-code irrégulier et est majoré par $D-1$. Nous devons donc évaluer au plus $D-1$ distributions différentes des probabilités *a posteriori* afin de pouvoir calculer la probabilité d'erreur à l'itération m. Cette probabilité d'erreur $P\mathrm{e}_m$ est égale à :

$$P\mathrm{e}_m = \sum_{d=2}^{D} \frac{f_d}{\sum_{i=2}^{D} f_i} P\mathrm{e}_m(d) \ , \tag{4.26}$$

où $P\mathrm{e}_m(d)$ est la probabilité d'erreur à l'itération m évaluée à partir des bits de degré d :

$$P\mathrm{e}_m(d) = \int_0^{+\infty} p_m^d(x)dx \ .$$

Dans cette équation, p_m^d désigne la densité de probabilité du rapport logarithmique des APP à l'itération m pour un bit de degré d. Pour déterminer les densités de probabilité p_m^d, il est nécessaire d'observer le voisinage d'un bit u_d de degré d, pour $d = 2,\dots,D$.

A l'entrée de chaque code, l'observation du canal est une variable aléatoire gaussienne de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$. Un bit relié à ce sommet est un bit de degré d avec une probabilité μ_d ; son information *a priori* s'écrit alors comme le produit de $d-1$ informations extrinsèques délivrées par les autres codes. Pour chaque bit u_d, nous appliquons l'algorithme aller-retour sur une fenêtre autour de u_d égale à la fenêtre de dépendance W du code. L'algorithme génère une information extrinsèque sur u_d dont nous évaluons la distribution. En fait, quel que soit le bit considéré u_d et son degré d, les informations extrinsèques à la sortie du décodeur sont indépendantes des informations *a priori* et ont par conséquent toutes la même distribution puisque le code C est isotrope. Un seul histogramme est donc nécessaire pour évaluer la distribution des informations extrinsèques $extLR_m$ de tous les bits u_d, $d = 2,\dots,D$. A l'itération suivante, l'information *a priori* d'un bit de degré d est obtenue à partir de la somme de $d-1$ informations extrinsèques distribuées selon la loi de $extLR_m$.

La distribution de l'APP est, quant à elle, différente selon le degré d du bit u_d. D'après l'équation (4.24), la probabilité *a posteriori* du bit u_d est la somme de l'observation et de d informations extrinsèques distribuées selon la loi de $extLR_m$. Par conséquent, pour chaque bit u_d, un histogramme Hist_d est nécessaire pour évaluer la distribution p_m^d de son APP. Pour remplir l'histogramme Hist_d, pour $d = 2, \ldots, D$, nous répétons les étapes suivantes jusqu'à obtenir une précision suffisante :

1. Génération d'un échantillon *obs* selon une distribution gaussienne de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$,
2. Génération de d échantillons ext_i, $i = 1, \ldots, d$, selon la distribution de $extLR_m$,
3. Somme de l'échantillon *obs* et des d échantillons ext_i, $i = 1, \ldots, d$,
4. Stockage de l'échantillon obtenu dans l'histogramme Hist_d.

A l'aide des $D-1$ histogrammes obtenus, nous pouvons évaluer les probabilités d'erreur $Pe_m(d)$ puis la probabilité d'erreur totale Pe_m à l'itération m.

La table 4.8 donne les performances limites de deux turbo-codes irréguliers dont le profil est constitué des degrés 1, 2, et 10 uniquement. Les variables R_I et r désignent respectivement le rendement du turbo-code irrégulier et celui de son code constituant poinçonné. Les valeurs indiquées entre parenthèses dans la 5ème et la 6ème colonnes du tableau correspondent aux limites du turbo-code régulier à deux niveaux construit à partir du même code RSC et de même rendement. Nous observons que le gain apporté par l'introduction d'irrégularité dans le graphe de Tanner du turbo-code est proche de 0.20 dB.

R_I	Polynômes générateurs	Profil de degrés	r	$\left(\frac{E_b}{N_0}\right)_{\min}$ (itératif)	$(N_0)_{\max}$
1/3	$(37, 21)$	$f_1 = 0.6667$ $f_2 = 0.3033$ $f_{10} = 0.03$	0.576	-0.38dB (-0.15 dB)	1.63716 (1.552712)
1/7	$(31, 27, 37, 21)$	$f_1 = 0.8572$ $f_2 = 0.1328$ $f_{10} = 0.01$	0.299	-0.81 dB (-0.66 dB)	4.21762 (4.07444)

TAB. 4.8 – Limites asymptotiques des turbo-codes irréguliers

4.6.3 Méthode de la fonction de transfert pour les turbo-codes irréguliers

Afin de visualiser graphiquement le décodage itératif des turbo-codes irréguliers, nous proposons dans cette section de nous intéresser aux fonctions de transfert des différents sommets du graphe (H est la fonction de transfert d'un code, tandis que G est celle d'un bit). Nous supposons que les informations extrinsèques sont des variables aléatoires gaussiennes indépendantes de moyenne $-\frac{2}{N_0(ap)}$ et de variance $\frac{4}{N_0(ap)}$. Rappelons que l'observation du canal est également une variable aléatoire gaussienne de moyenne $-\frac{2}{N_0}$ et de variance $\frac{4}{N_0}$.

Soit $\mathcal{C}_I(N,K)$ un turbo-code irrégulier dont le profil de degrés est $\{f_d\},\ d=1,\dots,D$. A partir de ce profil et de l'équation (4.23) nous déduisons les valeurs des probabilités $\{\mu_d\},\ d=1,\dots,D$. Évaluons maintenant les fonctions de transfert de chaque sommet du graphe. Les probabilités d'erreur calculées en entrée et en sortie de chaque sommet sont évaluées à partir de l'APP partielle des bits.

Au niveau d'un code

A l'entrée d'un code, les bits sont de degrés d avec une probabilité μ_d, pour $d=2,\dots,D$. L'APP partielle d'un bit de degré d est la somme de d variables aléatoires gaussiennes indépendantes : une observation et $d-1$ probabilités *a priori*. La probabilité d'erreur à l'entrée d'un code est donnée par :

$$Pe'_{\text{in}}(\text{code}) = \sum_{d=2}^{D} \mu_d Q\left(\sqrt{\frac{1}{N_0} + \frac{(d-1)}{N_0(ap)}}\right) .$$

A la sortie du code, la probabilité d'erreur est :

$$Pe'_{\text{out}}(\text{code}) = \Pr(\mathcal{B} > 0) ,$$

où la distribution de $\mathcal{B}$ est obtenue par simulation Monte-Carlo.

Au niveau d'un bit

A la sortie du décodeur, toutes les informations extrinsèques générées ont la même distribution quel que soit le degré des bits. A l'entrée d'un bit u_d, l'APP partielle est la somme de l'observation et d'une information extrinsèque délivrée par le décodeur. Par conséquent, la probabilité d'erreur s'écrit :

$$Pe'_{\text{in}}(\text{bit}) = Q\left(\sqrt{\frac{1}{N_0} + \frac{1}{N_0(ap)}}\right) .$$

Le bit considéré possède un degré d dans le graphe avec une probabilité μ_d. Il combine les $d-1$ informations extrinsèques disponibles à son entrée avec l'observation du canal pour générer une nouvelle APP partielle. L'APP partielle ainsi évaluée est une variable aléatoire gaussienne de moyenne $-\frac{2}{N_0} - \frac{2(d-1)}{N_0(ap)}$ et de variance $\frac{4}{N_0} + \frac{4(d-1)}{N_0(ap)}$. La probabilité d'erreur en sortie d'un bit est :

$$Pe'_{\text{out}}(\text{bit}) = \sum_{d=2}^{D} \mu_d Q\left(\sqrt{\frac{1}{N_0} + \frac{(d-1)}{N_0(ap)}}\right) .$$

La figure 4.16 représente les fonctions de transfert H et G d'un turbo-code irrégulier de rendement $R=\frac{1}{3}$ construit à partir d'un code RSC de polynômes générateurs $(37,21)$ et dont le profil de degrés est $(f_1 = 0.6667\ ;\ f2 = 0.3033\ ;\ f_{10} = 0.03)$. Le rapport signal-à-bruit sur cette figure est $\frac{E_b}{N_0} = -0.10$ dB. Contrairement aux turbo-codes réguliers, nous ne remarquons aucun point critique dans la propagation de l'information, et les limites des performances du décodage itératif de ce code ont été améliorées par rapport à sa version régulière. Le nouveau seuil est, d'après la méthode de l'évolution de la densité, voisin de -0.38 dB. Nous voyons également sur cette figure que l'absence de point critique rend la convergence vers zéro plus lente et plus uniforme tout au long de la propagation que pour un turbo-code régulier : la distance entre les deux fonctions de transfert reste en effet constante au fil des itérations. La figure 4.17 représente les fonctions H et G pour un rapport signal à bruit de -0.35 dB, c'est-à-dire très proche du seuil de ce turbo-code irrégulier. Nous observons que pour ce rapport signal-à-bruit, la méthode de la fonction de transfert indique que le point fixe Pe' n'est pas nul, contrairement à la méthode de l'évolution de la densité. Nous avons constaté un écart de 0.06 dB entre les estimations faites par les deux méthodes de la limite de ce turbo-code irrégulier.

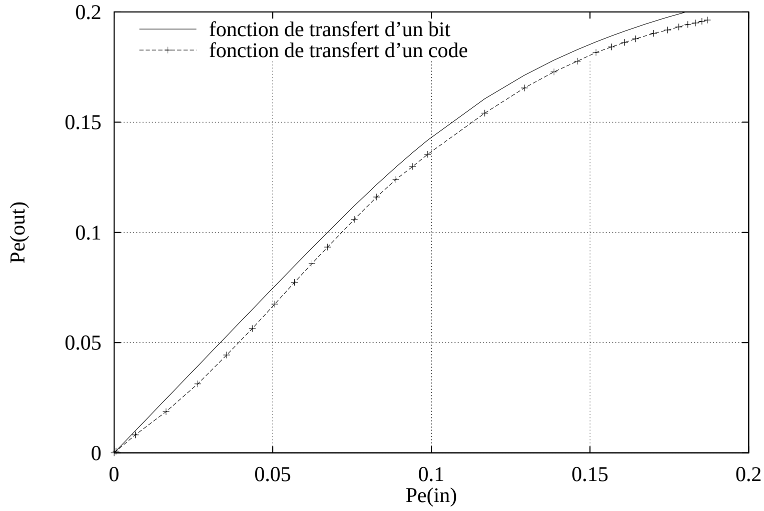


FIG. 4.16 – Propagation de la probabilité d'erreur pour un turbo-code irrégulier, $\frac{E_b}{N_0} = -0.10$ dB

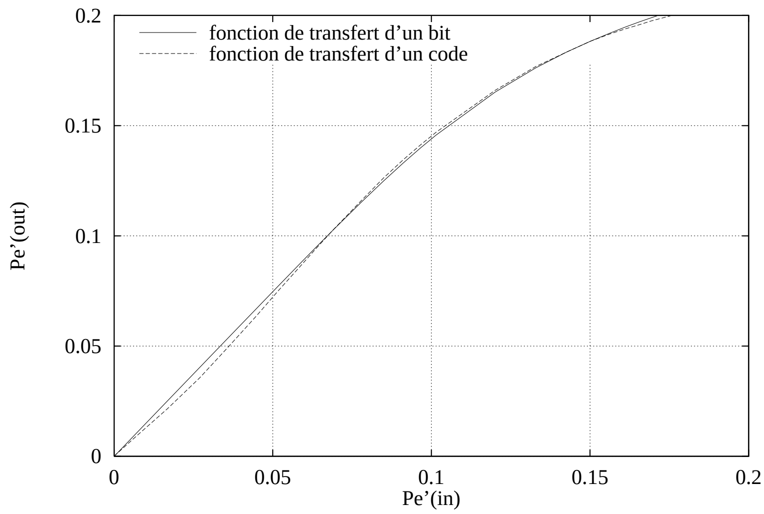


FIG. 4.17 – Propagation de la probabilité d'erreur pour un turbo-code irrégulier, $\frac{E_b}{N_0} = -0.35$ dB

Les performances des turbo-codes irréguliers varient en fonction du profil de degrés que l'on sélectionne. Ce profil conditionne le rendement du code constituant, et peut, s'il est mal choisi, diminuer l'efficacité du code RSC de telle sorte que le turbo-code irrégulier considéré a de moins bonnes performances que son équivalent régulier (*i.e.*, de même rendement et construit à partir du même code RSC avant poinçonnage). Nous proposons dans la section suivante d'optimiser le choix de ce profil de degrés en utilisant la méthode de l'évolution de la densité.

