Reconnaissance en aveugle de codeur à base de code convolutif : Contribution à la mise en oeuvre d'un récepteur intelligent

Mélanie Marazin

THÈSE / UNIVERSITÉ DE BRETAGNE OCCIDENTALE
sous le sceau de l'Université Européenne de Bretagne

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE
Mention : Sciences et Technologies de l'Information et de la Communication - Spécialité Communications Numériques
École Doctorale SICMA-ED 373

présentée par
Mélanie Marazin
Préparée au Lab-STICC UMR-CNRS 3192

Reconnaissance en aveugle de codeur à base de code convolutif : Contribution à la mise en œuvre d'un récepteur intelligent

Thèse soutenue le 10 décembre 2009
devant le jury composé de :

Claude BERROU
Professeur, Télécom-Bretagne, Département d'électronique, Membre de l'Académie des Sciences / *Président du jury*

Jean-Pierre CANCES
Professeur des Universités, ENSIL, Laboratoire XLIM / *Rapporteur*

Benoît GELLER
Enseignant-Chercheur, ENSTA, Laboratoire d'Electronique et d'Informatique / *Rapporteur*

Jean-François DIOURIS
Professeur des Universités, Polytech'Nantes, IREENA / *Examinateur*

Roland GAUTIER
Maître de Conférences, UBO, Lab-STICC / *Examinateur*

Gilles BUREL
Professeur des Universités, UBO, Lab-STICC / *Directeur de thèse*

Johannes HUISMAN
Professeur des Universités, UBO, Laboratoire de Mathématiques / *Invité*

Eric RANNOU
Maître de Conférences, UBO, Laboratoire de Mathématiques / *Invité*

THÈSE / UNIVERSITÉ DE BRETAGNE OCCIDENTALE
sous le sceau de l'Université Européenne de Bretagne

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE
Mention : Sciences et Technologies de l'Information et de la Communication - Spécialité Communications Numériques
École Doctorale SICMA-ED 373

présentée par
Mélanie Marazin
Préparée au Lab-STICC UMR-CNRS 3192

Reconnaissance en aveugle de codeur à base de code convolutif : Contribution à la mise en œuvre d'un récepteur intelligent

Thèse soutenue le 10 décembre 2009
devant le jury composé de :

Claude BERROU
Professeur, Télécom-Bretagne, Département d'électronique, Membre de l'Académie des Sciences / *Président du jury*

Jean-Pierre CANCES
Professeur des Universités, ENSIL, Laboratoire XLIM / *Rapporteur*

Benoît GELLER
Enseignant-Chercheur, ENSTA, Laboratoire d'Electronique et d'Informatique / *Rapporteur*

Jean-François DIOURIS
Professeur des Universités, Polytech'Nantes, IREENA / *Examinateur*

Roland GAUTIER
Maître de Conférences, UBO, Lab-STICC / *Examinateur*

Gilles BUREL
Professeur des Universités, UBO, Lab-STICC / *Directeur de thèse*

Johannes HUISMAN
Professeur des Universités, UBO, Laboratoire de Mathématiques / *Invité*

Eric RANNOU
Maître de Conférences, UBO, Laboratoire de Mathématiques / *Invité*

Remerciements

Le travail présenté dans ce mémoire a été réalisé au Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC UMR CNRS 3192) au sein de l'équipe Intelligence et furtivité des communications sur le site de l'Université de Bretagne Occidentale (UBO) de Brest. Je tiens à remercier la région Bretagne pour avoir financé ces trois années de thèse.

J'adresse mes sincères remerciements à l'ensemble de mon jury de thèse. Je remercie sincèrement Monsieur Claude Berrrou, professeur à Télécom-Bretagne, qui m'a fait l'honneur de présider ce jury de thèse. J'exprime ma plus profonde gratitude à Messieurs Jean-Pierre Cancès, professeur à l'ENSIL, et Benoît Geller, enseignant-chercheur à l'ENSTA, pour l'honneur qu'ils m'ont fait en acceptant de rapporter sur ces travaux. Vos regards extérieurs et la perspicacité de vos remarques m'ont permis d'améliorer la qualité de ce manuscrit.

Je remercie chaleureusement mon équipe d'encadrement. Je remercie tout d'abord Monsieur Gilles Burel, professeur à l'UBO, qui a dirigé ces travaux de thèse. J'ai beaucoup apprécié la pertinence et la qualité de ses interventions mais surtout sa gentillesse et sa disponibilité. Malgré votre emploi du temps surchargé, vous avez toujours su rester disponible, que ce soit pour l'avancée de mes travaux thèse ou pour mon avenir professionnel. J'adresse ma profonde gratitude à Monsieur Roland Gautier, maître de conférences à l'UBO, qui m'a permis de réaliser cette thèse dans d'excellentes conditions. Un grand merci pour ta bonne humeur, tes précieux conseils, ta disponibilité et tes encouragements. Sans toi, cette thèse n'aurait probablement jamais abouti... Merci d'y avoir cru jusqu'au bout.

Je tiens également à remercier l'ensemble des membres de l'équipe, Koffi, Philippe, Emanuel, Stéphane, Baghious, Ludovic, Ali et Rizwan, pour leurs conseils et leurs soutiens permanents. Un merci particulier à Koffi et Philippe qui m'ont donné envie de faire une thèse lorsque j'étais encore étudiante à l'UBO. Je remercie également l'ensemble des membres permanents du laboratoire et du département d'électronique, chercheurs, personnels administratifs et techniques, pour cette ambiance de travail très agréable qu'ils ont su créer. Un merci particulier à Marc, Paul et Roland qui m'ont permis de terminer cette thèse sereinement.

Je n'oublie pas l'ensemble des doctorants, actuels et anciens, qui ont grandement contribué à créer une ambiance de travail chaleureuse. Un grand merci à vous tous pour ces souvenirs inoubliables, en particulier lors de certaines soirées. Un clin d'oeil particulier à mes anciens collègues de bureau. Merci à Joe et Baptiste pour l'aide que vous m'avez apporté durant ma première année de thèse et pour la chaleureuse ambiance dans notre bureau. Merci à Yann pour ton passage bref mais remarquable dans ce bureau et merci à Vincent pour tes précieux conseils et la vie que tu as apporté dans ce bureau lors de ma dernière année de thèse.

Je ne pourrais pas ne pas remercier l'ensemble de mes amis, qu'ils soient bretons, toulousains, parisiens, etc, qui m'ont soutenue et supportée durant ces trois dernières années. J'adresse un immense merci à mon ami Julien. Merci pour ton soutien, ton aide et tout ce que tu m'as apporté durant cette dernière ligne droite.

Enfin mes derniers remerciements s'adressent à ma famille. En particuliers mes parents, Jacqueline et Christian, et mes grand-parents, Claude et Daniel, à qui je dois tout. Merci pour votre soutien inconditionnel. J'adresse un merci particulier à ma grand-mère et je leur dédie aveuglément l'ensemble de ces travaux.

Table des matières

Liste des acronymes et abréviations **v**

Notations **vii**

Liste des figures **xii**

Liste des algorithmes **xiii**

Liste des tableaux **xv**

Introduction **1**

1 Contexte de l'étude et théorie algébrique des codes linéaires **5**

1.1 Introduction . . . 5

1.2 Chaîne de transmission numérique . . . 5

1.2.1 Le contexte non-coopératif . . . 6

1.2.2 Le canal de transmission . . . 7

1.2.2.1 Le canal stationnaire . . . 7

1.2.2.2 Le canal non stationnaire . . . 8

1.2.2.3 Le canal binaire . . . 8

1.2.3 Le codage de canal . . . 9

1.3 Les codes en blocs . . . 10

1.3.1 Le codage . . . 11

1.3.2 Les codes duaux . . . 13

1.3.3 La capacité de correction . . . 14

1.4 Les codes convolutifs . . . 15

1.4.1 Présentation des codes convolutifs . . . 15

1.4.2 Les polynômes générateurs . . . 17

1.4.3 Les codeurs de forme NRNSC . . . 17

1.4.3.1 Représentation matricielle . . . 17

1.4.3.2 Représentation polynomiale . . . 20

1.4.3.3 La mémoire d'un code convolutif . . . 22

1.4.4 Les codeurs de forme RSC . . . 23
1.4.4.1 Les codeurs équivalents . . . 23
1.4.4.2 Passage d'un codeur NRNSC sous sa forme RSC équivalente . . . 23
1.4.5 Les codes duaux . . . 25
1.4.6 Représentation sous forme de matrices d'état . . . 28
1.4.6.1 Le codage . . . 28
1.4.6.2 Matrices d'état d'un codeur NRNSC . . . 29
1.4.6.3 Matrices d'état d'un codeur RSC . . . 30
1.5 Propriétés des codes convolutifs . . . 32
1.5.1 Le degré d'un code convolutif . . . 32
1.5.2 Les matrices génératrices catastrophiques . . . 33
1.5.3 Les codes convolutifs optimaux . . . 34
1.6 Conclusions . . . 36

2 Reconnaissance aveugle d'un code convolutif 37
2.1 Introduction . . . 37
2.2 Reconnaissance aveugle d'un code convolutif dans le cas non-bruité . . . 38
2.2.1 Détection et identification de paramètres . . . 38
2.2.1.1 Code en bloc . . . 40
2.2.1.2 Code convolutif . . . 41
2.2.2 Étude du rang des matrices R_l . . . 44
2.2.2.1 Code de rendement $(n-1)/n$ . . . 44
2.2.2.2 Code de rendement k/n . . . 45
2.2.2.3 Quelques cas particuliers . . . 47
2.2.3 Synchronisation aveugle . . . 48
2.2.4 Identification du code dual . . . 50
2.2.4.1 Code de rendement $(n-1)/n$ . . . 50
2.2.4.2 Code de rendement k/n . . . 52
2.2.5 Identification d'une matrice génératrice . . . 55
2.2.5.1 Code de rendement $1/n$ . . . 55
2.2.5.2 Code de rendement k/n . . . 56
2.2.6 Reconnaissance aveugle du $C(3,2,3)$ code convolutif . . . 58
2.2.6.1 Présentation des paramètres du code . . . 58
2.2.6.2 Reconnaissance aveugle du code . . . 58
2.3 Reconnaissance aveugle d'un code convolutif dans le cas bruité . . . 61
2.3.1 Quelques notations et définitions . . . 62
2.3.2 Principe de notre méthode de reconnaissance aveugle . . . 63
2.3.2.1 La matrice triangulaire inférieure . . . 63
2.3.2.2 Étude du seuil de détection γ . . . 66
2.3.3 Algorithme de reconnaissance . . . 69

2.3.3.1 Estimation de n et P_e . . . 69
2.3.3.2 Estimation d'une base du code dual . . . 73
2.3.3.3 Estimation des paramètres k, K et d'une matrice génératrice . . . 73
2.4 Analyse de l'algorithme . . . 76
2.4.1 Le gain apporté par notre algorithme itératif . . . 77
2.4.2 Les probabilités de détection . . . 81
2.5 Conclusions . . . 83