4.6.4 Optimisation du profil des turbo-codes irréguliers

Le choix d'un bon profil de degrés pour un turbo-code irrégulier dépend de plusieurs paramètres tels que le rendement du code constituant, le rendement du turbo-code irrégulier et les polynômes générateurs du code constituant. Nous proposons pour sélectionner de bons profils d'utiliser la méthode de l'évolution de la densité appliquée à un turbo-code de rendement fixé construit à partir d'un code RSC donné.

Soit $\mathcal{C}_{\mathcal{I}}$ un turbo-code irrégulier de rendement R_I construit à partir d'un code RSC de rendement r. Le profil de degrés de ce turbo-code est noté $\{f_d\}_{d=1,\ldots,D}$, où D est le degré maximal des bits codés.

Critère d'optimisation

En choisissant un bon profil de degrés, nous cherchons à améliorer le seuil du turbo-code $\mathcal{C}_{\mathcal{I}}$, c'est-à-dire à maximiser la variance du bruit pour laquelle la probabilité d'erreur tend vers zéro. Il s'agit donc d'optimiser le choix de $D+1$ variables : les D fractions f_d et le rendement r. Cette optimisation se fait sous plusieurs contraintes :

$$\sum_{d=1}^{D} f_d = 1 \tag{4.27}$$

$$\left(\sum_{d=1}^{D} d f_d\right)(1-r) = (1-R_I) \tag{4.28}$$

$$f_1 = 1 - R_I \qquad \text{et} \qquad 0 \leq f_i \leq R_I \tag{4.29}$$

Ces contraintes permettent de réduire le nombre de variables du problème. Nous passons ainsi de $D+1$ à $D-2$ variables à optimiser : les fractions $f_d,\ d = 3, \ldots, D$, les autres inconnues étant déduites des contraintes (4.27), (4.28), (4.29).

La méthode retenue pour choisir le profil de degrés est la suivante. Nous prenons comme profil initial celui du turbo-code régulier $\mathcal{C}_{\mathcal{R}}$ $(J = 2)$ construit à partir du code RSC, c'est-à-dire,

$$f_2 = R_I \quad \text{et} \quad f_d = 0 \ \ \forall\, d > 2\ .$$

Nous déterminons alors la limite de ce turbo-code en termes de variance de bruit, $N_0(\mathcal{C}_{\mathcal{R}})$, en appliquant la méthode de l'évolution de la densité, et pour un nombre fixé d'itérations m_0

nous évaluons la probabilité d'erreur Pe_{m_0} atteinte par $\mathcal{C}_{\mathcal{R}}$ pour cette variance de bruit. Puis la séquence de fractions du profil est modifiée à l'aide de la méthode du «downhill simplex» (appelée méthode DSM pour *Downhill Simplex Method*) [61] que nous détaillons ci-après. Si le nouveau profil considéré mène pour la même variance de bruit à une probabilité d'erreur plus faible au bout de m_0 itérations, alors ce profil est retenu et devient le nouveau profil de référence. L'opération est ensuite renouvelée jusqu'à ce que la probabilité d'erreur Pe_{m_0} atteinte n'évolue plus. Le seuil du dernier profil retenu est évalué. La recherche est alors relancée pour cette nouvelle variance de bruit et ainsi de suite, jusqu'à ce que la variance de bruit maximale reste inchangée pour les nouveaux profils sélectionnés.

Algorithme d'optimisation

Pour la modification des fractions, nous avons choisi la méthode DSM. Cette méthode s'applique aisément dans le cadre de l'optimisation d'une fonction de coût non linéaire multi-dimensionnelle. Si une seule fraction f_d avec $d \neq 2$ est à optimiser, il est possible de recourir à des méthodes d'optimisation globales relativement simples et efficaces. Nous nous limitons dans cette partie à la méthode DSM, qui permet de sélectionner $D-2$ inconnues indépendantes, sans connaissance *a priori* de la fonction à optimiser ni de ses dérivées. La fonction de coût est dans notre cas la probabilité d'erreur atteinte par le turbo-code irrégulier après m_0 itérations.

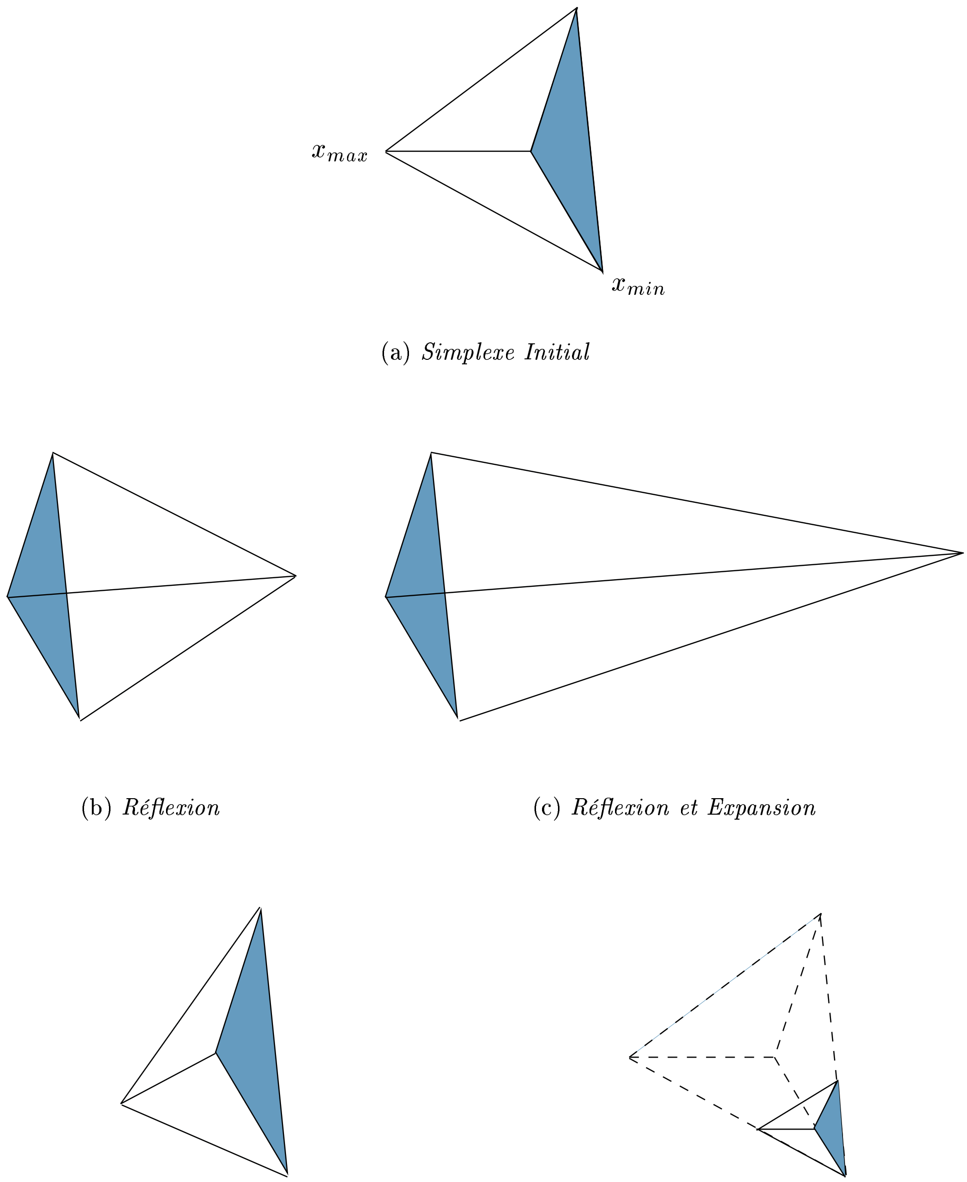
La méthode DSM a été introduite en 1965 par Nelder et Mead [61] pour optimiser un système à Q dimensions ($Q = D-2$ ici). Il s'agit de considérer un simplexe non dégénéré et de faire évoluer ce simplexe jusqu'à l'obtention d'un profil convenant au critère d'arrêt fixé pour notre recherche (*i.e.,* variance de bruit maximale).

Un simplexe est une figure géométrique à Q dimensions, contenant $Q+1$ sommets, $\frac{Q(Q+1)}{2}$ arêtes, et C_{Q+1}^{q+1} faces de dimension q, $q = 2, \ldots, Q$. En dimension 2, un simplexe est un triangle, tandis qu'il s'agit d'un tétraèdre non nécessairement régulier en dimension 3. Un simplexe non dégénéré délimite un volume fini en dimension Q. Ainsi, si nous définissons n'importe quel sommet comme origine, les Q autres sommets définissent les directions des vecteurs de l'espace à Q dimensions. Pour former ce simplexe, il faut tout d'abord sélectionner un point d'origine $\mathbf{x}_0$. Nous prenons par exemple pour $\mathbf{x}_0$ le profil de degrés du turbo-code régulier $\mathcal{C}_{\mathcal{R}}$. Les Q autres sommets sont déduits de $\mathbf{x}_0$ par la relation :

$$\mathbf{x}_q = \mathbf{x}_0 + \lambda \mathbf{e}_q \quad q = 1, \ldots, Q \ ,$$

où λ est une constante et $\mathbf{e}_q$, $q = 1, \ldots, Q$ sont des vecteurs unitaires.

La fonction de coût est évaluée en chacun des sommets du simplexe (figure 4.18(a)). Le sommet $x_{\max}$ ayant la fonction de coût la plus forte est réfléchi par rapport au centre de la face opposée du simplexe vers un point ayant un coût plus faible (figure 4.18(b)). La réflexion se fait de telle sorte que le volume du simplexe est conservé, ce qui permet de maintenir sa non-dégénérescence. Si le coût du nouveau point considéré est plus faible, une expansion du simplexe d'un facteur $f_{\exp}$ peut être réalisée dans la direction donnée par ce point (figure 4.18(c)). A chaque expansion, le facteur $f_{\exp}$ augmente tant que cela est possible. Si le coût du nouveau point est plus grand, alors le simplexe se rétracte selon la direction indiquée par le point, en cherchant un point de coût plus faible (figure 4.18(d)). Si de nouveau c'est un échec, le simplexe se contracte autour de son meilleur point $x_{\min}$, c'est-à-dire celui ayant le coût le plus faible (figure 4.18(e)). Ces étapes sont renouvelées jusqu'à l'obtention d'un minimum de la fonction de coût. Notons, que lors de ces différentes étapes, si les conditions données par les équations (4.27) et (4.28) ne sont pas vérifiées en un point, nous introduisons des fonctions de pénalité telles que la fonction de coût soit arbitrairement grande en ce point.



(a) *Simplexe Initial*

(b) *Réflexion* (c) *Réflexion et Expansion*

(d) *Contraction* (e) *Contraction Multiple*

Fig. 4.18 – Différentes étapes de la méthode DSM

Comme il s'agit d'une méthode d'optimisation locale, nous avons appliqué l'algorithme DSM à plusieurs reprises en changeant les conditions initiales, jusqu'à ce que le système se stabilise et sélectionne de nouveau le même profil. A chaque application de l'algorithme, nous avons utilisé le dernier profil retenu comme nouveau point initial $\mathbf{x}_0$, et observé si l'algorithme converge de nouveau vers ce même point ; la variance du bruit est déterminée par le seuil du turbo-code irrégulier de profil $\mathbf{x}_0$.

Résultats numériques

La table 4.9 donne quelques exemples de bons profils sélectionnés à l'aide de la méthode DSM, pour un turbo-code irrégulier de rendement $R_I = \frac{1}{3}$ construit à partir d'un code RSC de rendement initial $\frac{1}{2}$ et de polynômes générateurs $(37, 21)$. Le rendement r du code RSC après poinçonnage est précisé. Les limites asymptotiques de ces codes sont données en termes de rapport signal-à-bruit minimal et de variance de bruit maximale. Rappelons que la version régulière de ce turbo-code a pour limite asymptotique -0.15 dB. Nous constatons dans ce tableau que les rendements des codes RSC perforés sont tous inférieurs à 0.60, c'est-à-dire que moins d'un tiers des bits de parité sont poinçonnés.

4.7 Conclusion

Nous avons étudié dans ce chapitre les limites asymptotiques des codes composites décodés itérativement. Ces limites s'expriment en termes de rapport signal-à-bruit minimal ou de variance de bruit maximale, tels que la probabilité d'erreur tende vers zéro lorsque le nombre d'itérations de décodage est suffisant. Deux méthodes ont été introduites pour évaluer ces limites :

- *la méthode de l'évolution de la densité* : cette méthode est fondée sur l'analyse de l'évolution de la distribution des informations extrinsèques et des probabilités *a posteriori* au cours des itérations. Ces distributions permettent de déterminer la probabilité d'erreur à chaque itération.
- *la méthode de la fonction de transfert* : cette méthode repose sur l'approximation gaussienne des distributions des informations extrinsèques circulant dans le graphe du code composite considéré. Il s'agit alors de déterminer sous cette hypothèse les transformations engendrées par les sommets du graphe sur la probabilité d'erreur à chaque itération. Ces transformations sont représentées sur le même diagramme pour analyser graphiquement le décodage itératif d'un code composite et ses limites.

Les limites évaluées à l'aide de ces deux méthodes ont été comparées et les écarts trouvés (inférieurs à 0.10 dB) ont permis de valider les différentes approximations réalisées au cours de cette étude.

Ces techniques ont permis de mieux appréhender le comportement du décodage itératif et de proposer des voies d'amélioration des performances d'un code composite décodé itérativement. Elles ont été utilisées par exemple, pour optimiser le choix des polynômes constituants d'un turbo-code parallèle ou le profil de degrés d'un turbo-code irrégulier. Le critère de conception retenu a été l'optimisation des performances pour le décodage itératif et non pour le décodage à maximum de vraisemblance. Les performances asymptotiques de ces codes ont été comparées aux performances ML données par la capacité du canal.

Profil de degrés	r	$\left(\frac{E_b}{N_0}\right)_{\min}$ (itératif)	$(N_0)_{\max}$
$f_1 = 0.6667$ $f_2 = 0.3079$ $f_3 = 0.0003$ $f_4 = 0.0009$ $f_5 = 0.0242$	0.526	-0.18 dB	1.5634
$f_1 = 0.6667$ $f_2 = 0.2916$ $f_6 = 0.0417$	0.555	-0.26 dB	1.5925
$f_1 = 0.6667$ $f_2 = 0.2989$ $f_3 = 0.0004$ $f_4 = 0.0004$ $f_5 = 0.0003$ $f_6 = 0.0015$ $f_7 = 0.0318$	0.555	-0.32 dB	1.6147
$f_1 = 0.6667$ $f_2 = 0.3043$ $f_3 = 0.0004$ $f_4 = 0.0002$ $f_5 = 0.0003$ $f_6 = 0.0007$ $f_7 = 0.0013$ $f_8 = 0.0260$	0.555	-0.31 dB	1.6110
$f_1 = 0.6667$ $f_2 = 0.3033$ $f_{10} = 0.03$	0.576	-0.38 dB	1.6372
$f_1 = 0.6667$ $f_2 = 0.3033$ $f_{12} = 0.03$	0.592	-0.38 dB	1.6372
$f_1 = 0.6667$ $f_2 = 0.3033$ $f_9 = 0.0019$ $f_{12} = 0.0281$	0.590	-0.30 dB	1.6073

TAB. 4.9 – Bons profils de degrés pour un turbo-code irrégulier de rendement $R = 1/3$

Notons que l'utilisation de la fonction de transfert ne se limite pas aux codes composites. D'autres études ont été réalisées pour appliquer cette méthode à la compréhension et à l'évaluation des limites de tout système utilisant un processus itératif pour le décodage ou la démodulation [16][15][93].

Conclusions et Perspectives

Depuis 1993 et l'invention des turbo-codes, de nombreuses structures de codes composites ont été introduites (ou redécouvertes, comme les LDPC et les codes de Tanner) et exploitées. Ces codes présentent d'excellentes performances, très proches des limites données par la théorie de l'information.

Nous avons proposé dans ce document deux nouveaux schémas de codes composites.

Au chapitre 2, nous avons considéré des codes produits construits à partir de codes convolutifs à treillis circulaire comme une alternative aux codes produits classiques. Ces deux constructions ont des performances similaires pour de petites longueurs de code ($N < 1000$), mais notre construction atteint un gain proche de 1 dB pour de plus grandes longueurs ($N \sim 10000$). Ces codes possèdent une structure permettant le décodage simultané de chaque code constituant, afin de réduire le délai introduit par le décodage itératif. D'autre part, pour les codes produits construits à partir de codes en blocs linéaires quelconques, nous avons modifié l'algorithme proposé par Fossorier et Lin afin de l'intégrer dans un processus de décodage itératif efficace.

Au chapitre 3, nous avons introduit une généralisation des codes LDPC de Gallager, en remplaçant chaque code de parité simple par un code convolutif. Nous avons montré que ces codes, lorsqu'ils sont décodés selon le critère du maximum de vraisemblance, sont asymptotiquement bons, c'est-à-dire que leur distance minimale est une fonction linéaire de leur longueur. En pratique, étant donnée la structure des codes GLD convolutifs, il apparaît naturel de considérer un processus de décodage itératif, fondé sur le décodage successif des codes constituants. Les performances atteintes avec ces codes se situent à 0.8 dB de la limite de Shannon pour un rendement $R = \frac{1}{2}$. Nous avons encore pu améliorer ces performances en utilisant un entrelaceur quaternaire qui permet de réduire le nombre de cycles dans le graphe de Tanner du code GLD.

Afin d'évaluer et d'apprécier les performances asymptotiques des codes composites, nous avons proposé deux approches.

La première approche, présentée au chapitre 1, consiste à définir les performances optimales d'un code, sous l'hypothèse d'un décodage ML. La capacité d'un canal fixe les limites ultimes d'un code de longueur infinie. Par ailleurs, une borne inférieure de la probabilité d'erreur optimale pouvant être atteinte sur le canal gaussien a été développée par Shannon, lorsque la longueur et le rendement du code sont fixés. Nous avons comparé les performances des codes composites à ces différentes limites pour le canal AWGN. Nous avons également étudié l'évolution de la probabilité d'erreur optimale obtenue par Shannon en présence d'évanouissements de Rayleigh.

La seconde approche envisagée au chapitre 4 consiste à déterminer les performances asymptotiques des codes composites décodés itérativement. A cette fin, nous avons proposé deux méthodes : la méthode de l'évolution de la densité, qui repose sur l'étude de l'évolution des densités de probabilité des informations souples lors du décodage itératif, d'une part, et la méthode de la fonction de transfert, qui représente graphiquement l'évolution de la probabilité d'erreur au cours des itérations, d'autre part. Nous avons comparé les performances asymptotiques ainsi obtenues avec celles fournies par la capacité du canal. Nous avons, par ailleurs, montré que ces méthodes permettent de dégager des critères de conception pour les codes composites, et d'améliorer les schémas actuels en introduisant de l'irrégularité dans le graphe des codes.

De nombreux points restent encore à élucider quant aux performances des codes composites.

Nous avons vu dans ce document que le décodage itératif est équivalent au décodage ML lorsqu'il n'existe pas de cycle dans le graphe de Tanner du code. Malgré la présence de cycles dans leur graphe, les codes composites atteignent des performances très proches de l'optimal. Une meilleure compréhension de l'influence réelle des cycles au cours du décodage itératif serait nécessaire.

D'autre part, les méthodes que nous avons étudiées pour déterminer les performances asymptotiques d'un code composite décodé itérativement font l'hypothèse que la taille de l'entrelaceur est infinie. Les critères de conception que nous en avons dégagés s'appliquent ainsi à de longs codes. Il serait intéressant d'élaborer des critères d'optimisation des codes constituants pour des codes de petites longueurs.

Enfin, les codes LDPC et leurs différentes généralisations (codes de Tanner, codes GLD) présentent des propriétés et des performances très proches de celles du codage aléatoire. Malgré quelques avancées récentes [67], il reste à diminuer la complexité de leurs schémas de codage, notamment lorsque la longueur du code est grande.

Annexe A

Capacité du canal de Rayleigh

Cette démonstration s'appuie sur le raisonnement introduit par [42] pour démontrer le théorème de codage dans le cas d'un canal AWGN discret sans mémoire.

Soit $\mathcal{C}(N, K)$ un code en blocs de rendement R. Les $M = 2^K$ mots de code de $\mathcal{C}$ sont notés :

$$\mathbf{x_1}, ..., \mathbf{x_M} = (x_{1,1}, x_{1,2}, ..., x_{1,N}), ..., (x_{M,1}, \dots, x_{M,N}) \ .$$

Pour alléger certaines équations, nous adopterons indifféremment la notation :

$$\mathbf{x} = (x_1, x_2, ..., x_N)$$

Chaque mot de code est choisi avec une égale probabilité indépendamment des autres. L'étiquetage des mots de code se fait de telle sorte que la séquence correspondant à l'entier m à l'entrée du codeur est le mot de code $\mathbf{x_m}$.

La sortie du canal et l'évanouissement sont notés respectivement $\mathbf{y} = (y_1, \dots, y_N)$ et $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_N)$. Nous considérons les hypothèses suivantes :
- l'état du canal est toujours parfaitement connu du récepteur (estimation du canal parfaite),
- le canal est entrelacé, de telle sorte qu'il peut être considéré sans mémoire (canal DMC, *Discrete Memoryless Channel*),
- les évanouissements de Rayleigh sont non-sélectifs en fréquence, *i.e.*, il n'y a pas d'interférence entre symboles (IES),
- l'évanouissement est supposé constant sur un temps symbole.

Par conséquent, à l'instant i, la sortie du canal y_i ne dépend que du symbole en entrée du canal x_i, de l'évanouissement α_i et du bruit blanc additif gaussien z_i. Par ailleurs, du fait de la détection cohérente, nous restreignons notre étude à un évanouissement réel. La distribution de chaque évanouissement α_i est :

$$p(\alpha_i) = 2\alpha_i e^{-\alpha_i^2} \quad \textit{avec} \quad E[\alpha_i^2] = 1.$$

La fonction du décodeur est d'estimer la séquence émise par le codeur ainsi que l'étiquette correspondante en observant la sortie $\mathbf{y}$ du canal. Une erreur de décodage apparaît lorsque la séquence décodée m' diffère de la séquence émise m. Le décodeur est supposé optimiser le critère du maximum de vraisemblance, c'est-à-dire décoder la séquence reçue $\mathbf{y}$ en m pour laquelle :

$$\Pr(\mathbf{y}|\mathbf{x_m}) \geq \Pr(\mathbf{y}|\mathbf{x_{m'}})\ \forall m \neq m'.$$

La probabilité d'erreur par mot, Pe_W, après décodage est donnée par :

$$Pe_W = \Pr(m \neq m') = \frac{1}{M}\sum_{i=1}^{M} \Pr(m \neq m'|m = i),$$

puisque les messages $m,\ m = 1,\ldots,M$, sont équiprobables.

Soit Y_m l'ensemble des séquences $\mathbf{y}$ décodées en m (Y_m est la région de décision de $\mathbf{x_m}$ dans l'espace discret), et Y_m^c son complémentaire. Quand la source émet un message m, $\mathbf{x_m}$ est émis sur le canal et une erreur apparaît si $\mathbf{y}$ appartient à Y_m^c. La probabilité de faire une erreur conditionnellement au message m, notée Pe_m, et à l'évanouissement $\boldsymbol{\alpha}$ s'écrit donc :

$$\begin{aligned}
Pe_m &= \sum_{\mathbf{y}\in Y_m^c} P(\mathbf{y}|\mathbf{x_m}) = \sum_{\boldsymbol{\alpha}}\sum_{\mathbf{y}\in Y_m^c} P(\mathbf{y},\boldsymbol{\alpha}|\mathbf{x_m}) = \sum_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}) \sum_{\mathbf{y}\in Y_m^c} P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha}),\\
Pe_m &= \sum_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}) \sum_{\mathbf{y}} P(\mathbf{y}\in Y_m^c,\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha}) = \sum_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}) \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha})P(\mathbf{y}\in Y_m^c|\mathbf{y},\mathbf{x_m},\boldsymbol{\alpha}).
\end{aligned}$$

En moyennant sur les séquences $\mathbf{x_m}$, nous obtenons :

$$\bar{P}e_m = \sum_{\mathbf{x_m}} Q(\mathbf{x_m}) \sum_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}) \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha})P(\mathbf{y}\in Y_m^c|\mathbf{y},\mathbf{x_m},\boldsymbol{\alpha}).$$

où $Q(\mathbf{x_m})$ est la probabilité a priori du mot de code $\mathbf{x_m}$.