3 Les codes convolutifs poinçonnés 85
3.1 Introduction . . . 85
3.2 Présentation des codes convolutifs poinçonnés . . . 85
3.2.1 Principe du poinçonnage . . . 86
3.2.2 Construction du code poinçonné . . . 87
3.3 Reconnaissance aveugle d'un code convolutif poinçonné . . . 93
3.3.1 Principe de la méthode de reconnaissance . . . 94
3.3.2 Reconnaissance aveugle pour un code parent de rendement $1/n$ . . . 96
3.3.3 Reconnaissance aveugle pour un code parent de rendement k/n . . . 102
3.4 Analyse de l'algorithme . . . 106
3.4.1 Le gain apporté par notre algorithme itératif . . . 107
3.4.2 Les probabilités de détection . . . 109
3.5 Conclusions . . . 111

4 Reconnaissance des codes concaténés et des turbocodes 113
4.1 Introduction . . . 113
4.2 Concaténation de codes . . . 113
4.2.1 Les entrelaceurs . . . 114
4.2.2 Les turbocodes série . . . 115
4.2.3 Les turbocodes parallèle . . . 116
4.2.3.1 Les codes concaténés en parallèle . . . 116
4.2.3.2 Les turbocodes . . . 117
4.3 Reconnaissance aveugle des codes concaténés en parallèle . . . 118
4.3.1 Identification des paramètres des codes concaténés en parallèle . . . 119
4.3.1.1 Les chutes de rang maximales . . . 119
4.3.1.2 Les chutes de rang moyennes . . . 121
4.3.2 Identification du nombre de sorties des deux codes . . . 123
4.3.3 Identification des deux codes et de l'entrelaceur . . . 126
4.4 Reconnaissance aveugle des turbocodes . . . 133
4.4.1 Identification des paramètres du turbocode et du premier codeur . . . 133
4.4.2 Identification du second codeur . . . 136
4.4.2.1 Les hypothèses et notations . . . 136

4.4.2.2 Méthode d'identification 140
4.5 Conclusions 147

Conclusion **149**

A Les polynômes générateurs d'un codeur RSC **153**
A.1 Déterminant et mineurs d'une matrice 153
A.2 Polynômes générateurs d'un codeur RSC 154
A.3 Mineurs d'ordre k de la matrice génératrice d'un codeur NRNSC 156

B Matrice de parité d'un code convolutif **157**

C Représentation d'état des codes convolutifs **161**
C.1 Représentation d'état 161
C.2 Quelques définitions sur les matrices 162
C.3 Matrices d'état d'un codeur NRNSC 163
C.3.1 Les matrices d'état d'un codeur de forme NRNSC 163
C.3.2 Validation de nos matrices d'état d'un codeur NRNSC 164
C.4 Matrices d'état d'un codeur RSC 166
C.4.1 Les matrices d'état d'un codeur de forme RSC 166
C.4.2 Validation de nos matrices d'état d'un codeur RSC 167

D Tables des codes convolutifs optimaux **173**

E Tables des codes convolutifs poinçonnés **175**

Bibliographie **184**

Liste des publications **189**

Liste des acronymes et abréviations

BBAG	Bruit Blanc Additif Gaussien
BER	Bit Error Rate
BSC	Binary Symmetric Channel
c-à-d	c'est-à-dire
CAN	Convertisseur Analogique Numérique
CCP	Code Concaténé en Parallèle
dB	Décibel
DVB-S	Digital Video Broadcasting - Satellite
DVB-T	Digital Video Broadcasting - Terrestrial
EM	Expectation-Maximization
GF	Galois Field
iid	Indépendant et Identiquement Distribué
LDPC	Low Density Parity Check
ms	milliseconde
Mbps	Mégabits par seconde
NRNSC	Non Recursive and Non Systematic Code
PCCC	Parallel Concatenated Convolutional Code
PCPC	PolyCyclic PseudoCirculante
PGCD	Plus Grand Commun Diviseur
QoS	Quality of Service
RSC	Recursive and Systematic Code
TEB	Taux d'Erreur Binaire
TEB_r	Taux d'Erreur Binaire résiduel
UMTS	Universal Mobile Telecommunications System

Notations

Notations du mémoire

Une matrice de taille $m \times n$ sera notée, par soucis de concision, une $m \times n$ matrice.

Pour toute matrice comprenant de nombreux "0", comme la matrice A dans l'exemple ci-dessous, les "0" seront omis afin de simplifier la notation.

$$A = \begin{pmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & a_3 & 0 \\ 0 & 0 & 0 & a_4 \end{pmatrix} = \begin{pmatrix} a_1 & & & \\ & a_2 & & \\ & & a_3 & \\ & & & a_4 \end{pmatrix}$$

Notations mathématiques

$a(l)$	Le l-ième bit du vecteur $\mathbf{a}$
$\mathbf{a} = (.)$	Vecteur composé d'éléments binaires
$A = (.)$	Matrice composée d'éléments binaires
$\mathbf{a}_i = (.)$	La colonne i de la matrice A
$a(D)$	Polynôme
$A(D) = [.]$	Matrice composée de polynômes
$(.)^T$	Vecteur ou matrice transposé(e)
$\lfloor x \rfloor$	Arrondi à l'entier inférieur de x
$\lceil x \rceil$	Arrondi à l'entier supérieur de x

Notations des codes linéaires

n	Nombre de sorties d'un codeur
k	Nombre d'entrées d'un codeur
r	Rendement d'un code
$\mathbf{m}$	Vecteur d'entrée d'un codeur
$\mathbf{c}$	Vecteur de sortie d'un codeur
$\tilde{\mathbf{c}}$	Vecteur de sortie du codeur désynchronisé
$\mathbf{y}$	Vecteur de sortie du codeur entaché d'erreurs
$w(\mathbf{a})$	Poids de Hamming du vecteur $\mathbf{a}$
P_e	Probabilité d'erreur du canal de transmission

Notations des codes en bloc

$C(n,k)$	Code en bloc de paramètres n et k
G	Matrice génératrice
H	Matrice de parité

Notations des codes convolutifs

K	Longueur de contrainte du codeur
μ_i	Mémoire de la i-ème entrée du codeur
μ	Mémoire du codeur
$\mu^{\perp}$	Mémoire du code dual
$C(n,k,K)$	Code convolutif de paramètres n, k et K (code parent dans le chapitre 3)
$m(D)$	Séquence d'entrée du codeur
$c(D)$	Séquence de sortie du codeur
$g_{i,j}(l)$	Le l-ième coefficient du (i,j)-ème polynôme générateur ($g_{i,j}(D)$)
$\mathbf{g}_{i,j}$	Vecteur composé des coefficients du (i,j)-ème polynôme générateur ($g_{i,j}(D)$)
$g_{i,j}(D)$	Le (i,j)-ème polynôme générateur
$G(D)$	Matrice génératrice
F_l	La l-ième sous-matrice de codage composée des éléments binaires des polynômes générateurs
F	Matrice de codage composée des sous-matrices de codage
$h_{i,j}(l)$	Le l-ième coefficient du (i,j)-ème polynôme ($h_{i,j}(D)$)
$\mathbf{h}_{i,j}$	Vecteur composé des coefficients du (i,j)-ème polynôme ($h_{i,j}(D)$)
$h_{i,j}(D)$	Le (i,j)-ème polynôme
$H(D)$	Matrice de parité
H_l	La l-ième sous-matrice de parité composée des éléments binaires des polynômes
H	Matrice de parité composée des sous-matrices de parité

Notations spécifiques à l'évaluation des performances de nos algorithmes de reconnaissance (chapitre 2 et 3)

$\mathcal{P}_{det}$	Probabilité de détection
$\mathcal{P}_{fa}$	Probabilité de fausse alarme
$\mathcal{P}_{nd}$	Probabilité de non détection

Notations spécifiques du chapitre 2

n_a	Taille de la première matrice de rang déficient
$\mathbf{h}_i$	La i-ème relation de parité du code
R_l	Une $M \times l$ matrice composée des éléments binaires de $\mathbf{c}$ ou $\tilde{\mathbf{c}}$
$\tilde{R}_l$	Une $M \times l$ matrice composée des éléments binaires de $\mathbf{y}$
T_l	Matrice triangulaire inférieure (triangularisation par pivot de Gauss : $T_l = A_l.\tilde{R}_l.B_l$)
$rg(R_l)$	Rang de la matrice R_l
$Q(l)$	La déficience de rang de la matrice R_l
γ	Seuil de détection
P_{det}	Probabilité de détection (dans le cadre de l'estimation du seuil γ)
P_{fa}	Probabilité de fausse alarme (dans le cadre de l'estimation du seuil γ)
P_{nd}	Probabilité de non détection (dans le cadre de l'estimation du seuil γ)

Notations spécifiques du chapitre 3

$\mathcal{M}$	La profondeur du poinçonnage
P	Matrice de poinçonnage
N'	Nombre de bits non nuls dans la matrice P
$G^{[\mathcal{M}]}(D)$	Matrice génératrice du code regroupé à la profondeur de $\mathcal{M}$
$\mathcal{Q}_{(i,j)}^{[\mathcal{M}]}$	La M-ième matrice polycyclique pseudocirculante du polynôme $g_{i,j}(D)$
$q_{i,j_{(l,m)}}(D)$	Le (l,m)-ème polynôme de la matrice $\mathcal{Q}_{(i,j)}^{[\mathcal{M}]}$
n_p	Nombre de sorties du code poinçonné
k_p	Nombre d'entrées du code poinçonné
r_p	Rendement du code poinçonné
K_p	Longueur de contrainte du code poinçonné
$C_p(n_p, k_p, K_p)$	Code poinçonné de paramètres n_p, k_p et K_p
$g_{p_{(i,j)}}(D)$	Le (i,j)-ème polynôme générateur du code poinçonné
$G_p(D)$	Matrice génératrice du code poinçonné

Notations spécifiques du chapitre 4

$C_1(n_1, k_1, K_1)$	Code convolutif de paramètres n_1, k_1 et K_1
$C_2(n_2, k_2, K_2)$	Code convolutif de paramètres n_2, k_2 et K_2
$\mu_j^{\perp}$	Mémoire du code dual $C_j^{\perp}$
μ_j^i	Mémoire de la i-ème entrée du code C_j
r_s	Rendement d'un turbocode série
r_{par}	Rendement d'un code concaténé en parallèle
r_t	Rendement d'un turbocode parallèle
$\mathbf{m}'$	Vecteur d'entrée entrelacé
$\mathbf{c}$	Vecteur de sortie du codeur C_1
$\mathbf{c}'$	Vecteur de sortie du codeur C_2
$\mathbf{x}$	Vecteur de sortie du code concaténé
l_e	Taille de l'entrelaceur
p	Vecteur de permutation
E	Matrice de permutation
E_g	Matrice d'entrelacement global

Liste des figures

1 Synoptique d'une transmission numérique 1

1.1 Chaîne de transmission numérique 5
1.2 Codage et chiffrement 7
1.3 Modélisation du canal BBAG 7
1.4 Description d'un canal binaire 8
1.5 Canal binaire symétrique 9
1.6 Modèle du canal de Gilbert-Elliot 9
1.7 Mots de code 12
1.8 Poids de Hamming des mots de code 15
1.9 Schéma d'implémentation d'un code convolutif à l'aide de registre à décalage 16
1.10 Schéma d'implémentation d'un codeur NRNSC 21
1.11 Schéma d'implémentation d'un codeur RSC 24

2.1 Réorganisation des données interceptées sous forme de matrice 39
2.2 Rang des matrices R_l dans le cas de données non codées 39
2.3 Rang des matrices R_l dans le cas d'un code de Hamming 41
2.4 Rang des matrices R_l du $C(2,1,3)$ code 43
2.5 Rang des matrices R_l du $C(3,1,4)$ code 46
2.6 Rang des matrices R_l du $C(4,1,6)$ code 47
2.7 Rang des matrices R_l du $C(3,1,4)$ code avec un décalage de $t_0 = 4$ 49
2.8 Synchronisation aveugle du $C(3,1,4)$ code 49
2.9 Rang des matrices R_l du $C(3,2,3)$ code avec un décalage de $t_0 = 11$ 59
2.10 Synchronisation aveugle du $C(3,2,3)$ 59
2.11 Rang des matrices R_l du $C(3,2,3)$ code 60
2.12 Matrice triangulaire inférieure 64
2.13 Représentation du produit $A_l.\tilde{R}_l$ 64
2.14 Matrice triangulaire inférieure T_l pour $l \neq \alpha.n$ ou $l < n_a$ 65
2.15 Matrice triangulaire inférieure T_l pour $l = \alpha.n \geq n_a$ 65
2.16 Seuil de détection γ en fonction de P_e 68
2.17 Seuil de détection γ en fonction de $w(\mathbf{h})$ 69

2.18 Déficiences de rang du $C(2,1,7)$ code pour $P_e = 0$ et $P_e = 0.02$ 71
2.19 $C(2,1,7)$: Probabilité de détection en fonction de P_e 78
2.20 $C(3,2,3)$: Probabilité de détection en fonction de P_e 78
2.21 $C(3,1,4)$: Probabilité de détection en fonction de P_e 79
2.22 Nombre d'itérations pour le $C(2,1,7)$ code 80
2.23 Les TEB_r théoriques 82
2.24 Les probabilités de détection 83

3.1 Comparaison des TEB_r entre les codes poinçonnés et non poinçonnés 91
3.2 Matrice de codage du code parent et du code poinçonné 93
3.3 $C(2,1,7)$ et $\mathcal{M} = 2$: Probabilité de détection en fonction de P_e 107
3.4 $C(3,1,7)$ et $\mathcal{M} = 3$: Probabilité de détection en fonction de P_e 108
3.5 $C(3,2,3)$ et $\mathcal{M} = 2$: Probabilité de détection en fonction de P_e 108
3.6 Les TEB_r théoriques des codes poinçonnés 110
3.7 Les probabilités de détection 111

4.1 Schéma d'un code concaténé sans et avec entrelacement 114
4.2 Schéma d'un turbocode 114
4.3 Schéma d'une concaténation série 116
4.4 Représentation d'un code concaténé en parallèle 117
4.5 Représentation d'un turbocode 117
4.6 Rang des matrices R_l du CCP[1] (cf tableau 4.1) 121
4.7 Identification de n_1 du CCP[1] (cf tableau 4.1) 125
4.8 Rang des matrices R_l du CCP[2] (cf tableau 4.1) 126
4.9 Identification de n_1 du CCP[2] (cf tableau 4.1) 126
4.10 Rang des matrices R_l du CCP[3] (cf tableau 4.1) 130
4.11 Identification de n_1 du CCP[3] (cf tableau 4.1) 130
4.12 Rang des matrices R_l du turbocode[1] (cf tableau 4.1) 135
4.13 Identification de n_1 du turbocode[1] (cf tableau 4.1) 135
4.14 Entrelaceur suivi du codeur C_2 136
4.15 Les matrices de codage du turbocode[1] (cf tableau 4.1) 139
4.16 Matrice de codage entrelacée du turbocode[1] (cf tableau 4.1) 139
4.17 Matrice de codage entrelacée identifiée du turbocode[1] (cf tableau 4.1) 143

Liste des Algorithmes

1 Recherche de matrices de rang probablement déficient 71
2 Estimation de $\hat{n}$, γ_{opt} et $\hat{P}_e$ 73
3 Estimation d'une base du code dual 74
4 Estimation de $\hat{k}$ et $\hat{\mu^{\perp}}$ 74
5 Estimation des $(\hat{n} - \hat{k})$ relations de parité 75
6 Algorithme d'identification du code parent et du motif de poinçonnage 98
7 Identification du vecteur de permutation 128
8 Étapes de notre algorithme de reconnaissance du code C_2 d'un turbcode 142

Liste des tableaux

1.1 Propriétés de trois matrices équivalentes 36

2.1 Étude de la première matrice de rang déficient pour un code de rendement 1/3 . . . 46
2.2 Déficiences de rang détectées du $C(2,1,7)$ code pour $P_e = 0.02$ 71
2.3 Gain de détection pour le $C(2,1,7)$ code 77
2.4 Gain de détection pour le $C(3,2,3)$ code 78
2.5 Gain de détection pour le $C(2,1,7)$ code avec plusieurs valeurs de M 81

3.1 Gain de détection pour le $C(2,1,7)$ code poinçonné à la profondeur de $\mathcal{M} = 2$ 107
3.2 Gain de détection pour le $C(3,1,7)$ code poinçonné à la profondeur de $\mathcal{M} = 3$ 108
3.3 Gain de détection pour le $C(3,2,3)$ code poinçonné à la profondeur de $\mathcal{M} = 2$ 109

4.1 Tableaux récapitulatif des schémas de codage étudiés 118

Introduction

De nos jours, communiquer de n'importe quel lieu et avec n'importe qui n'est plus considéré comme un luxe mais fait partie courante de notre vie. Depuis l'avènement de l'ère des transmissions numériques, ces communications sont devenues accessibles à tous.

Les transmissions numériques

Une transmission numérique permet de transmettre une information numérique, c'est-à-dire une suite de "0" et de "1", d'un émetteur à un récepteur via un canal de transmission. Le schéma de principe d'une chaîne de communication numérique est présenté sur la figure 1. Un signal numérique n'existant pas d'un point de vue physique, il est nécessaire de le transformer en un signal analogique afin de le transmettre via le canal. Ce canal de transmission permet de transmettre un signal d'un point A à un point B, mais comme tout canal de transmission il va engendrer des erreurs sur le signal. Le signal qui sera reçu au point B ne correspondra pas exactement au signal qui a été émis en A. Afin de lutter contre ces erreurs de transmission, l'émetteur est équipé d'un bloc de codage canal et le récepteur d'un bloc de décodage. Au niveau de l'émetteur, ce bloc a pour rôle de rajouter de la redondance au message qui doit être émis. Cette redondance va permettre, au niveau du récepteur, de détecter et/ou de corriger d'éventuelles erreurs qui seraient intervenues lors de la transmission du message. Dès lors, toute transmission numérique moderne, qu'elle soit filaire, terrestre ou satellitaire, est équipée d'un bloc de codage de canal afin de lutter contre les erreurs de transmission.

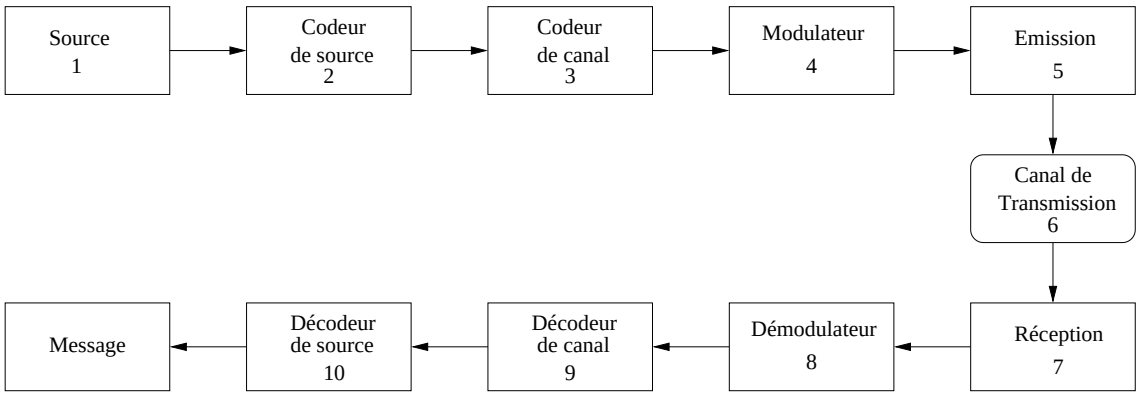


Figure 1 — Synoptique d'une transmission numérique

1. *La source :* génère l'information à transmettre (0 et 1) ;
2. *Le codage de source :* comprime les données en supprimant les redondances intrinsèques au message ;

3. *Le codage canal :* introduit de la redondance contrôlée, qui permettra de protéger la transmission contre les erreurs ;
4. *Le modulateur :* transforme le train binaire en une suite de symboles complexes ;
5. *L'émission :* transforme la suite de symboles en un signal physique qui pourra se propager ;
6. *Le canal de transmission :* représente le support permettant la propagation du signal (la transmission sur le canal perturbe le signal) ;
7. *La réception :* transforme le signal reçu en une suite de symboles ;
8. *Le démodulateur :* retraduit la suite de symboles en message binaire ;
9. *Le décodeur de canal :* détecte et/ou corrige les erreurs binaires éventuelles en exploitant la redondance introduite à l'émission ;
10. *Le décodeur de source :* décomprime les données pour restituer le message d'origine ;

Contexte et objectif

Lors d'une transmission classique, ou en contexte dit coopératif, le récepteur connaît les paramètres des blocs de traitement utilisés à l'émission, et peut donc aisément retrouver le message qui a été transmis. Or, si l'on se place dans un contexte dit non-coopératif, le récepteur ne connaît pas les paramètres utilisés à l'émission : il devra retrouver par lui-même ces paramètres avec la seule connaissance des données reçues. Dans le cadre de cette thèse, nous nous placerons dans ce contexte et l'objectif sera d'estimer en aveugle les paramètres du bloc de codage canal qui a été utilisé à l'émission.