Nous avons par ailleurs le lemme suivant [24] :

Lemme 3 *Si $P(A_1)$,...,$P(A_M)$ sont les probabilités a priori sur un ensemble d'événements, et $P\left(\bigcup_m A_m\right)$ est la probabilité de leur union alors nous avons :*

$$\forall\, \rho \text{ tel que } 0 \leq \rho \leq 1,\quad P\left(\bigcup_m A_m\right) \leq \left[\sum_m P(A_m)\right]^{\rho}. \tag{A.1}$$

Soit $A_{m'}$ l'événement défini par {le mot de code m' est décodé sachant m émis } $\forall m' \neq m$. Sachant que $Y_m^c = \bigcup_{m'\neq m} A_{m'}$, nous pouvons écrire d'après le lemme précédent :

$$P(\mathbf{y}\in Y_m^c|\mathbf{x_m},\mathbf{y},\boldsymbol{\alpha}) = P\left(\bigcup_{m'\neq m} A_{m'}\right) \leq \left[\sum_{m'\neq m} P(A_{m'})\right]^{\rho} \quad \forall\, 0 \leq \rho \leq 1.$$

Lorsque $\mathbf{x_m}$, $\mathbf{x_{m'}}$ et $\boldsymbol{\alpha}$ sont fixés, l'événement $A_{m'}$ est équivalent à l'inégalité $P(\mathbf{y}|\mathbf{x_{m'}},\boldsymbol{\alpha}) \geq P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha})$. Nous pouvons maintenant majorer $P(A_{m'})$ par :

$$P(A_{m'}) = \sum_{\mathbf{x_{m'}},\ P(\mathbf{y}|\mathbf{x_{m'}},\boldsymbol{\alpha})\geq P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha})} Q(\mathbf{x_{m'}}) \quad\leq\quad \sum_{\mathbf{x_{m'}}} Q(\mathbf{x_{m'}}) \left(\frac{P(\mathbf{y}|\mathbf{x_{m'}},\boldsymbol{\alpha})}{P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha})}\right)^{s}, \quad \forall\ s > 0.$$

Dans cette inégalité, $\mathbf{x_{m'}}$ est une variable muette et peut être remplacée par $\mathbf{x}$; la borne devient ainsi indépendante de m'. Nous avons alors $M-1$ choix pour $m' \neq m$:

$$P(\mathbf{y} \in Y_c^m | \mathbf{x_m}, \mathbf{y}, \boldsymbol{\alpha}) \leq \left[(M-1) \sum_{\mathbf{x}} Q(\mathbf{x}) \frac{P(\mathbf{y}|\mathbf{x},\boldsymbol{\alpha})^s}{P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha})^s} \right]^\rho, \tag{A.2}$$

d'où

$$\bar{P}\mathrm{e}_m \leq (M-1)^\rho \sum_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}) \sum_{\mathbf{y}} \left[\sum_{\mathbf{x_m}} Q(\mathbf{x_m}) P(\mathbf{y}|\mathbf{x_m},\boldsymbol{\alpha})^{1-s\rho} \right] \left[\sum_{\mathbf{x}} Q(\mathbf{x}) P(\mathbf{y}|\mathbf{x},\boldsymbol{\alpha})^s \right]^\rho. \tag{A.3}$$

En posant $s = \frac{1}{1+\rho}$, nous obtenons pour $0 < \rho \leq 1$:

$$\bar{P}\mathrm{e}_m \leq (M-1)^\rho \sum_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}) \sum_{\mathbf{y}} \left[\sum_{\mathbf{x}} Q(\mathbf{x}) P(\mathbf{y}|\mathbf{x},\boldsymbol{\alpha})^{\frac{1}{1+\rho}} \right]^{1+\rho}. \tag{A.4}$$

Pour un canal DMC, nous pouvons par ailleurs écrire :

$$P(\mathbf{y}|\mathbf{x},\boldsymbol{\alpha}) = \prod_n P(y_n|x_n,\alpha_n).$$

En se restreignant au cas où les symboles de $\mathbf{x}$ sont choisis indépendamment les uns des autres :

$$Q(\mathbf{x}) = \prod_{n=1}^N Q(x_n).$$

De plus, les évanouissements sont indépendants ; par conséquent, nous avons également :

$$P(\boldsymbol{\alpha}) = \prod_{n=1}^N P(\alpha_n).$$

Nous avons donc :

$$\begin{aligned}
\bar{P}\mathrm{e}_m &\leq (M-1)^\rho \sum_{\alpha_1} \cdots \sum_{\alpha_N} \prod_{n=1}^N P(\alpha_n) \sum_{y_1} \cdots \sum_{y_N} \left[\sum_{x_1} \cdots \sum_{x_N} \prod_{n=1}^N Q(x_n) P(y_n|x_n,\alpha_n)^{\frac{1}{1+\rho}} \right]^{1+\rho}, \\
&\leq (M-1)^\rho \sum_{\alpha_1} \cdots \sum_{\alpha_N} \prod_{n=1}^N P(\alpha_n) \sum_{y_n} \left[\sum_{x_n} Q(x_n) P(y_n|x_n,\alpha_n)^{\frac{1}{1+\rho}} \right]^{1+\rho}, \\
&\leq (M-1)^\rho \prod_{n=1}^N \sum_{\alpha_n} P(\alpha_n) \sum_{y_n} \left[\sum_{x_n} Q(x_n) P(y_n|x_n,\alpha_n)^{\frac{1}{1+\rho}} \right]^{1+\rho}, \\
&\leq (M-1)^\rho \prod_{n=1}^N \sum_{\alpha_n} \sum_{y_n} P(\alpha_n) \left[\sum_{x_n} Q(x_n) P(y_n|x_n,\alpha_n)^{\frac{1}{1+\rho}} \right]^{1+\rho}, \\
&\leq (M-1)^\rho \prod_{n=1}^N \sum_{\alpha_n} \sum_{y_n} \left[\sum_{x_n} P(\alpha_n)^{\frac{1}{1+\rho}} Q(x_n) P(y_n|x_n,\alpha_n)^{\frac{1}{1+\rho}} \right]^{1+\rho}, \\
&\leq (M-1)^\rho \prod_{n=1}^N \sum_{\alpha_n} \sum_{y_n} \left[\sum_{x_n} Q(x_n) P(y_n,\alpha_n|x_n)^{\frac{1}{1+\rho}} \right]^{1+\rho}.
\end{aligned}$$

En majorant $(M-1)$ par M et en remarquant que le terme à l'intérieur du produit est indépendant de l'indice n, nous obtenons :

$$\bar{P}\mathrm{e}_m \leq M^\rho \left(\sum_\alpha \sum_y \left[\sum_x Q(x) P(y, \alpha|x)^{\frac{1}{1+\rho}} \right]^{1+\rho} \right)^N .$$

Posons

$$E_0(\rho, Q) = -\log_2 \left(\sum_\alpha \sum_y \left[\sum_x Q(x) P(y, \alpha|x)^{\frac{1}{1+\rho}} \right]^{1+\rho} \right).$$

La probabilité d'erreur $\bar{P}\mathrm{e}_m$ s'écrit alors :

$$\bar{P}\mathrm{e}_m \leq 2^{(-N(E_0(\rho,Q)-\rho R))} \tag{A.5}$$

Comme pour le canal gaussien, nous définissons $E(R)$ l'exposant de fiabilité par :

$$E(R) = \max_{0\leq\rho\leq 1} \max_Q \left[E_0(\rho, Q) - \rho R\right]. \tag{A.6}$$

Nous cherchons maintenant les valeurs de R pour lesquelles l'exposant $E(R)$ est positif.

Le coefficient $(E_0(\rho, Q) - \rho R)$ s'annule en $\rho = 0$; par conséquent, si la tangente à $(E_0(\rho, Q) - \rho R)$ en $\rho = 0$ est positive, alors $\max_{0\leq\rho\leq 1}(E_0(\rho, Q) - \rho R)$ sera nécessairement positif.

Évaluons la dérivée partielle de $(E_0(\rho, Q) - \rho R)$ par rapport à ρ, en $\rho = 0$. Après calculs nous obtenons :

$$\left.\frac{\partial E_0(\rho, Q)}{\partial \rho}\right|_{\rho=0} = \sum_\alpha \sum_x \sum_y Q(x) P(y, \alpha|x) \log_2 \frac{P(y, \alpha|x)}{P(y, \alpha)}.$$

Par définition, l'information mutuelle moyenne entre la variable aléatoire X et le couple de variables aléatoires (Y, A) qui caractérisent respectivement l'entrée, la sortie et l'évanouissement du canal à chaque instant et dont les réalisations sont x, y et α, est :

$$I(X; Y, A) = \sum_\alpha \sum_x \sum_y P(y, x, \alpha) \log_2 \frac{P(y, \alpha|x)}{P(y, \alpha)}.$$

Par conséquent :

$$\left.\frac{\partial E_0(\rho, Q)}{\partial \rho}\right|_{\rho=0} = I(X; Y, A).$$

Par un raisonnement analogue à celui utilisé pour le canal gaussien, nous déduisons que $E(R)$ est positif si :

$$R < \max_{Q(x)} I(X; Y, A)$$

L'évanouissement α étant parfaitement connu du récepteur, $\max_{Q(x)} I(X;Y,A)$ désigne la quantité maximale d'information que l'on peut obtenir en connaissant Y et A. Il s'agit de la capacité C du canal de Rayleigh.

Nous pouvons ainsi écrire le théorème de codage suivant :

Théorème A.0.1 *(Théorème de codage aléatoire pour le canal de Rayleigh idéal)*
Considérons un canal de Rayleigh idéal caractérisé par les hypothèses énoncées au début de cette annexe.
Pour un taux d'émission $0 < R < C$, où C est la capacité du canal en bits/dimension, il existe un code de longueur N tel que sa probabilité d'erreur soit arbitrairement petite lorsque N tend vers l'infini. Cette probabilité d'erreur peut être majorée par une fonction de la forme :

$$Pe \leq 2^{-NE(R)} \ ,$$

où $E(R)$ est donné par :

$$E(R) = E_0(\rho, Q) - \rho R = -\log_2 \left(\sum_\alpha \sum_y \left[\sum_x Q(x) P(y, \alpha | x)^{\frac{1}{1+\rho}} \right]^{1+\rho} \right) - \rho R. \qquad \text{(A.7)}$$

$E(R)$ est positif pour $R < C$.
Nous pouvons de même démontrer la réciproque du théorème de codage pour le canal de Rayleigh.

D'après ce qui précède, la capacité du canal de Rayleigh s'écrit :

$$C = \max_{Q(x)} I(X;Y,A).$$

Or,

$$\begin{aligned} I(X;Y,A) &= I(X;Y) - I(X;Y|A) \\ &= I(X;Y|A) \end{aligned}$$

car X et A sont des variables aléatoires indépendantes. Par conséquent, nous obtenons :

$$I(X;Y,A) = \sum_\alpha I(X;Y^\alpha) P(\alpha)$$

où Y^α désigne la sortie du canal pour une valeur de l'évanouissement α fixée.

Lorsque l'évanouissement α n'est connu que du récepteur, la puissance en émission reste constante quel que soit α (aucun contrôle de puissance n'est possible). Le canal de Rayleigh à chaque instant est alors équivalent à un canal AWGN dont le bruit est de variance $\frac{N_0}{\alpha^2}$. Ainsi, la distribution $Q(x)$ maximisant l'information mutuelle pour le canal gaussien maximise également l'information mutuelle pour le canal de Rayleigh, lorsque α est fixé. Cette distribution est, par ailleurs, la même pour chaque évanouissement, puisqu'elle ne dépend que de la puissance en émission. Par conséquent,

$$\begin{aligned} C &= \max_{Q(x)} \sum_\alpha I(X;Y^\alpha) P(\alpha) \\ &= \sum_\alpha \left(\max_{Q(x)} I(X;Y^\alpha) \right) P(\alpha) \end{aligned}$$

où $C_{AWGN}(\alpha) = \max_{Q(x)} I(X;Y^\alpha)$ est la capacité d'un canal AWGN de variance $\frac{N_0}{\alpha^2}$.

Annexe B

Treillis des codes en blocs

Nous avons présenté au chapitre 2 la construction du treillis d'un code en blocs linéaire proposée par Bahl *et al.* [2]. D'autres constructions sont possibles menant également au treillis minimal d'un code donné. Nous détaillons ici trois de ces constructions en nous inspirant de [81].