Le contexte non-coopératif est, de manière naturelle, celui du domaine militaire ou de la surveillance du spectre. Cependant, les travaux décrits dans ce document présentent également un intérêt dans le domaine civil, dans le contexte du développement récent de la radio cognitive et des récepteurs dits "intelligents". En effet, la popularité des communications sans fil et l'évolution des technologies radio engendrent une prolifération des normes ou standards permettant de transmettre l'information. De ce fait, il devient intéressant de concevoir des récepteurs auto-configurants et/ou intelligents, c'est à dire des récepteurs qui seraient capables d'identifier en aveugle les paramètres des blocs utilisés du côté émetteur avec la seule connaissance des données reçues.

Le fait de pouvoir identifier en aveugle les paramètres du bloc de codage canal avec pour seule connaissance le train binaire reçu du coté récepteur permettrait d'une part d'augmenter légèrement le débit utile puisqu'il devient inutile de transmettre les paramètres de l'émetteur, mais surtout, d'autre part, de rendre la chaîne de transmission beaucoup plus souple et évolutive en terme de changement de standard, puisque le récepteur s'adapte automatiquement à celui de l'émetteur par lui même sans intervention humaine de reconfiguration et sans nécessiter de changement de matériel.

Au delà du domaine des télécommunications, la reconnaissance aveugle de codes présente un intérêt dans des domaines tels que la recherche du codage caché dans l'ADN [Sic06] et la cryptographie à clés publiques. Dans ce dernier domaine, on peut noter que toute la cryptographie à clés publiques actuelle repose sur la complexité de la factorisation de grands nombres entiers en facteurs premiers. Une avancée mathématique significative dans le domaine des algorithmes de factorisation mettrait en péril toute la cryptographie à clés publiques, avec les conséquences que l'on peut imaginer dans le domaine de la sécurité des transactions bancaires, de la sécurité de l'identification/authentification sur Internet (protocole https), etc. Ce risque majeur a conduit certains chercheurs à envisager d'autres méthodes de cryptographie à clés publiques. Ainsi, du fait de la très grande difficulté à identifier, en aveugle, un code correcteur d'erreurs, McEliece a proposé dans [McE78] une méthode de cryptographie à clés publiques basée sur les codes correcteurs d'erreurs. Les travaux décrits dans le présent document pourront être utilisés pour mettre à l'épreuve la sécurité de telles méthodes, même s'il ne s'agit pas de l'objectif premier recherché.

Organisation du mémoire

Le document est organisé de la manière suivante :

Le chapitre 1

Ce chapitre présente tout d'abord le contexte de l'étude. Puis, une étude sur la théorie algébrique des codes linéaires est proposée. Notre objectif étant de reconnaître en aveugle un code convolutif, cette théorie permettra d'obtenir les propriétés indispensables pour le développement de nos méthodes de reconnaissance.

Le chapitre 2

Dans ce chapitre, nous nous intéressons à la reconnaissance aveugle de codes convolutifs. Un premier algorithme permettant l'identification de ces codes avec la seule connaissance d'une trame de mots de code non-entachée d'erreur est développé. Puis, une seconde méthode permettant l'identification d'un code convolutif lors d'une transmission bruitée est présentée. Enfin, une analyse de ce dernier algorithme est proposée.

Le chapitre 3

Dans ce chapitre, nous présentons tout d'abord une étude théorique sur une technique bien connue permettant d'augmenter le débit d'une transmission. Cette technique, nommée poinçonnage, consiste simplement à ne pas transmettre tous les bits des mots de code. A partir des propriétés théoriques mises en évidence, une méthode capable d'identifier en aveugle l'ensemble des paramètres d'un code convolutif poinçonné est développée lorsque le train binaire reçu a été codé et poinçonné. Cette méthode est développée dans le cas d'une transmission non-bruitée puis d'une transmission bruitée. Enfin, une analyse sur les performances d'identification de cet algorithme est proposée.

Le chapitre 4

Dans le but d'améliorer les performances de correction, de nouveaux schémas de codage ont été développés ces dernières années : il s'agit notamment des codes concaténés et des turbocodes. Dans ce chapitre, une étude visant à mettre en évidence les propriétés (utiles pour notre objectif) de ces nouveaux schémas de codage est présentée. D'après les différentes propriétés de ces schémas de codage, des méthodes d'identification sont proposées afin de reconnaître en aveugle ces schémas de codage lors d'une réception présentant une plage non-bruitée. L'extension au contexte fortement bruité fait partie des perspectives.

La conclusion et les perspectives clôtureront le document.

CHAPITRE 1

Contexte de l'étude et théorie algébrique des codes linéaires

1.1 Introduction

L'objectif de ce chapitre est d'introduire le principe d'une chaîne de transmission numérique et les généralités qui serviront à la bonne compréhension de ce manuscrit. Dans un premier temps, nous allons décrire le principe d'une chaîne de transmission et le principe d'une transmission non-coopérative. Une brève analyse des canaux de propagation les plus utilisés sera faite et enfin les principales propriétés algébriques des codes linéaires, en particuliers des codes convolutifs, seront exposées.

1.2 Chaîne de transmission numérique

Une chaîne de transmission numérique permet de transmettre une information numérique, soit une suite d'éléments binaires, d'une source à un ou plusieurs destinataire(s). Cette information est transmise via un support physique qui peut-être filaire, terrestre ou satellitaire. L'information numérique qui est transmise est, soit d'origine numérique comme dans les réseaux informatiques, soit d'origine analogique comme la parole, le son, etc, et convertie ensuite en un signal numérique. La conversion d'un signal analogique en un signal numérique est réalisée par un CAN (Convertisseur Analogique Numérique). Le schéma 1.1 représente le synoptique d'une chaîne de transmission numérique standard.

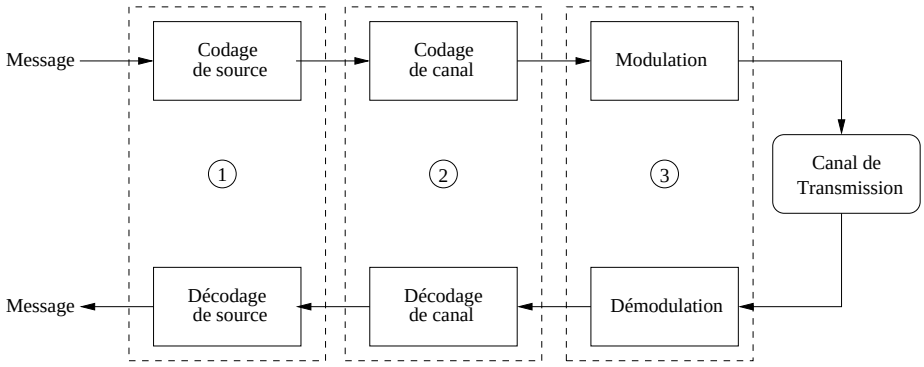


Figure 1.1 — Chaîne de transmission numérique

L'objectif d'une chaîne de transmission étant de transmettre une information d'une source à un destinataire avec le plus de fiabilité et le plus rapidement possible, l'émetteur réalise différentes

opérations. Ces différentes étapes sont modélisées sur le schéma 1.1 par différents blocs distincts ayant chacun un rôle bien spécifique. La première étape ①, nommée codage de source, consiste à augmenter l'efficacité de la transmission et à optimiser l'utilisation des ressources du système en compressant les données. L'étape suivante ②, appelée codage de canal, permet de lutter contre les erreurs introduites lors de la transmission du signal en ajoutant de la redondance au message. Un signal numérique n'existant pas d'un point de vue physique, il faut le transformer en un signal physique afin de pouvoir le transmettre, c'est le rôle du modulateur ③. Ce signal physique pourra ensuite être transmis via un canal de propagation. A la réception de ce signal, le récepteur effectue les opérations inverses afin d'obtenir le message source.

1.2.1 Le contexte non-coopératif

Le rôle du récepteur est d'obtenir le message émis en effectuant les opérations inverses de l'émetteur. Afin de pouvoir remonter jusqu'à ce message, il est évident que le récepteur doit avoir connaissance des opérations qui ont été effectuées par l'émetteur. Dans une communication classique, dite coopérative, le récepteur connaît la norme utilisée par l'émetteur, il peut aisément remonter jusqu'au message émis. Considérons maintenant le cas plus défavorable dit contexte non-coopératif. Dans cette configuration le récepteur n'aura aucune connaissance sur la norme utilisée par l'émetteur. Comment, dans ce contexte, le récepteur peut-il obtenir le message émis ? Cette problématique, issue dans un premier temps de la guerre électronique, a pour unique solution de concevoir des récepteurs capables de reconnaître en aveugle (i.e. avec la seule connaissance des données reçues en sortie du canal) les paramètres de l'émetteur. Ces dernières années, afin de répondre à un besoin réel (que ce soit dans le domaine militaire ou civil) de nombreux travaux de recherche ont porté sur la reconnaissance aveugle des paramètres de l'émetteur.

En effet, la popularité des communications sans-fil et l'évolution des technologies radio ont engendré ces dernières années une prolifération des normes et/ou des standards permettant de transmettre une information. Ces nouvelles normes pouvant entraîner des problèmes d'incompatibilité avec les anciennes, il devient nécessaire de concevoir des récepteurs auto-configurants. Il s'agit de récepteurs qui seraient capables d'estimer les paramètres de la norme utilisée par l'émetteur avec la seule connaissance des données reçues en sortie du canal. De tels récepteurs permettraient d'une part d'augmenter légèrement le débit utile puisqu'il deviendrait inutile de transmettre des paramètres de l'émetteur mais surtout de rendre les chaînes de transmission beaucoup plus souples et évolutives en terme de changement de standard. En effet, le récepteur s'adapterait automatiquement à celui de l'émetteur et cela sans intervention humaine de reconfiguration. De tels récepteurs ne sont bien sûr pas aisés à mettre en oeuvre. L'architecture d'un émetteur étant composée de plusieurs blocs (modulation, codage de canal, codage de source), l'idée est, dans un premier temps, de traiter le problème d'identification aveugle bloc par bloc. Dans ce cadre, on s'intéressera ici à la reconnaissance aveugle des paramètres du bloc de codage de canal.

Pour l'ensemble des utilisateurs des technologies de l'information, la sécurité des transmissions est et restera toujours une priorité. Malheureusement, il existera toujours des personnes malintentionnées qui essayeront d'intercepter et de décoder une communication. L'un des moyens pour protéger une communication est d'utiliser la cryptographie. La cryptographie permet d'assurer la confidentialité, l'intégrité et l'authenticité d'un message émis. Le principe d'un cryptosystème est de chiffrer (ou coder) un message avec une clé afin de rendre ce message incompréhensible pour quiconque n'étant pas doté de la clé de chiffrement. Comme le montre la figure 1.2, le chiffrement du message intervient en général après la compression des données.