Soit C un code en blocs linéaire de longueur n et de dimension k. Son treillis minimal est noté $T = (V, E, A)$ où :

- V est l'ensemble des sommets de T : $V = V_0 \cup V_1 \cup \ldots \cup V_n$, où V_t est l'ensemble des sommets (ou états) à l'instant t,
- E est l'ensemble des branches du treillis : $E = E_0 \cup E_1 \cup \ldots \cup E_n$,
- A est l'alphabet des étiquettes sur les branches : $A = A_0 \cup A_1 \cup \ldots \cup A_n$ avec $A_t = GF(q)$, $\forall\, t$.

B.1 Construction de Massey[59]

Afin d'expliquer la construction introduite par Massey, plusieurs définitions sont au préalable nécessaires.

Définition B.1.1 *Soit un vecteur non nul* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *sur* $GF(q)$ *;* $L(\mathbf{x})$ *est le plus petit entier* i *tel que* $x_i \neq 0$. $L(\mathbf{x})$ *est appelé l'indice inférieur de* $\mathbf{x}$.

Définition B.1.2 *Soit* $M = (x_{ij})$ *une matrice de dimension* $k \times n$ *sur* $GF(q)$.
M est une matrice échelon si ses lignes $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ *sont telles que* $L(\mathbf{x_1}) \leq L(\mathbf{x_2}) \leq \ldots \leq L(\mathbf{x_k})$ *et si les* k *colonnes trouvées aux positions* $L(\mathbf{x_1}),\ldots,L(\mathbf{x_k})$ *dans* M *sont de poids* 1. *Alors si* $j = L(\mathbf{x_i})$, x_{ij} *est la seule valeur non nulle dans la* j*-ième colonne de* M.

Soit G la matrice génératrice d'un code linéaire C. G est supposée être une matrice échelon, et les indices inférieurs sont appelés $\gamma_1, \gamma_2, \ldots, \gamma_k$. Les k positions correspondant à ces k indices forment un ensemble d'information de C ; par conséquent, si $(c_1, c_2, \ldots, c_n) = (u_1, u_2, \ldots, u_k)G$,

$(c_{\gamma_1}, \ldots, c_{\gamma_k}) = (u_1, \ldots, u_k)$ peut être appelé vecteur des symboles d'information tandis que les $n-k$ autres symboles sont les symboles de parité.

Le treillis de Massey est construit en identifiant les sommets de V_i aux symboles de parité non observés à l'instant i et évalués à partir des symboles d'information déjà observés à cet instant, en supposant que tous les autres symboles d'information sont nuls. Si m est le plus grand entier tel que $\gamma_m \leq i$, alors :

$$V_i \stackrel{\triangle}{=} \{(c_{i+1}, c_{i+2}, \ldots, c_n) \text{ tel que } (c_1, c_2, \ldots, c_n) = (u_1, u_2, \ldots, u_m, 0, 0, \ldots, 0)G\} \qquad \text{(B.1)}$$

et

$$V_0 = \{\mathbf{0}\} \qquad V_n = \{\epsilon\},$$

où ϵ représente la chaîne nulle. Les branches de T sont définies par :

- si $i > \gamma_m$: une branche $e \in E_i$ entre $v \in V_{i-1}$ et $v' \in V_i$ existe si et seulement si un mot de code $(c_1, c_2, \ldots, c_n) \in C$ existe tel que :

$$(c_i, c_{i+1}, \ldots, c_n) = v,$$

et

$$(c_{i+1}, c_{i+2}, \ldots, c_n) = v'.$$

L'étiquette de e est alors c_i. Dans ce cas, seulement une branche part de chaque sommet v de V_{i-1}.
- si $i = \gamma_m$: une branche $e \in E_i$ entre $v \in V_{i-1}$ et $v' \in V_i$ existe si et seulement si deux mots de code de C, $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ et $\mathbf{c}' = (c'_1, c'_2, \ldots, c'_n)$, existent tels que :

$$(c_i, c_{i+1}, \ldots, c_n) = v,$$

$$(c'_{i+1}, c'_{i+2}, \ldots, c'_n) = v'.$$

Alors, soit $\mathbf{c} = \mathbf{c}'$, soit $\beta(\mathbf{c}' - \mathbf{c})$ est égal à la m-ième ligne de G pour tout β non nul de $GF(q)$. L'étiquette de e est c'_i et dans ce cas chaque sommet $v \in V_{i-1}$ a un degré égal à q, i.e., q branches partent de chaque sommet v.

Exemple 3

Considérons le même code que celui proposé au chapitre 2. La matrice G est choisie sous la forme d'une matrice échelon (des opérations linéaires sur les lignes de la matrice G permettent de l'écrire sous cette forme) :

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

Les indices inférieurs sont $\gamma_1 = 1$, $\gamma_2 = 2$ et $\gamma_3 = 5$. Ces indices donnent les instants auxquels deux branches partent de chaque état (ces deux branches correspondent aux deux symboles d'information possibles 0 et 1). Les sommets sont donnés par : $V_0 = \{0\} \qquad V_7 = \{\epsilon\}$,

$$V_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \textit{1 symbole d'information observé } (\gamma_1).$$

$$V_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad V_3 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} V_4 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$\Rightarrow$ *2 symboles d'information observés (γ_1 et γ_2).*

$$V_5 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad V_6 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$\Rightarrow$ *3 symboles d'information observés (γ_1, γ_2 et γ_3).*

Le treillis correspondant est représenté sur la figure B.1. Les sommets manquant sont obtenus par combinaison linéaire des sommets donnés par V_i à chaque instant i. Ce treillis est isomorphe au treillis BCJR.

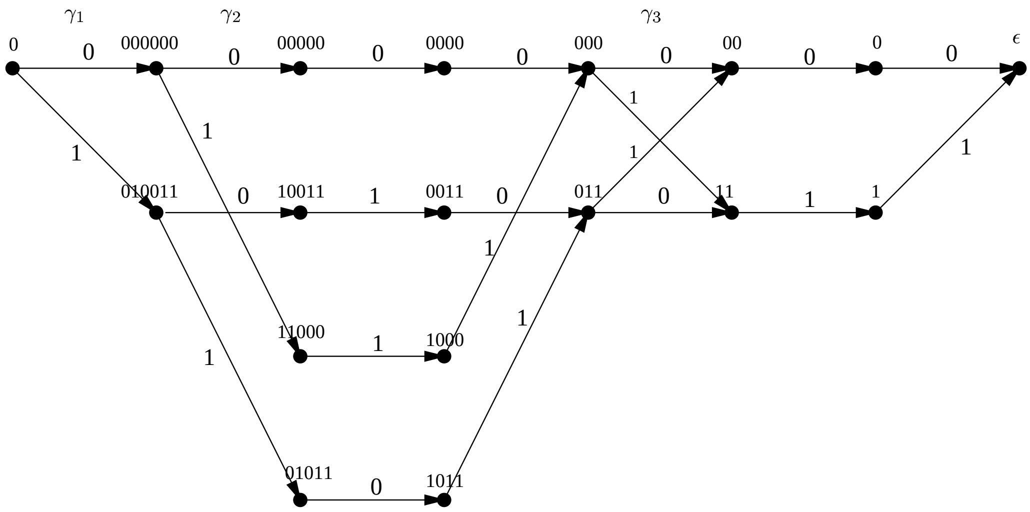


FIG. B.1 – Construction de Massey.

B.2 Construction de Forney [36]

Soient P_i et F_i deux versions raccourcies du code C à l'instant i :

- $P_i = \{(c_1, c_2, \ldots, c_i) : (c_1, c_2, \ldots, c_i, c_{i+1}, \ldots, c_n) \in C \quad \text{pour } c_{i+1} = \ldots = c_n = 0\}$: sous-code passé de C à l'instant i,
- $F_i = \{(c_{i+1}, c_{i+2}, \ldots, c_n) : (c_1, c_2, \ldots, c_i, c_{i+1}, \ldots, c_n) \in C \quad \text{pour } c_1 = \ldots = c_i = 0\}$: sous-code futur de C à l'instant i,

avec $P_n = F_0 = C$, $P_0 = F_n = \{0\}$ et $P_i \oplus F_i$ un sous-code linéaire de C. L'ensemble des sommets V_i du treillis T à l'instant i est défini par :

$$V_i \triangleq C/(P_i \oplus F_i) \qquad \forall i = 0, \ldots, n, \tag{B.2}$$

et $V_0 = V_n = C$.
Une branche $e \in E_i$ existe entre $v \in V_{i-1}$ et $v' \in V_i$ si et seulement si un mot de code $\mathbf{c} = (c_1, c_2, \ldots, c_n) \in C$ existe tel que $\mathbf{c} \in v \cap v'$. L'étiquette de cette branche e est c_i.

Exemple 4
On considère toujours le même code C ; sa matrice génératrice est donnée par :

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

Les sous-codes passés et futurs de C sont définis à chaque instant i par :
$P_0 = P_1 = P_2 = P_3 = \{\mathbf{0}\}$,
$P_4 = \{\mathbf{0}, 0111\}$,
$P_5 = \{\mathbf{0}, 01110, 10101, 11011\}$,
$P_6 = \{\mathbf{0}, 011100, 101010, 110110\}$,
$P_7 = C$.

$F_0 = C$,
$F_1 = \{\mathbf{0}, 111000, 000111, 111111\}$,
$F_2 = \{\mathbf{0}, 00111\}$,
$F_3 = \{\mathbf{0}, 0111\}$,
$F_4 = \{\mathbf{0}, 111\}$,
$F_5 = F_6 = F_7 = \{\mathbf{0}\}$.

Par conséquent, les sommes directes $P_i \oplus F_i$ pour $i = 0, \ldots, n$, sont :
$P_0 \oplus F_0 = C$,
$P_1 \oplus F_1 = \{\mathbf{0}, 0111000, 0000111, 0111111\}$,
$P_2 \oplus F_2 = \{\mathbf{0}, 0000111\}$,
$P_3 \oplus F_3 = \{\mathbf{0}, 0000111\}$,
$P_4 \oplus F_4 = \{\mathbf{0}, 0111000, 0000111, 0111111\}$,
$P_5 \oplus F_5 = \{\mathbf{0}, 0111000, 1010100, 1101100\}$,
$P_6 \oplus F_6 = \{\mathbf{0}, 0111000, 1010100, 1101100\}$,
$P_7 \oplus F_7 = C$.

Ceci donne la structure du treillis à chaque instant. L'ensemble des sommets peut en effet être déduit sachant que V_i représente les sous-ensembles modulo $P_i \oplus F_i$. Notons tout d'abord que :

$$V_1 = V_4, \quad V_5 = V_6, \quad V_2 = V_3.$$

De plus,

$$V_1 = V_4 = \{(\mathbf{0}, 0111000, 0000111, 0111111), (1010100, 1101100, 1010011, 1101011)\},$$

$$V_2 = V_3 = \{(\mathbf{0}, 0000111), (0111000, 0111111), (1010100, 1010011), (1101011, 1101100)\},$$

$$V_5 = V_6 = \{(\mathbf{0}, 0111000, 1010100, 1101100), (0000111, 0111111, 1010011, 1101011)\}.$$

Les sommets sont caractérisés sur le treillis par un représentant de chaque coset ; par convention, le premier vecteur de chaque coset est choisi comme représentant. Le treillis résultant est dessiné sur la figure B.2. Ce treillis est isomorphe aux treillis de Massey et au treillis BCJR bien

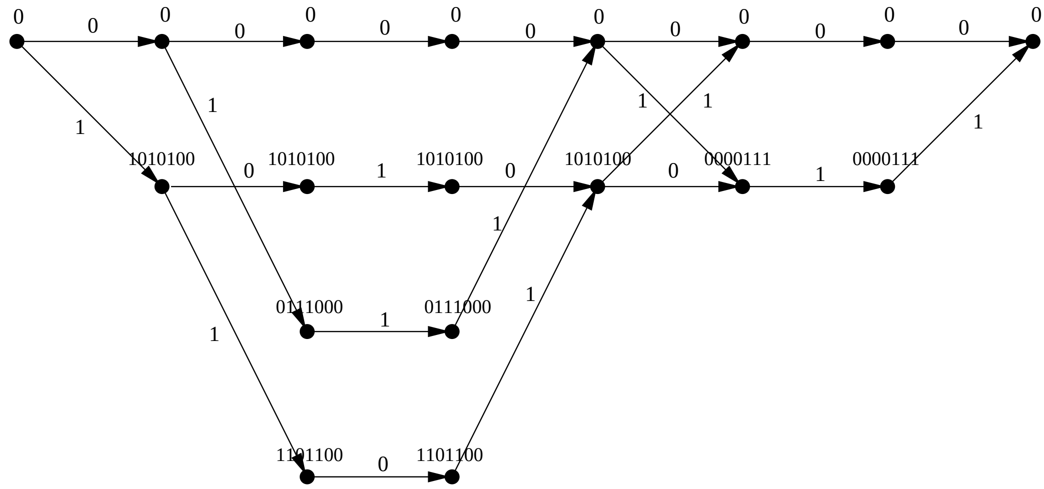


Fig. B.2 – Construction de Forney.

que les sommets et les branches n'aient pas la même signification.