Si la cryptographie représente l'art de chiffrer une information, la cryptanalyse représente l'art de casser ce chiffrement. En se mettant directement à la place d'un attaquent potentiel (i.e une personne interceptant une communication), la cryptanalyse offre de nombreuses méthodes permettant de tester la sécurité d'une transmission. En ce plaçant dans le contexte le plus défavorable, l'ob-

Figure 1.2 — Codage et chiffrement

servateur n'a aucune connaissance des paramètres de l'émetteur. Avant d'obtenir un train binaire cryptanalysable, il devra posséder des outils lui permettant dans un premier temps de reconnaître la modulation qui a été utilisée, puis les paramètres du bloc de codage de canal. De nombreux travaux ont été développés afin d'identifier en aveugle la modulation qui a été utilisée à l'émission, le lecteur intéressé pourra se référer à [AN96] et [DBNS05] [1]. En faisant l'hypothèse que l'observateur arrive à remonter au train binaire obtenu en sortie du démodulateur, il devra reconnaître les paramètres du bloc de codage de canal pour obtenir un train binaire cryptanalysable. De plus, certains systèmes de cryptographie étant apparentés à la théorie des codes correcteurs d'erreurs, les algorithmes de reconnaissance de code pourraient également permettre de tester la sécurité de ces systèmes de cryptographie.

Nous pouvons voir l'intérêt de mettre en place des systèmes capables d'identifier en aveugle les paramètres du bloc de codage de canal. Dans la suite de ce manuscrit, nous ferons l'hypothèse que nous avons accès au train binaire issu du démodulateur. Ce train binaire ayant été entaché d'erreurs par le canal de propagation, nous allons tout d'abord présenter les modèles de canaux les plus couramment utilisés.

1.2.2 Le canal de transmission

Le canal de transmission, appelé également canal de propagation, représente le support physique qui permet de transmettre le signal. La forme physique du signal sera adaptée au milieu, par exemple : une onde électromagnétique pour le milieu ambiant, un signal électrique pour un câble, une onde lumineuse pour une fibre optique, etc. Ces milieux provoquent certaines modifications du signal que l'on y transmet, telles que des atténuations, des retards, des interférences, des distorsions, etc. Afin de modéliser au mieux ces différents phénomènes, de nombreux travaux de recherche ont porté sur la modélisation des canaux de propagation. Les canaux de transmissions peuvent être classés en deux groupes, les canaux stationnaires dont les paramètres sont fixes au cours du temps (fibres optiques, câbles métalliques, etc.) et les canaux non stationnaires dont les paramètres évoluent au cours du temps (communications radio-mobiles).

1.2.2.1 Le canal stationnaire

Les canaux stationnaires sont en règle générale modélisés par un canal appelé BBAG (pour Bruit Blanc Additif Gaussien). Ce canal, représenté sur la figure 1.3, est l'un des plus utilisés pour la simulation de transmissions numériques car il permet de représenter assez fidèlement les phénomènes physiques qui interviennent dans les canaux de transmission radioélectrique tout en gardant une simplicité de mise en oeuvre et d'analyse.

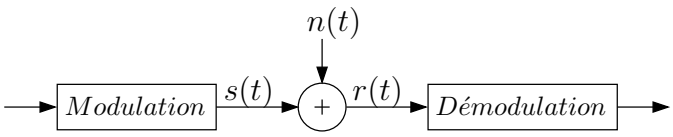


Figure 1.3 — Modélisation du canal BBAG

A chaque instant t, il modifie le symbole émis, $s(t)$, par l'addition d'un bruit blanc gaussien,

1. dans la plupart des travaux réalisé, l'hypothèse que le train binaire est iid est faite

$n(t)$:

$$r(t) = s(t) + n(t) \tag{1.1}$$

Le BBAG, $n(t)$, est caractérisé par un processus aléatoire gaussien de moyenne nulle, de variance σ_n^2 et de densité spectrale de puissance $\frac{N_0}{2}$.

1.2.2.2 Le canal non stationnaire

Lors d'une transmission radio-mobile, les mouvements de l'émetteur et du récepteur engendrent des fluctuations du canal. Ces fluctuations du canal vont entraîner des erreurs de transmission qui arriveront par rafales.

De plus, dans ce type de canaux, l'environnement est souvent riche en échos ce qui se caractérise par de nombreux trajets multiples. La modélisation de ces types de canaux ne peut plus se faire avec un simple BBAG : il faut en effet utiliser des modèles plus élaborés qui tiendront compte des différences de propagation du milieu, appelés encore atténuations ou évanouissements. La sortie d'un canal à évanouissement comportant N trajets multiples s'exprime alors par :

$$r(t) = \sum_{n=0}^{N-1} \alpha_n.s(t - \tau_n) + n(t) \tag{1.2}$$

où $n(t)$ représente le BBAG, τ_n le retard affectant chaque trajet et α_n l'atténuation complexe. L'atténuation α_n est une variable aléatoire gaussienne qui suit une loi uniforme pour la phase et une loi de Rayleigh ou de Rice pour le module.

1.2.2.3 Le canal binaire

En sortie du canal de transmission, le signal bruité est démodulé afin d'obtenir une suite d'éléments binaires. Le taux d'erreur binaire, appelé couramment TEB ou BER en anglais (pour Bit Error Rate) va permettre d'évaluer l'impact du bruit du canal sur le signal en comparant le message en entrée du modulateur et celui obtenu après démodulation. Il est possible de représenter l'ensemble des parties *modulation*, *canal de transmission* et *démodulation* par un canal binaire. Ce modèle de canal qui sera à entrée et à sortie ferme (c-à-d des suites de "0" et de "1") permettra d'analyser les performances du système (*modulation*, *canal de transmission* et *démodulation*). Le principe d'un canal binaire, représenté sur la figure 1.4, est d'associer à chaque bit d'entrée une certaine probabilité pour que le bit reçu soit erroné. Cette probabilité, qui représente le TEB, dépendra d'une part des erreurs engendrées par le canal de propagation et d'autre part des erreurs dû à la démodulation du signal

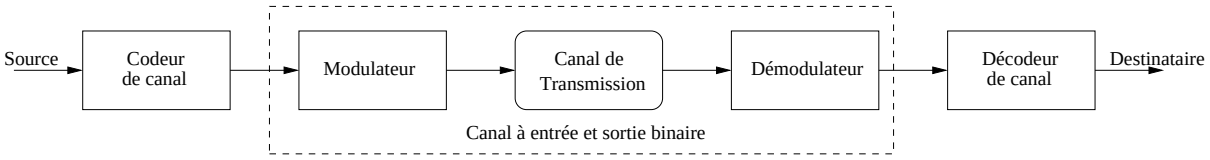


Figure 1.4 — Description d'un canal binaire

Le canal binaire symétrique

Le modèle le plus simple est le canal binaire symétrique appelé BSC (pour "Binary Symmetric Channel"). Un BSC est défini par sa probabilité d'erreur, notée p. La valeur de cette probabilité qui dépend du canal et de la modulation correspond au TEB obtenu en sortie du démodulateur.

Si l'on note c et y les éléments en entrée et en sortie du BSC, alors la probabilité pour que le

symbole reçu soit erroné sera égale à p et inversement la probabilité pour que le symbole reçu soit correcte sera de $1-p$:

$$\begin{aligned} Pr\left(y=0|c=1\right) &= Pr\left(y=1|c=0\right) = p \\ Pr\left(y=0|c=0\right) &= Pr\left(y=1|c=1\right) = 1-p \end{aligned} \tag{1.3}$$

La figure 1.5 représente le fonctionnement d'un BSC. Chaque élément binaire en sortie du canal dépend uniquement de l'élément binaire d'entrée, on parle de canal sans mémoire.

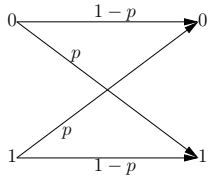


Figure 1.5 — Canal binaire symétrique

Le canal burst

Pour certains canaux, tels que les canaux satellitaires, les conditions de propagation (fréquence de transmission, environnement,...) évoluent au cours de la transmission. Dans ce cas, il faut considérer un modèle de canal plus fin afin de représenter les erreurs qui arrivent par paquets. Le modèle de Gilbert-Elliot [Gil60] permet de caractériser ces canaux appelés de type *burst*. Ce modèle de canal, représenté sur la figure 1.6 comprend deux états : l'état *bon* noté G et l'état *mauvais* noté B. L'état G correspond à une plage où les données seront très peu bruitées alors que l'état B représente une plage où les erreurs arriveront par rafale.

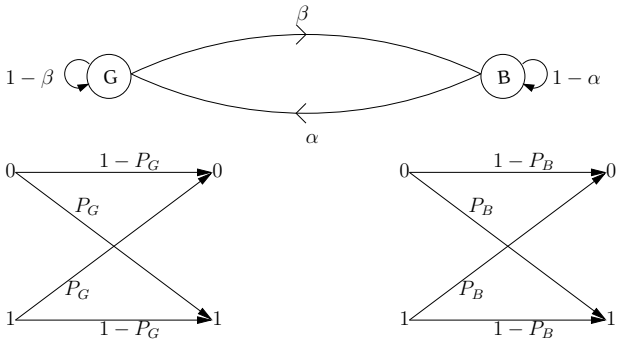


Figure 1.6 — Modèle du canal de Gilbert-Elliot

Les probabilités de transitions $Pr(B|G)$ et $Pr(G|B)$ sont respectivement α et β. La probabilité d'erreur dans l'état B est P_B et celle dans l'état G est P_G. Le TEB moyen, noté p, est donné par :

$$p = \frac{1}{\alpha+\beta}\left[\alpha.P_B + \beta.P_G\right] \tag{1.4}$$

1.2.3 Le codage de canal

Le codage de canal joue un rôle fondamental dans les systèmes de communication puisqu'il permet de protéger les données des perturbations introduites par le canal de transmission. L'une des questions fondamentales est de savoir la quantité d'information qu'il est possible de transmettre de façon fiable. En 1948, C. Shannon [Sha48] démontre que pour une source d'information de débit D_s (bit/s) qui transmet sur un canal de capacité C_c (bit/s), il existe, si $D_s < C_c$, un code garantissant une transmission quasi parfaite. Ce théorème ne dit pas comment construire de tels codes, mais simplement qu'il est possible en optimisant le codage d'obtenir après le décodage une information

avec très peu d'erreurs. Dès lors, de nombreux chercheurs se sont penchés sur la construction de ces codes.

Le bloc de codage de canal est en général équipé de un ou plusieurs code(s) correcteur(s) d'erreurs et d'un entrelaceur afin de lutter efficacement contre les erreurs de transmission. Le rôle du codeur est d'introduire à l'émission de la redondance dans le train binaire informatif ; cette redondance permettant, du coté récepteur, de corriger et/ou de détecter d'éventuelles erreurs. Les bruits perturbateurs présents en pratique lors de transmissions radio-mobiles engendrent généralement des erreurs de type *burst* ou encore par paquets, c'est-à-dire que les perturbations affectant les bits d'information se produisent en rafales. Dans cette situation, la présence d'un entrelaceur permet de limiter les effets de ce type d'erreurs en dispersant à l'émission les bits de chaque mot d'information à l'intérieur d'une trame de taille conséquente, très supérieure à celles des mots en question. Grâce à cela, les bits consécutifs concernés par la rafale d'erreurs n'appartiennent pas aux même mots informatifs et peuvent être alors plus facilement et surtout efficacement corrigés après désentrelacement. La succession des deux traitements liés à l'opération d'entrelacement à l'émetteur et de désentrelacement au récepteur peut être vue pour simplifier comme une dispersion des erreurs tout au long de la trame, au lieu quelles soient regroupées sur un ou plusieurs mots informatifs empêchant toutes tentatives de correction de celles-ci par le décodeur aux capacités de correction malgré tout limitées.

Dans la littérature, on distingue deux grandes familles de codes correcteurs d'erreurs, les codes en blocs et les codes convolutifs. Dans le cas des codes en blocs, l'information est découpée en blocs de k bits. Chaque bloc est transformé en un nouveau bloc de n bits ($n > k$), qui forme un mot de code. Dans le cas d'un code convolutif, chaque bloc de n bits en sortie du codeur dépend non seulement des k bits présents en entrée du codeur mais également d'autres blocs de k bits ayant été introduits précédemment.

De même que les codeurs, il existe également deux grandes familles d'entrelaceurs, les entrelaceurs blocs et convolutifs. Un entrelaceur par bloc est un dispositif qui à partir d'un bloc de symboles reçu en entrée, va restituer en sortie un bloc contenant les mêmes symboles mais dans un ordre différent. Un entrelaceur convolutif (ou multiplexé) de période S est construit en multiplexant la séquence d'entrée de façon séquentielle en S sous-séquences, introduisant un retard pour chaque sous-séquence et en démultiplexant ces S sous-séquences avec prise en compte de ces retards.

De par leurs performances en terme de correction des erreurs et le fait qu'ils traitent l'information en flux, les codes convolutifs sont très prisés dans les communications radio-mobiles. En effet, que ce soit dans les standards de seconde génération (GSM [3GP05a] et CDMA-2000 [3GP09]), de troisième génération (UMTS [3GP05b] et WDMA) ou encore dans les futurs standards de quatrième génération (LTE [3GP08]), on retrouve toujours l'utilisation de un ou plusieurs code(s) convolutif(s). De ce fait, nos travaux se sont portés sur la reconnaissance en aveugle des codes convolutifs. L'identification aveugle des paramètres d'un code convolutif étant fortement basée sur les propriétés algébriques de ces codes, nous proposons dans ce premier chapitre une étude sur les propriétés des codes linéaires.

1.3 Les codes en blocs

Bien qu'il existe de nombreuses familles de codes en blocs, telles que les codes de Hamming [Ham50], les codes BCH [BRC60] [Hoc59], les codes Reed-Solomon [RS60], les codes LDPC [Gal63], etc., le principe de codage reste le même : le regroupement d'un message d'entrée en blocs afin d'ajouter à chacun des messages une quantité contrôlée de redondance. Cette redondance introduite permettra à la réception de détecter et/ou de corriger d'éventuelles erreurs de transmission.

1.3.1 Le codage

Notons C un (n, k) code en bloc linéaire de rendement $r = k/n$, dans le corps des symboles F. Dans le reste de notre étude, nous prendrons le corps de Galois à deux éléments ("0" et "1"), noté $GF(2)$. La matrice génératrice d'un code en bloc, notée G, est une matrice de taille $k \times n$ et de rang k qui permet d'engendrer le code C. Le nombre de sorties d'un codeur, n, étant strictement supérieur au nombre d'entrées, k, le codeur introduira systématiquement de la redondance au message. Nous appellerons un *mot d'information*, un vecteur de k bits d'information introduit simultanément en entrée du codeur et un *mot de code*, le vecteur de n bits qui en résulte. Notons $\mathbf{m}(t) = (m_1(t), \cdots, m_k(t))$, le mot d'information transmis à l'instant t et respectivement $\mathbf{c}(t) = (c_1(t), \cdots, c_n(t))$, le mot de code reçu à l'instant t de tel sorte que :

$$\mathbf{c}(t) = \mathbf{m}(t).G \quad \text{pour } t = 0, 1, \cdots . \tag{1.5}$$

Lors d'une transmission numérique, l'information qui est transmise ne correspond pas uniquement à un mot d'information mais à une séquence de mots d'information. En notant L la longueur de la séquence à coder, la séquence d'entrée notée $\mathbf{m}$ et la séquence de sortie notée $\mathbf{c}$ sont telles que :

$$\begin{array}{l} \mathbf{m} = (\mathbf{m}(0), \cdots, \mathbf{m}(L-1)) = (m_1(0), \cdots, m_k(0), \cdots, m_1(L-1), \cdots, m_k(L-1)) \\ \mathbf{c} = (\mathbf{c}(0), \cdots, \mathbf{c}(L-1)) = (c_1(0), \cdots, c_n(0), \cdots, c_1(L-1), \cdots, c_k(L-1)) \end{array} \tag{1.6}$$

La matrice de codage liant ces L mots de code aux L mots d'information est une matrice bloc-diagonale constituée de L matrices G. Cette matrice de codage étant composée de nombreux "0", ces éléments ont été omis afin de simplifier la notation. L'équation de codage (1.5) devient :

$$\mathbf{c} = \mathbf{m}.\begin{pmatrix} G & & & \\ & G & & \\ & & \ddots & \\ & & & G \end{pmatrix} \tag{1.7}$$

Cette matrice de codage est une matrice de taille $k.L \times n.L$ et chaque mot de code dépend uniquement du mot d'information présent en entrée du codeur.

Pour un même rendement, il existe un ensemble de matrices génératrices pouvant générer un même code.

Définition 1.1. *Deux codes sont équivalents si ils ont même distribution de distances entre mots. Mathématiquement, deux matrices génératrices G et G' sont équivalente si il existe une matrice inversible quelconque A et une matrice de permutation B vérifiant $G = A.G'.B$. matrice de permutation.*

La notion de codeur *équivalent* signifie que les séquences codées par deux codeurs équivalents appartiendront à la même famille ou au même code. Parmi l'ensemble des matrices génératrices d'un code, il existera toujours une forme systématique équivalente de la matrice génératrice.

Définition 1.2. *Un code $C(n, k)$ est un code systématique si tous les mots de code contiennent les k bits d'information non modifiés et les $n - k$ bits restant sont appelés bits de parité.*

D'après la définition 1.2, la matrice génératrice d'un code systématique peut être de la forme $G = (\mathbf{I}_k \mid P)$ où $G = (P \mid \mathbf{I}_k)$, avec $\mathbf{I}_k$ la matrice identité de taille k et P une $(k \times n - k)$ matrice.

Exemple 1.1.

Prenons l'exemple d'un code de Hamming de paramètres $n = 7$, $k = 4$ et de matrice génératrice :

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

Soit $\mathbf{m}(t) = \begin{pmatrix} m_1(t) & m_2(t) & m_3(t) & m_4(t) \end{pmatrix}$, un mot d'information composé de 4 bits et $\mathbf{c}(t) = \begin{pmatrix} c_1(t) & c_2(t) & c_3(t) & c_4(t) & c_5(t) & c_6(t) & c_7(t) \end{pmatrix}$ le mot de code de 7 bits qui en résulte.

D'après l'équation de codage (1.5), nous pouvons écrire que :

$$\begin{cases} c_1(t) = m_1(t) \\ c_2(t) = m_2(t) \\ c_3(t) = m_3(t) \\ c_4(t) = m_4(t) \\ c_5(t) = m_2(t) + m_3(t) + m_4(t) \\ c_6(t) = m_1(t) + m_3(t) + m_4(t) \\ c_7(t) = m_1(t) + m_2(t) + m_4(t) \end{cases}$$

La matrice génératrice de ce code étant systématique, un mot de code est un vecteur composé des 4 bits du mot d'information ($m_1(t)$, $m_2(t)$, $m_3(t)$, $m_4(t)$) et de 3 bits de parité ($c_5(t)$, $c_6(t)$, $c_7(t)$). En prenant le mot d'information $\mathbf{m}(t) = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}$, le mot de code sera :

$$\mathbf{c}(t) = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}.G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

En permutant l'ordre des lignes de la matrice génératrice G, il est possible d'obtenir une matrice génératrice équivalente G' :

$$G' = L.G = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Un code $C(n, k)$ permet d'engendrer 2^k mots de code de n bits. Avec notre exemple, nous obtenons une famille de 16 mots de code de 7 bits. La figure 1.7 représente les 16 mots d'information et les 16 mots de code obtenus avec la matrice génératrice G et ceux obtenus avec G'. Sur cette figure, les bits à "0" sont représentés par les cases blanches et ceux à "1" par les cases noires.

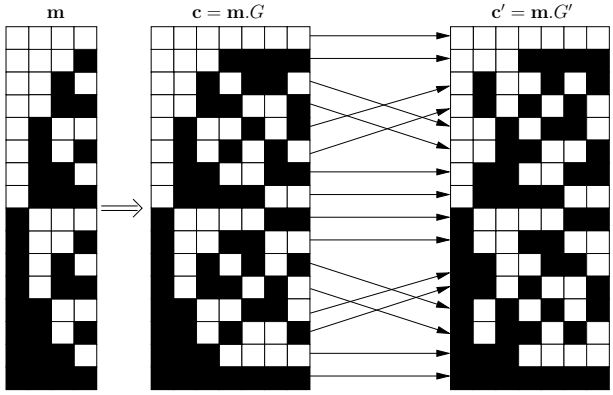


Figure 1.7 — Mots de code

Tout d'abord, nous remarquons qu'un mot d'information codé par G et G' ne donne pas néces-

sairement le même mot de code. En revanche, en regardant tous les mots de code obtenus avec la matrice G, on remarque que l'on obtient les mêmes avec la matrice G' mais dans un ordre différent. Ces deux codes sont équivalent puisque leurs mots de code appartiennent à la même famille.

1.3.2 Les codes duaux

Pour le décodage ou simplement la détection d'une erreur dans un mot de code, les *codes duaux* vont jouer un rôle très important. On définit le code dual de C, noté $C^{\perp}$, comme étant le code orthogonal à C.

Définition 1.3. *L'espace dual d'un (n, k) code C de dimension k est le $(n, n-k)$ code dual de C.*

On note H, la matrice génératrice du code dual $C^{\perp}$ appelée également *matrice de parité* ou *matrice de contrôle* du code C.