B.3 Construction de Kschischang et Sorokine [51]

La construction de Kschischang et Sorokine permet de spécifier une catégorie de treillis non isomorphes pour le code C ; seul l'un de ces treillis est minimal. Le code C peut être représenté comme la somme de sous-codes élémentaires, et le treillis de C est alors construit à partir du produit des treillis minimaux de chacun des sous-codes élémentaires.

B.3.1 Produit de treillis

Définition B.3.1 *(Produit de deux treillis)*
Soient $T' = (V', E', A')$ et $T^\star(V^\star, E^\star, A^\star)$ deux treillis de longueur n. Le produit des treillis T' et $T^\star$, noté $T = T' \times T^\star$, a la propriété suivante :

$$C = C(T') + C(T^\star) \stackrel{\triangle}{=} \{\mathbf{c}' + \mathbf{c}^\star, \mathbf{c}' \in C(T') \text{ et } \mathbf{c}^\star \in C(T^\star)\}. \tag{B.3}$$

où $C(T')$ et $C(T^\star)$ sont deux sous-codes de C de treillis respectifs T' et $T^\star$.

L'ensemble des sommets de $T = T' \times T^\star$, à l'instant i, est le produit cartésien :

$$V_i = V'_i \times V^\star_i \triangleq \{(v', v^\star) : v' \in V' \text{ et } v^\star \in V^\star\}. \quad \text{(B.4)}$$

Une branche $e \in E_i$ existe entre $(v'_1, v^\star_1) \in V_{i-1}$ et $(v'_2, v^\star_2) \in V_i$ si et seulement si (v'_1, v'_2, a') est une branche de E'_i pour $a' \in A$ et $(v^\star_1, v^\star_2, a^\star)$ est une branche de $E^\star_i$ pour $a^\star \in A$. L'étiquette de cette branche est $a(e) = a' + a^\star$. L'ensemble des étiquettes de T à l'instant i est donné par le produit cartésien :

$$E_i \triangleq \{((v'_1, v^\star_1), (v'_2, v^\star_2), a' + a^\star) : (v'_1, v'_2, a') \in E'_i \text{ et } (v^\star_1, v^\star_2, a^\star) \in E^\star_i\}. \quad \text{(B.5)}$$

Notons que T n'est pas toujours le treillis minimal de C, même si T' et $T^\star$ sont deux treillis minimaux de C' et $C^\star$. Ceci est illustré sur la figure B.3. Le nombre de sommets (respectivement le nombre de branches) de T à l'instant i est le produit du nombre de sommets (respectivement du nombre de branches) de T' par le nombre de sommets (respectivement le nombre de branches) de $T^\star$.

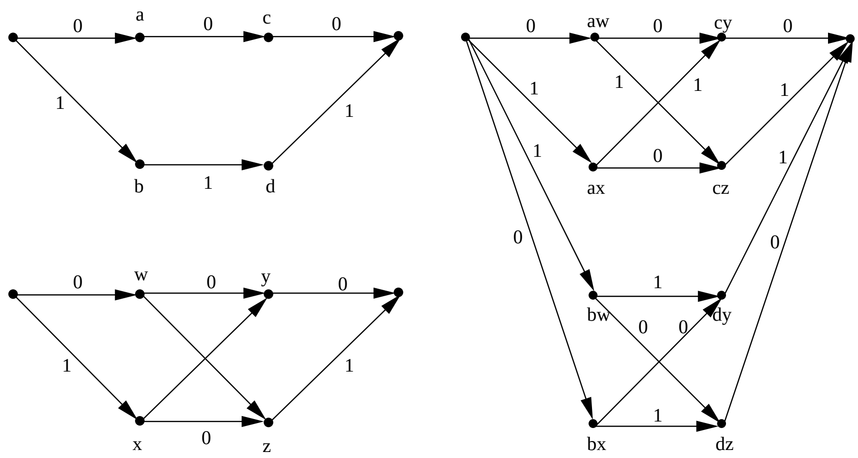


Fig. B.3 – Deux treillis et leur produit.

Soit G la matrice génératrice de C. Les lignes de G sont notées $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ et appelées générateurs. Elles forment une base de C ; chaque ligne génère un sous-code de C noté $< \mathbf{x}_i >$ de dimension 1. Alors :

$$C = < \mathbf{x_1} > + < \mathbf{x_2} > + \ldots + < \mathbf{x_k} > .$$

Si $T_1, T_2, \ldots, T_k$ sont les treillis respectifs de $< \mathbf{x_1} >, < \mathbf{x_2} >, \ldots, < \mathbf{x_k} >$, alors leur produit donne un treillis de C. Soit $T_{\mathbf{x_i}}$ le treillis minimal de $< \mathbf{x_i} >$. On définit :

$$T_G \triangleq T_{\mathbf{x_1}} \times T_{\mathbf{x_2}} \times \ldots \times T_{\mathbf{x_k}},$$

comme le *treillis de Kschischang-Sorokine de C fondé sur la matrice génératrice* G. Le treillis T_G n'est pas toujours minimal mais un choix judicieux de $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ peut le rendre ainsi.

Considérons maintenant la structure du treillis $T_\mathbf{x}$ de $< \mathbf{x} >$.
Soit $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ un mot de code non nul de C. Par analogie avec l'indice inférieur de $\mathbf{c}$, l'indice supérieur de $\mathbf{c}$ est défini comme l'entier i le plus grand tel que $c_i \neq 0$, et est noté $R(\mathbf{c})$. Nous avons alors la définition suivante :

Définition B.3.2 *(Span d'un mot de code)*
Le span d'un mot de code non nul de C, noté $[\mathbf{c}]$, est l'intervalle non vide :

$$[\mathbf{c}] \stackrel{\triangle}{=} [L(\mathbf{c}), R(\mathbf{c})] \subseteq \{1, 2, \ldots, n\}.$$

$\mathbf{c}$ *commence en $L(\mathbf{c})$, finit en $R(\mathbf{c})$ et est actif sur l'intervalle $[L(\mathbf{c}), R(\mathbf{c}) - 1]$.*

Nous pouvons remarquer que :
- Le span d'un vecteur nul est l'intervalle vide par convention.
- Si $L(\mathbf{c}) = R(\mathbf{c})$ alors le poids de $\mathbf{c}$ égal à 1 et $\mathbf{c}$ n'est jamais actif.

Nous appelons *durée d'activité* de $\mathbf{c}$, notée $s(\mathbf{c})$, les différents instants pendant lesquels $\mathbf{c}$ est actif. Elle est donnée par $s(\mathbf{c}) = R(\mathbf{c}) - L(\mathbf{c})$.

La figure B.4 représente le treillis minimal $T_\mathbf{x}$ d'un code binaire $<\mathbf{x}>$ de span $[a, b]$.

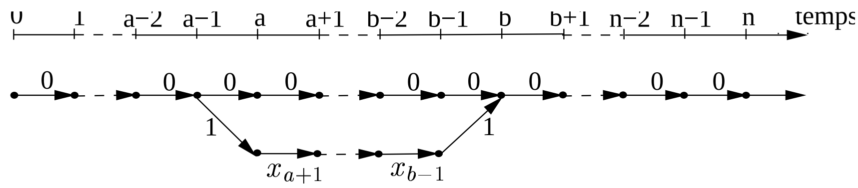


FIG. B.4 – Treillis minimal de $< \mathbf{x} >$.

Cette structure permet de compter aisément le nombre de sommets du treillis de Kschischang-Sorokine. En fait, le nombre de sommets à l'instant i dans le treillis $T_G = (V, E, GF(q))$ est q^{s_i}, où s_i est le nombre de générateurs actifs de G à l'instant i. Si l'on définit le span total par :

$$\sigma \stackrel{\triangle}{=} s(\mathbf{x_1}) + s(\mathbf{x_2}) + \ldots + s(\mathbf{x_k}),$$

alors, pour minimiser le nombre de sommets du treillis il faut considérer une base $(\mathbf{x_1}, \ldots, \mathbf{x_k})$ ayant un span total le plus petit possible.

Définition B.3.3 *(Matrice génératrice à span minimal)*
Une matrice génératrice G d'un code linéaire en blocs $C(n, k)$ a un span minimal si le span total de G est le plus petit possible, c'est-à-dire :

$$s(\mathbf{x_1}) + s(\mathbf{x_2}) + \ldots + s(\mathbf{x_k}) = \min_{\mathbf{x'_1}, \mathbf{x'_2}, \ldots, \mathbf{x'_k}} s(\mathbf{x'_1}) + s(\mathbf{x'_2}) + \ldots + s(\mathbf{x'_k}),$$

où $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ est l'ensemble des lignes de G. Le minimum est pris sur les $(q^n - 1)(q^n - q)\ldots(q^n - q^{k-1})$ bases $\mathbf{x'_1}, \mathbf{x'_2}, \ldots, \mathbf{x'_k}$ de C.

De manière intuitive, si les durées d'activité de $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ sont faibles, chaque ligne de la matrice génératrice sera active sur un petit intervalle, et contribuera faiblement aux nombre de sommets du treillis.

Théorème B.3.1 *Le treillis proposé par Kschischang-Sorokine, T_G, fondé sur la matrice génératrice G est le treillis minimal de C si et seulement si G a un span minimal.*

B.3.2 Comment générer une matrice G à span minimal pour un code linéaire ?

Théorème B.3.2 *Une matrice génératrice est à span minimal si et seulement si elle ne contient aucune ligne commençant ou finissant dans la même position.*

Par conséquent, la conversion de n'importe quel ensemble de générateurs $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ en un ensemble à span minimal peut être fait grâce à l'algorithme suivant :

Tant qu'il existe $\mathbf{x_i}, \mathbf{x_j}$ tels que $L(\mathbf{x_i}) = L(\mathbf{x_j})$ ou $R(\mathbf{x_i}) = R(\mathbf{x_j})$ faire :

$$
\begin{array}{l}
\{ \\
\quad \text{si} \;\; [\mathbf{x_j}] \subseteq [\mathbf{x_i}] \;\; \text{alors } \mathbf{x_i} := \mathbf{x_i} + \mathbf{x_j} \\
\quad \phantom{\text{si} \;\; [\mathbf{x_j}] \subseteq [\mathbf{x_i}] \;\;} \text{sinon } \mathbf{x_j} := \mathbf{x_i} + \mathbf{x_j} \\
\}
\end{array}
$$

Cet algorithme se termine dans les premières kn itérations car le span total diminue à chaque étape. En fait, Kschischang et Sorokine ont montré que l'algorithme est en $o(k^2)$ itérations.