Définition 1.4. *Soit G une matrice génératrice du code C. Une matrice H de rang $n-k$ et de taille $(n-k) \times n$ est une matrice de parité pour C si et seulement si :*

$$G.H^T = 0 \tag{1.8}$$

où ^{T} désigne la transposition.

De cette définition découle le corollaire suivant :

Corollaire 1.1. *Soit H une matrice de parité du code C. Un mot $\mathbf{c}$ appartient au code C si et seulement si :*

$$\mathbf{c}.H^T = 0 \tag{1.9}$$

Nous avons montré que la matrice génératrice d'un code C n'était pas unique. En effet, il existe plusieurs matrices G équivalentes qui engendrent le même code C. De ce fait, la matrice de parité d'un code n'est pas unique. Et comme pour la matrice génératrice d'un code, la matrice de parité pourra également se mettre sous la forme particulière d'une matrice systématique.

Exemple 1.2.

Suite de l'exemple 1.1.
La matrice de parité du code précédent peut s'écrire :

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

ou encore sous sa forme systématique en permutant l'ordre des colonnes :

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

Nous pouvons vérifier que ces deux matrices de parité vérifient la définition 1.4 ainsi que le corollaire 1.1.

Après le codage des séquences d'entrée, les mots de code sont transmis via un canal de transmission. Cette transmission va engendrer d'éventuelles erreurs sur la séquence reçue. Nous noterons $\mathbf{y}$, la séquence des mots reçue après transmission. Les erreurs de transmission impliquent que certains bits de la séquence reçue seront différents de la séquence des mots de code. Nous appellerons syndrome, un vecteur de k bits noté $\mathbf{s}$ et défini par :

$$\mathbf{s} = \mathbf{y}.H^T \tag{1.10}$$

Si le mot reçu $\mathbf{y}$ est un mot de code alors le syndrome sera nul. En revanche, si le syndrome est non nul, cela impliquera que le mot $\mathbf{y}$ est erroné. Il peut également arriver que le syndrome soit nul alors que le mot reçu $\mathbf{y}$ est erroné. Nous verrons ci-dessous que chaque code a une capacité de détection/correction limitée et que si cette capacité est atteinte alors le code ne sera plus capable de détecter et/ou corriger à coup sûr ces erreurs ; de ce fait il peut arriver que le syndrome soit nul en présence d'erreurs.

Exemple 1.3.

Suite de l'exemple 1.1.

Le mot de code envoyé est $\mathbf{c} = \begin{pmatrix}1 & 0 & 1 & 1 & 0 & 1 & 0\end{pmatrix}$ et le mot reçu après sa transmission dans le canal est $\mathbf{y} = \begin{pmatrix}1 & 0 & 1 & 1 & 1 & 1 & 0\end{pmatrix}$. En prenant la matrice de parité sous sa forme systématique, d'après l'équation (1.10) le syndrome $\mathbf{s}$ est :

$$\mathbf{s} = \mathbf{y}.H^T = \begin{pmatrix}1 & 0 & 1 & 1 & 1 & 1 & 0\end{pmatrix}. \begin{pmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\\ 1 & 1 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\end{pmatrix}$$

Le syndrome est un vecteur non nul ce qui implique qu'il y a au moins une erreur dans le mot reçu.

1.3.3 La capacité de correction

L'objectif d'un code en bloc étant de permettre à la réception, de détecter et/ou de corriger d'éventuelles erreurs étant intervenues lors de la transmission, il est nécessaire de pouvoir évaluer la capacité de correction d'un code. Nous noterons $w(\mathbf{c})$ le poids de Hamming du mot de code $\mathbf{c}$ qui représente le nombre de bits à "1" qu'il contient. La distance de Hamming, notée d_H, entre deux mots de code est définie par le nombre de positions binaires qui diffèrent entre ces deux mots de code.

Notons $\mathbf{c}$ et $\mathbf{c}'$ deux mots de code de dimension n. Alors, le mot de code, noté $\mathbf{y}$ résultant de la somme des deux mots est tel que :

$$\mathbf{y} = \mathbf{c} + \mathbf{c}' = (y(0), \cdots, y(n-1)) \tag{1.11}$$

où les bits $y(i)$ sont obtenus par une addition binaire des éléments $c(i)$ et $c'(i)$:

$$y(i) = c(i) + c'(i), \quad \forall i = 0, \cdots, n-1 \tag{1.12}$$

La distance de Hamming entre les mots $\mathbf{c}$ et $\mathbf{c}'$ est égale au nombre d'éléments à "1" du vecteur $\mathbf{y}$.

Exemple 1.4.

Soient deux mots de code $\mathbf{c}_1 = \begin{pmatrix}1 & 1 & 0 & 1 & 0 & 1 & 1 & 0\end{pmatrix}$ et $\mathbf{c}_2 = \begin{pmatrix}1 & 0 & 1 & 1 & 0 & 0 & 1 & 0\end{pmatrix}$. Les

mots de code $\mathbf{c}_1$ et $\mathbf{c}_2$ ont pour poids de Hamming respectifs 5 et 4. La somme des deux mots de code $\mathbf{c}_1 + \mathbf{c}_2 = \begin{pmatrix}0 & 1 & 1 & 0 & 0 & 1 & 0 & 0\end{pmatrix}$ a pour poids de Hamming 3, ce qui donne une distance de Hamming entre ces deux mots de code de 3.

La distance minimale d_{min} d'un code en bloc correspond à la plus petite distance de Hamming entre ses mots de code :

$$d_{min} = \min_{\mathbf{c} \neq \mathbf{c}'} \left(\mathbf{c}, \mathbf{c}'\right) \qquad \forall \mathbf{c}, \mathbf{c}' \in C(n,k) \tag{1.13}$$

La distance entre deux mots de code étant égale au poids de leur somme, la distance minimale d'un code en bloc correspond également au poids minimal de ces mots de code non nuls.

$$d_{min} = \min_{\mathbf{c} \neq 0} w\left(\mathbf{c}\right) \qquad \forall \mathbf{c} \in C(n,k) \tag{1.14}$$

Un code en bloc de paramètre d_{min} est capable de détecter $(d_{min} - 1)$ bits erronés et d'en corriger $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$.

Exemple 1.5.

Suite de l'exemple 1.1.

Nous avons reporté sur la figure 1.8, en plus de la liste des 16 mots de code précédents, leurs poids de Hamming.

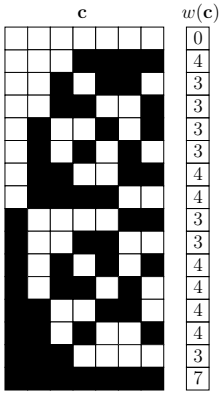


Figure 1.8 — Poids de Hamming des mots de code

Nous pouvons voir que le poids minimal de ces mots de code non nuls est de 3. D'après l'équation (1.14), la distance minimale de ce code est : $d_{min} = 3$. Ce code sera capable de détecter 2 bits erronés et d'en corriger 1 seul.

1.4 Les codes convolutifs

1.4.1 Présentation des codes convolutifs

Les codes convolutifs ont été introduit par Elias [Eli55] en 1955, mais sans algorithme de décodage rapide et efficace ce type de code a été laissé de côté. Le premier algorithme capable de décoder de tels codes, appelé *décodage séquentiel* a été introduit par Wozencraft [Woz57] en 1957 puis amélioré par Fano [Fan63] en 1963. Il faudra encore attendre quatre ans pour qu'un algorithme de décodage particulièrement intéressant apparaisse, le décodage de Viterbi [Vit67]. Dès lors, de nombreux travaux ont porté sur la construction des codeurs convolutifs. L'enjeu de ces travaux étant d'obtenir des codes au pouvoir de correction le plus élevé possible. En 1970, puis en 1973,

Forney [For70] et [For73] démontre les propriétés algébriques des *bons* codes convolutifs, soit des codes ayant de forts pouvoir de correction. En parallèle avec ces avancées sur les propriétés des bons codes, un nouvel algorithme de décodage est apparu. Cet algorithme inventé par Bahl, Cocke, Jelinek et Raviv [BCJR74] en 1974 est référencé dans la littérature sous trois dénominations : BCJR (initiales des inventeurs), MAP (Maximum A Posteriori) ou APP (A Posteriori Probability).

Tout comme les codes en blocs, le principe d'un code convolutif est d'ajouter de la redondance au message émis. Un mot d'information est également un vecteur de k bits et un mot de code un vecteur de n bits, mais nous allons introduire un nouveau paramètre, noté K, qui correspond à la longueur de contrainte du codeur. Chaque mot de code dépend du mot d'information présent en entrée du codeur, comme dans le cas d'un code en bloc, mais également des $(K-1)$ mots d'information ayant été introduits précédemment. De ce fait, un code convolutif peut être représenté, comme sur la figure 1.9, par un ensemble de registres à décalage, tel que chaque sortie du codeur est une combinaison linéaire des $k \times K$ cellules du registre à décalage. La longueur de contrainte d'un codeur représente le nombre de blocs de k bits présents à l'intérieur du registre à décalage. Intuitivement, on peut dès à présent voir que de par la mémoire d'un code convolutif, ces codes engendrent des séquences codées de longueurs infinies, qui ne peuvent pas être traitées par paquets de bits disjoints comme dans le cas des codes en blocs.

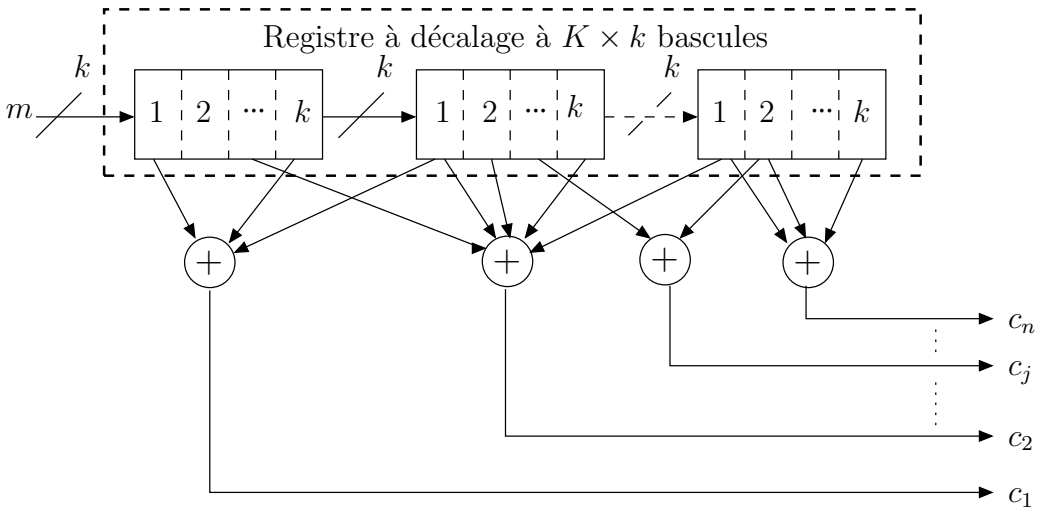


Figure 1.9 — Schéma d'implémentation d'un code convolutif à l'aide de registre à décalage

De même que les codes en blocs, il existe différentes familles de codes convolutifs. Ces différentes familles peuvent-être distinguées par deux termes : *systématique* et *récursif*.