Exemple 5
Pour illustrer cette construction, nous considérons toujours le même exemple. Soit G_1 la matrice échelon suivante :

$$G_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Le treillis correspondant T_{G_1} est donné sur la figure B.5. Cependant, G_1 n'est pas à span minimal

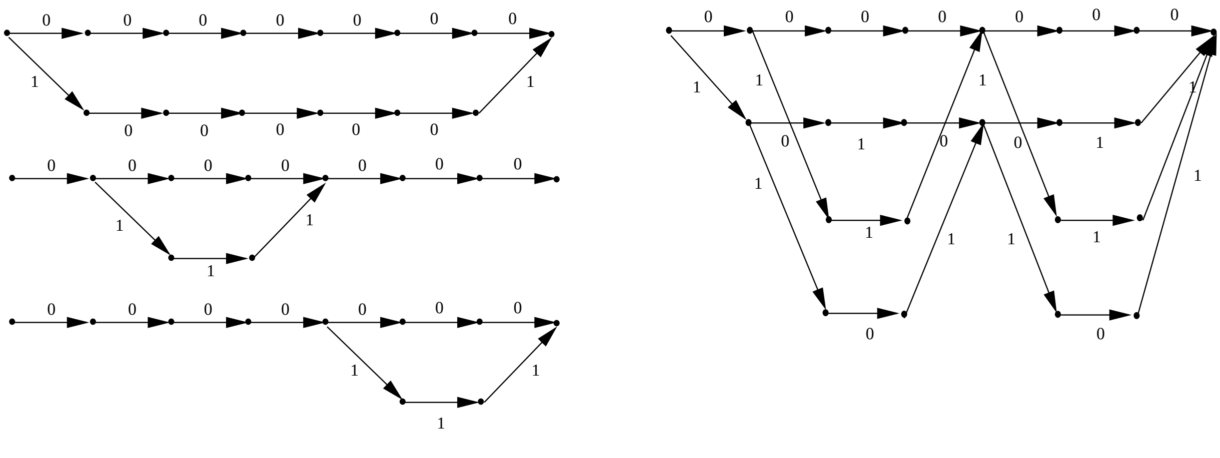


FIG. B.5 – Treillis T_{G1}.

car $\mathbf{x_1}$ et $\mathbf{x_3}$ se terminent au même instant $i = 7$. Comme $[\mathbf{x_3}] \subseteq [\mathbf{x_1}]$, $\mathbf{x_1}$ est remplacée par $\mathbf{x_1} + \mathbf{x_3}$. Ceci donne la matrice suivante :

$$G_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Les spans de $\mathbf{x_1}$*,* $\mathbf{x_2}$ *et* $\mathbf{x_3}$ *sont respectivement* $[1,5]$*,* $[2,4]$ *et* $[5,7]$*. Le treillis correspondant* T_{G_2} *est illustré sur la figure B.6. Ce treillis est isomorphe aux treillis obtenus par les autres*

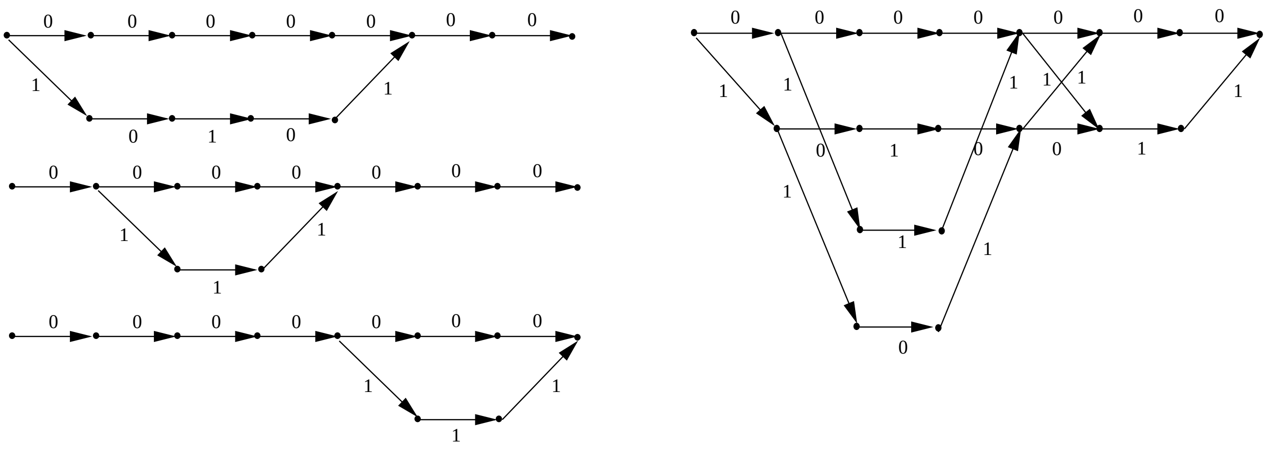


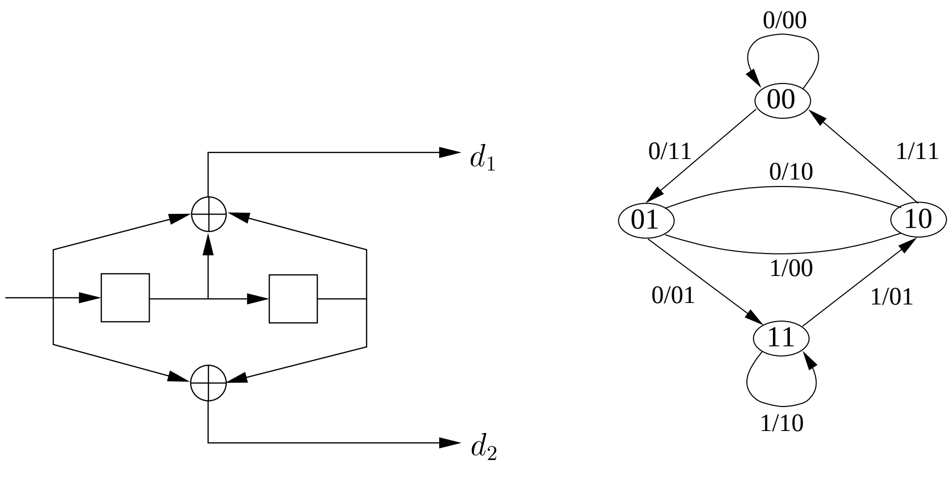
FIG. B.6 – Treillis T_{G2}.

constructions.

Annexe C

Calcul des polynômes énumérateurs de poids conditionnels d'un code convolutif

Nous proposons dans cette annexe de calculer les polynômes énumérateurs de poids d'un code convolutif en nous inspirant de [8] et [27]. Pour simplifier notre présentation, nous nous plaçons dans le cas d'un code convolutif C non récursif non systématique de rendement $r = \frac{1}{2}$ et de polynômes générateurs $(7, 5)$. Une représentation du codeur de ce code est donnée sur la figure C.1(a). Nous nous intéressons à une fenêtre du treillis de longueur $W = 2w + 1$ branches. Nous ne spécifions ni l'état initial ni l'état final du codeur, la fenêtre étant choisie au milieu d'un treillis de taille infinie.



(a) *Codeur convolutif* (b) *Diagramme d'états*

FIG. C.1 – Code convolutif non récursif non systématique de rendement $r = \frac{1}{2}$ et de polynômes générateurs $(7, 5)$

Pour construire la matrice de transfert T du code C, nous considérons son diagramme d'états.

Celui-ci décrit les transitions T_{jk} entre les états du code, chaque transition étant étiquetée par le bit d'entrée i et les deux bits de sortie d_1 d_2 (notée i/d_1d_2). Le diagramme d'états du code C est représenté sur la figure C.1(b). Lorsqu'une transition existe entre l'état j et l'état k dans le treillis, nous remplissons la case T_{jk} dans la matrice de transfert T en indiquant le poids de l'entrée et celui de la sortie sous la forme $LI^iD_1^{d_1}D_2^{d_2}$. Pour le code C par exemple, nous obtenons la matrice suivante :

$$T = \begin{bmatrix} L & 0 & LID_1D_2 & 0 \\ LD_1D_2 & 0 & LI & 0 \\ 0 & LD_1 & 0 & LID_2 \\ 0 & LD_2 & 0 & LID_1 \end{bmatrix} .$$

Pour évaluer les polynômes énumérateurs de poids conditionnels d'un code convolutif par rapport à l'un des bits de la fenêtre du treillis observée, il faut fixer la valeur de ce bit. Selon si ce bit est en position d_1 ou d_2, et est égal à 0 ou à 1, nous obtenons 4 nouvelles matrices de transition, construites en fixant la valeur du bit et en supprimant les transitions où il n'a pas cette valeur :

$$T_{d_1=0} = \begin{bmatrix} L & 0 & 0 & 0 \\ 0 & 0 & LI & 0 \\ 0 & 0 & 0 & LID_2 \\ 0 & LD_2 & 0 & 0 \end{bmatrix}, T_{d_1=1} = \begin{bmatrix} 0 & 0 & LID_1D_2 & 0 \\ LD_1D_2 & 0 & 0 & 0 \\ 0 & LD_1 & 0 & 0 \\ 0 & 0 & 0 & LID_1 \end{bmatrix},$$

$$T_{d_2=0} = \begin{bmatrix} L & 0 & 0 & 0 \\ 0 & 0 & LI & 0 \\ 0 & LD_1 & 0 & 0 \\ 0 & 0 & 0 & LID_1 \end{bmatrix}, T_{d_2=1} = \begin{bmatrix} 0 & 0 & LID_1D_2 & 0 \\ LD_1D_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & LID_2 \\ 0 & LD_2 & 0 & 0 \end{bmatrix}.$$

Nous nous intéressons à une fenêtre de W branches du code. Le coefficient de la ligne j et de la colonne k du code de la matrice T^W obtenue en élevant la matrice T à la puissance W est de la forme :

$$A_{jk}(L, I, D_1, D_2) = L^W \sum_{i=0}^{W} \sum_{d_1=0}^{W} \sum_{d_2=0}^{W} I^i D_1^{d_1} D_2^{d_2} A_{jk}(i, d_1, d_2) .$$

En posant $L = I = 1$ et $D_1 = D_2$, on obtient :

$$A_{jk}(D) = \sum_{d=0}^{2W} D^d A_{jk}(d) ,$$

où $A_{jk}(d)$ est le nombre de mots de code de longueur W, de poids d, partant de l'état j et arrivant à l'état k. Le nombre total de mots de code de longueur w, de poids d, partant d'un état quelconque et arrivant à un état quelconque, est :

$$A(d) = \sum_j \sum_k A_{jk}(d) .$$

Pour obtenir le nombre de mots de code $A_{d_1=1}$ de longueur W tels que le bit d_1 de la branche se trouvant au milieu de la fenêtre soit égal à 1, nous considérons la matrice $T^wT_{d_1=1}T^w$ et procédons de même que précédemment pour obtenir $A(d)$.

Bibliographie

[1] M. Abramowitz, I. Stegun. Handbook of mathematical functions. Dover Publications Inc, New York, 9ème édition, 1972.

[2] L. R. Bahl, J. Cocke, F. Jelinek, J. Raviv. « Optimal decoding of linear codes for minimizing symbol error rate ». IEEE Transactions on Information Theory, vol. 20, pp. 284–287, mars 1974.

[3] G. Battail. Théorie de l'information. Masson, Paris, 1997.

[4] G. Battail, H. Magalhaes de Oliveira. « Probabilité d'erreur du codage aléatoire avec décodage optimal sur le canal à bruit gaussien additif affecté ou non de fluctuations d'amplitude ». Annales des Télécommunications, vol. 48, no 1-2, pp. 15-28, 1993.

[5] G. Battail, J. Fang. « Décodage pondéré optimal des codes linéaires en blocs. II. - Analyse et résultats de simulation ». Annales des Télécommunications, vol. 41, no 11-12, pp. 1-25, 1986.

[6] S. Benedetto, G. Montorsi, D. Divsalar, F. Pollara. « Serial concatenation of interleaved codes : performance analysis, design and iterative decoding ». TDA Progress Report 42-126, JPL, avril-juin 1996.

[7] S. Benedetto, G. Montorsi. « Serial concatenation of block and convolutional codes ». Electronic Letters, vol. 32, pp. 887-888, 1996.

[8] S. Benedetto, G. Montorsi. « Unveiling turbo-codes : some results on parallel concatenated coding schemes ». IEEE Transactions on Information Theory, vol. 42, no 2, pp. 409-429, mars 1996.

[9] S. Benedetto, G. Montorsi. « Design of parallel concatenated convolutional codes ». IEEE Transactions on Communication, vol. 44, no 5, pp. 591-600, mai 1996.

[10] E. R. Berlakamp. Algebraic coding theory. Aegean Park Press, Edition révisée, 1984.

[11] C. Berrou, A. Glavieux, P. Thitimajshima. « Near Shannon limit error-correcting coding and decoding : turbo-codes ». Proceedings of ICC'93, Genève, pp. 1064-1070, mai 1993.

[12] E. Biglieri, J. Proakis, S. Shamaï. « Fading channels : information-theoretic and communications aspects ». IEEE Transactions on Information Theory, vol. 44, no 6, octobre 1998.

[13] J. Boutros, O. Pothier. « Convergence analysis of turbo decoding ». Workshop on Information Theory, Toronto, juin 1997.

[14] J. Boutros, O. Pothier, G. Zémor. « Generalized low density (Tanner) codes ». Proceedings of ICC'99, Vancouver, 6-10 juin 1999.

[15] J. Boutros. « APP decoding of linear transformations ». Proceedings of the ISITA Conference, Honolulu, Hawaï, 5-8 novembre 2000.

[16] J. Boutros, G. Caire. « Iterative multiuser joint decoding : unified framework and asymptotic analysis ». Soumis à IEEE Transactions on Information Theory, août 2000.

[17] R. Buz. « Information theoretic limits on communication over multipath channels with ideal and non-ideal channel state information ». Rapport de thèse de doctorat, Dépt. ECE, Queen's University, Canada, 1994.

[18] Consultative Committee for Space Data Systems. « Telemetry : summary of concept and rationale ». CCSDS 100.0-G-1, Green Book, décembre 1987.

[19] D. Chase. « A class of algorithms for decoding block codes with channel measurements information ». IEEE Transactions on Information Theory, vol. 18, pp. 170-182, janvier 1972.

[20] S. Chung, R. L. Urbanke, T. L. Richardson. « Analysis of sum-product decoding of low density parity check codes using a gaussian approximation ». Soumis à IEEE Transactions on Information Theory, 1999.

[21] S. Chung, G. D. Forney, T. L. Richardson, R. L. Urbanke. « On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit ». Accepté pour publication dans IEEE Communications Letters.

[22] G. F. Conte, C. de Boor. Elementary numerical analysis : an algorithmic approach. 3ème édition, Mc Graw Hill, 1980.

[23] J. H. Conway, N. J. Sloane. Sphere packings, lattices and groups. 3ème édition, Springer-Verlag, New York, 1998.

[24] T. M. Cover, J. A. Thomas. Elements of information theory. Wiley, 1991.

[25] M. C. Davey. « *Error-correction using low-density parity-check codes* ». Rapport de thèse de doctorat, University of Cambridge, décembre 1999.

[26] D. G. Daut, J. W. Modestino, L. D. Wismer. « New short constraint-length convolutional code construction for selected rational rates ». IEEE Transactions on Information Theory, vol. 28, no 5, pp. 793-799, septembre 1982.

[27] D. Divsalar, S. Dolinar, R. J. McEliece, F. Pollara. « Transfer function bounds on the performance of turbo codes ». TDA Progress Report 42-122, avril-juin 1995.

[28] S. Dolinar, D. Divsalar, F. Pollara. « Code performance as a function of block size ». TMO Progress Report 42-133, mai 1998.

[29] S. Dolinar, D. Divsalar. « Weight distributions for turbo codes using random and nonrandom permutations ». TMO Progress Report 42-122, août 1995.

[30] T. M. Duman, M. Salehi. « New performance bounds for turbo codes ». IEEE Transactions on Communications, vol. 46, no 6, pp. 717-723, juin 1998.

[31] P. Elias. « Error-free coding ». IRE Transactions on Information Theory, pp. 29-37, 1954.

[32] T. Ericson. « A gaussian channel with slow fading ». IEEE Transactions on Information Theory, vol. 16, no 3, pp. 353-355, mai 1970.

[33] G. D. Forney. Concatenated codes. MIT Press, 1966.

[34] G. D. Forney. « Generalized minimum distance ». IEEE Transactions on Information Theory, vol. 12, pp. 125-131, 1966.

[35] G. D. Forney. « The Viterbi algorithm ». Proceedings IEEE, vol. 61, pp. 268-278, 1973.

[36] G. D. FORNEY. « Coset codes II : binary lattices and related codes ». IEEE Transactions on Information Theory, vol. 34, pp. 1152-1187, septembre 1988.

[37] M. P. C. FOSSORIER, S. LIN. « Soft decision decoding of linear block codes based on ordered statistics ». IEEE Transactions on Information Theory, vol. 41, no 5, septembre 1995.

[38] M. P. C. FOSSORIER, S. LIN, J. SNYDERS. « Reliability based syndrome decoding of linear block codes ». IEEE Transactions on Information Theory, vol. 44, no 1, janvier 1998.

[39] M. P. C. FOSSORIER, S. LIN. « Soft-input soft-output decoding of linear block codes based on ordered statistics ». Proceedings of Globecom 98, pp 2828-2833, 1998.

[40] B. J. FREY, D. J. MACKAY. « A revolution : belief propagation in graphs with cycles ». Advances in Neural Information Processing Systems 10, MIT Press, Cambridge MA, 1998.

[41] B. J. FREY, D. J. MACKAY. « Irregular turbo-codes ». Proceedings of the 37^{th} Allerton Conference on Communication, Control and Computing, 22-24 septembre 1999.

[42] R. G. GALLAGER. Low density parity check codes. MIT Press, 1963.

[43] R. G. GALLAGER. Information theory and reliable communication. John Wiley & Sons, 1968.

[44] H. EL GAMAL. « *On the theory and application of space-time and graph based codes* ». Rapport de thèse de doctorat, University of Maryland, 1999.

[45] H. EL GAMAL, A. .R. HAMMONS JR. « Analyzing the turbo decoder using the gaussian approximation ». Proceedings of the International Symposium on Information Theory, Sorrento, Italy, 25-30 juin 2000.

[46] S. W. GOLOMB, R. E. PEILE, R. A. SCHOLTZ. Basic concepts in information theory and coding. Applications of Communications Theory, 1994.

[47] A. J. GOLDSMITH, P. P. VARAIYA. « Capacity of fading channels with channel side information ». IEEE Transactions on Information Theory, vol. 43, no 6, novembre 1997.

[48] J. HAGENAUER, E. OFFER, L. PAPKE. « Iterative decoding of binary block and convolutional codes ». IEEE Transactions on Information Theory, vol. 42, no 2, pp. 429-445, mars 1996.

[49] S. HAYKIN. Digital communications. Wiley, 1988.

[50] N. KAHALE, R. URBANKE. « On the minimum distance of parallel and serially concatenated code ». soumis à IEEE Transactions on Information Theory.

[51] F. R. KSCHISCHANG, V. SOROKINE. « On the trellis structure of block codes ». IEEE Transactions on Information Theory, vol. 41, no 6, novembre 1995.

[52] F. R. KSCHISCHANG, B. FREY. « Iterative decoding of compound codes by probability in graphical models ». IEEE Journal on Selected Area in Communications, vol. 16, no 2, pp. 219-230, février 1998.

[53] D. E. LAZIC, T. BETH, M. CALIC. « How close are turbo codes to optimal codes ? ». Proceedings of the International Symposium on Turbo Codes, pp. 192-195, Brest, septembre 1997.

[54] W. C. Y. LEE. « Estimate of channel capacity in Rayleigh fading environment ». IEEE Transactions on Vehicular Technology, vol. 39, no 3, août 1990.

[55] M. LENTMAIER, K. S. ZIGANGIROV. « On generalized low-density parity-check codes based on Hamming component codes ». IEEE Communication Letters, vol. 3, no 8, pp. 248, août 1999.

[56] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D .A. Spielman. « Improved low-density parity-check codes using irregular graphs and belief propagation ». Proceedings of the International Symposium on Information Theory (ISIT'98), p. 117, 16-21 août 1998.

[57] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D .A. Spielman. « Analysis of low density codes and improved designs using irregular graphs ». Proceedings of the 30th ACM STOC, 23-26 mai 1998.

[58] D. J. C. MacKay. « Good error-correcting codes based on very sparse matrices ». IEEE Transactions on Information Theory, vol. 45, pp. 388-431, mars 1999.

[59] J. L. Massey. « Foundation and methods of channel encoding ». Proceedings of International Conference on Information Theory and Systems, vol. 65, pp. 148-157, NTG-Fachberichte, Berlin, 1978.

[60] F. J. MacWilliams, N. J. A. Sloane. The theory of error-correcting codes. North-Holland Mathematical Library, 1977.

[61] J. A. Nelder, R. Mead. Computer Journal, vol. 7, pp. 308-313.

[62] O. Pothier. « *Codes composites construits à partir de graphes et de leur décodage itératif* ». Rapport de thèse de doctorat, ENST, janvier 2000.

[63] J. G. Proakis. Digital communications. New York, McGraw Hill, 3ème édition, 1995.

[64] R. Pyndiah, A. Glavieux, A. Picart, S. Jacq. « Near optimum product codes ». Proceedings of GLOBECOM'94, San Francisco, pp. 339-343, novembre 1994.

[65] R. Pyndiah. « Near optimum decoding of product codes : block turbo codes ». IEEE Transactions on Communications, vol. 46, no 8, août 1998.

[66] T. Richardson, A. Shokrollahi, R. Urbanke. « Design of provably good low-density parity check codes ». Soumis à IEEE Transactions on Information Theory, avril 1999.

[67] T. Richardson, R. Urbanke. « Efficient encoding of low-density parity-check codes ». Soumis à IEEE Transactions on Information Theory, septembre 1999.

[68] T. Richardson, R. Urbanke. « The capacity of low-density parity check codes under message-passing decoding ». Soumis à IEEE Transactions on Information Theory, 1999.

[69] I. Sason, S. Shamaï (Shitz). « Improved upper bounds on the ML decoding error probability of parallel and serial concatenated turbo-codes via their ensemble distance spectrum ». soumis à IEEE Transactions on Information Theory, mars 1999.

[70] H. E. Sawaya, J. Boutros. « Error probability of optimal lattice codes ». Proceedings of SITA'99, Niigata, Japon, 30 novembre-3 décembre 1999.

[71] S. Shamaï (Shitz). *Communications privées.*

[72] C. E. Shannon. « A mathematical theory of communication ». The Bell Labs Technical Journal, vol. 27, pp. 379-457, pp. 623-656, juillet-octobre 1948.

[73] C. E. Shannon. « Probability of error for optimal codes in a Gaussian channel ». The Bell Labs Technical Journal, vol. 38, no 3, mai 1959.

[74] D. Slepian. « Bounds on communications ». The Bell System Technical Journal, vol. 42, pp. 687-707, 1963.

[75] O. Y. Takeshita, O. M. Collins, P. C. Massey, D. J. Costello. « A note on asymmetric Turbo Codes ». IEEE Communications Letters, vol. 3, no 3, mars 1999.

[76] R. M. Tanner. « A recursive approach to low complexity codes ». IEEE Transactions on Information Theory, vol. 27, septembre 1981.

[77] S. Ten Brink. « Iterative Decoding Trajectories of Parallel Concatenated Codes ». Proceedings of Spring VTC, Vancouver, juin 1999.

[78] S. Ten Brink. « Design of serially concatenated codes based on iterative decoding convergence ». Proceedings of the International Symposium on Turbo Codes, Brest, France, 4-7 septembre 2000.

[79] L. M. G. M. Tolhuizen, C. P. M. J. Bagen. « On the weight enumerator of product codes ». Discrete Mathematics, North-Holland, pp. 483-488, 1992.

[80] G. Ungerboeck. « Channel coding with multilevel/phase signals ». IEEE Transactions on Information Theory, vol. 28, no 1, janvier 1982.

[81] A. Vardy. « *Handbook of coding theory* », Trellis structure of codes. Édition V. Pless, W. Cary Huffman and R. Brualdi, Elsevier Science Publishers, Amsterdam.

[82] A. J. Viterbi. « Error bounds for convolutional codes and an asymptotically optimum decoding algorithm ». IEEE Transactions on Information Theory, vol. 13, pp. 260-269, 1967.

[83] N. Wiberg, H. A. Loeliger, R. Kötter. « Codes and iterative decoding on general graphs ». European Transactions on Telecommunications, vol. 6, septembre-octobre 1995.

[84] N. Wiberg. « *Codes and decoding on general graphs* ». Rapport de thèse de doctorat, Linköpings University, 1996.

[85] J. M. Wozencraft, I. M. Jacobs. Principles of communication engineering. John Wiley & Sons, 1965.

Rapports et Publications

[86] S. Vialle, J. Boutros. « Capacity of Gaussian and Rayleigh channels ». Rapport interne Motorola, décembre 1998.

[87] S. Vialle. « Principles and performance of product codes iteratively decoded ». Rapport interne Motorola, août 1999.

[88] S. Vialle, J. Boutros. « A Gallager-Tanner construction based on convolutional codes ». Proceedings of the Workshop on Coding and Cryptography, WCC'99, Paris, 11-14 janvier 1999.

[89] S. Vialle, J. Boutros. « Performance of optimal codes on Gaussian and Rayleigh channels : a geometrical approach ». Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing, Allerton, 22-24 septembre 1999.

[90] S. Vialle, J. Boutros. « Performance limits of concatenated codes under iterative decoding ». Proceedings of the International Symposimum on Information Theory, ISIT'00, Sorrento, 25-30 juin 2000.

[91] H. Sawaya, S. Vialle, J. Boutros, G. Zémor. « Performance limits of compound codes under symbol-based iterative decoding ». Soumis au Workshop on Coding and Cryptography, Paris, 8-12 janvier 2001.

[92] S. Vialle, J. Boutros, O. Pothier, G. Zémor. « An asymptotically good family of generalized low density codes ». En préparation en vue d'être soumis à IEEE Transactions on Information Theory.

[93] J. BOUTROS, S. VIALLE, G. CAIRE, E. VITERBO. « Performance limits of coded systems and compound codes under iterative decoding ». En préparation en vue d'être soumis à IEEE Transactions on Information Theory.