- Le terme *systématique* signifie que l'on retrouve les k bits d'entrée en sortie du codeur.
- Le terme *récursif* signifie que les sorties du codeur peuvent dépendre des sorties précédentes. Dans cette configuration, le codeur possède des registres avec rétroaction (rebouclage de la sortie en entrée).

Dans le cadre de notre étude, nous nous sommes intéressé aux cas des codeurs de types Non Récursifs et Non Systématiques, noté NRNSC, et des codeurs de types Récursifs et Systématiques, noté RSC. Ces deux types de codes étant les plus répandus dans les standards radio-mobiles.

Un code convolutif peut être entièrement défini par ses 3 paramètres entiers (n, k et K) et par sa matrice génératrice. La matrice génératrice d'un code convolutif est composée de $k \times n$ polynômes générateurs qui sont soit de simples polynômes dans le cas d'un codeur de type NRNSC, soit des fractions rationnelles polynomiales dans le cas d'un RSC. Ces polynômes générateurs pourront également être représentés sous forme vectorielle ou encore en octal.

1.4.2 Les polynômes générateurs

Notons $a(D)$ un polynôme d'indéterminée en D de degré L défini par :

$$a(D) = \sum_{t \geq 0}^{L} a(t).D^t = a(0) + a(1).D + \cdots + a(L).D^L \tag{1.15}$$

où $a(t)$ est le t-ième coefficient du polynôme. Alors, ce polynôme pourra également être décrit par un vecteur, noté $\mathbf{a}$, qui sera composé des $(L+1)$ coefficients du polynôme qui dans le cas présent sont des coefficients à valeur dans $GF(2)$, donc des éléments binaires :

$$\mathbf{a} = \begin{pmatrix} a(0) & a(1) & a(2) & \cdots & a(L) \end{pmatrix} \tag{1.16}$$

Ce vecteur $\mathbf{a}$ pourra ensuite être converti en octal en regroupant les bits par paquets de trois.

Exemple 1.6.

Conversion du nombre octal (542) en binaire :

$$\begin{cases} 5 \Rightarrow \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \\ 4 \Rightarrow \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \\ 2 \Rightarrow \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \end{cases}$$

Le nombre (542) en octal s'écrira $\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$ en binaire.

1.4.3 Les codeurs de forme NRNSC

La matrice génératrice d'un code convolutif peut être représentée sous différentes formes, nous allons dans un premier temps introduire une représentation sous forme matricielle qui nous permettra dans un second temps d'introduire leur représentation sous forme polynomiale qui est plus couramment utilisée.

1.4.3.1 Représentation matricielle

Nous noterons $\mathbf{g}_{i,j}$ $(\forall i = 1, \cdots, k$ et $\forall j = 1, \cdots, n)$, les vecteurs composés des coefficients des polynômes générateurs. Ces vecteurs sont composés de K éléments binaires, tels que :

$$\mathbf{g}_{i,j} = (g_{i,j}(0), g_{i,j}(1), \cdots, g_{i,j}(K-1)) \tag{1.17}$$

Alors, nous noterons G la matrice génératrice d'un code convolutif constituée de ces $k \times n$ vecteurs :

$$G = \begin{pmatrix} \mathbf{g}_{1,1} & \cdots & \mathbf{g}_{1,n} \\ \vdots & \cdots & \vdots \\ \mathbf{g}_{k,1} & \cdots & \mathbf{g}_{k,n} \end{pmatrix} \tag{1.18}$$

Nous noterons $\mathbf{m}$ la séquence d'entrée, telle que :

$$\mathbf{m} = (\mathbf{m}(0), \mathbf{m}(1), \cdots) \tag{1.19}$$

avec $\mathbf{m}(t) = \begin{pmatrix} m_1(t) & m_2(t) & \cdots & m_k(t) \end{pmatrix}$ qui représente le mot d'information à l'instant t.
La séquence de sortie du code, $\mathbf{c}$, sera telle que :

$$\mathbf{c} = (\mathbf{c}(0), \mathbf{c}(1), \cdots) \tag{1.20}$$

avec $\mathbf{c}(t) = \begin{pmatrix} c_1(t) & c_2(t) & \cdots & c_n(t) \end{pmatrix}$, le mot de code généré à l'instant t.

L'expression du bit de la j-ème sortie à l'instant t en fonction des bits d'entrée successifs s'exprime de la manière suivante :

$$c_j(t) = \sum_{l=0}^{K-1} \sum_{i=1}^{k} m_i(t-l).g_{i,j}(l) \qquad \forall j = 1, \cdots, n \tag{1.21}$$

En définissant K sous-matrices de codage F_l, $\forall l = 0, \cdots, K-1$, comme ci-dessous :

$$F_l = \begin{pmatrix} g_{1,1}(l) & \cdots & g_{1,n}(l) \\ \vdots & \cdots & \vdots \\ g_{k,1}(l) & \cdots & g_{k,n}(l) \end{pmatrix}, \tag{1.22}$$

l'expression liant le mot de code reçu à l'instant t aux K mots d'information présent dans le registre est :

$$\mathbf{c}(t) = \mathbf{m}(t).F_0 + \mathbf{m}(t-1).F_1 + \cdots + \mathbf{m}(t-(K-1)).F_{K-1} = \sum_{l=0}^{K-1} \mathbf{m}(t-l).F_l \tag{1.23}$$

De par la mémoire introduite par le code, un code convolutif est adapté pour coder des séquences de taille infinie. Il sera nécessaire de réaliser une troncature afin de traiter des séquences de taille finie. Par convention, le premier mot d'information à entrer dans le codeur est noté $\mathbf{m}(0)$ et le premier mot de code est noté $\mathbf{c}(0)$. L'expression $\mathbf{m}(t-l)$, $\forall l = 1, \cdots, K-1$, n'existe que pour $(t-l) \geq 0$. Afin de coder les $K-1$ premiers mots de code, les registres à décalage devront être initialisés à "0". Cette initialisation, qui est également appelée "remplissage" des registres, consiste à considérer que les $(K-1)$ premiers mots d'information $\mathbf{m}(-1), \cdots, \mathbf{m}(-(K-1))$ sont égaux à 0. Pour une séquence d'entrée composée de L mots d'information, $\mathbf{m}(t-l)$ n'est défini que pour $0 \leq (t-l) < L$. Tant que le dernier mot d'information, $\mathbf{m}(L-1)$ est présent dans le registre, le codage est en cours. Or, entre l'instant où $\mathbf{m}(L-1)$ entre dans le codeur et celui où il en ressort, il est nécessaire de placer des bits supplémentaires après le message. Par convention, il sera rajouté $k.(K-1)$ bits à zéros à la fin de chaque trame. Par conséquence, pour une séquence d'entrée de L mots d'information on obtiendra une séquence de sortie composée de $L + (K-1)$ mots de code. En général $L >> K-1$, de ce fait les $K-1$ derniers mots de code qui correspondent à la partie "vidange" des registres sont négligés.

Exemple 1.7.

Prenons l'exemple d'un codeur de rendement 1/2 ($k = 1$ et $n = 2$) et de longueur de contrainte égale à 3, $K = 3$. La matrice génératrice de ce codeur est composée de 2 polynômes générateurs :

$$G = \begin{pmatrix} 5 & 7 \end{pmatrix}_{\text{octal}} = \begin{pmatrix} (1\,0\,1) & (1\,1\,1) \end{pmatrix}_{\text{binaire}}$$

Les K sous-matrices de codage F_l, définies à l'équation (1.22), sont :

$$F_0 = \begin{pmatrix} 1 & 1 \end{pmatrix} \qquad F_1 = \begin{pmatrix} 0 & 1 \end{pmatrix} \qquad F_2 = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

Les séquences d'entrée et de sortie sont telles que :

$$\mathbf{m} = \begin{pmatrix} \mathbf{m}(0), \mathbf{m}(1), \mathbf{m}(2), \cdots \end{pmatrix} = \begin{pmatrix} m_1(0) & m_1(1) & m_1(2) & \cdots \end{pmatrix}$$
$$\mathbf{c} = \begin{pmatrix} \mathbf{c}(0), \mathbf{c}(1), \mathbf{c}(2), \cdots \end{pmatrix} = \begin{pmatrix} c_1(0) & c_2(0) & c_1(1) & c_2(1) & c_1(2) & c_2(2) & \cdots \end{pmatrix}$$

D'après l'équation (1.21), les liens entre les bits de sortie et d'entrée à l'instant t sont :

$$c_1(t) = m_1(t) + m_1(t-2)$$
$$c_2(t) = m_1(t) + m_1(t-1) + m_1(t-2)$$

Prenons une séquence d'entrée composée de 4 mots d'information. A la fin de cette séquence, il faudra rajouter $K-1$ bits à zéros, soit :

$$\mathbf{m} = \begin{pmatrix} m_1(0) & m_1(1) & m_1(2) & m_1(3) & 0 & 0 \end{pmatrix}$$

Les registres du codeur sont initialisés à zéros donc $m_1(-1) = m_1(-2) = 0$. La longueur de la séquence étant égale à 4, nous obtiendrons en sortie 6 mots de code, le tableau ci-dessous récapitule les bits des sorties 1 et 2 obtenus aux instants $t = 0, \cdots, 5$:

t	$c_1(t)$	$c_2(t)$
0	$m_1(0) + 0$	$m_1(0) + 0 + 0$
1	$m_1(1) + 0$	$m_1(0) + m_1(1) + 0$
2	$m_1(0) + m_1(2)$	$m_1(0) + m_1(1) + m_1(2)$
3	$m_1(1) + m_1(3)$	$m_1(1) + m_1(2) + m_1(3)$
4	$m_1(2) + 0$	$m_1(2) + m_1(3) + 0$
5	$m_1(3) + 0$	$m_1(3) + 0 + 0$

On peut écrire une matrice de codage qui tiendra compte de la partie "remplissage" des registres et qui négligera la partie "vidange". En notant L le nombre de mots d'information de la séquence à coder, la matrice de codage d'un code convolutif est une $k.L \times n.L$ matrice définie par :

$$F = \begin{pmatrix} F_0 & \cdots & F_{K-1} & & \\ & \ddots & \vdots & \ddots & \\ & & F_0 & \cdots & F_{K-1} \\ & & & \ddots & \vdots \\ & & & & F_0 \end{pmatrix} \tag{1.24}$$

et l'expression de la séquence de sortie en fonction de la séquence d'entrée est :

$$\mathbf{c} = \mathbf{m}.F \tag{1.25}$$

Exemple 1.8.

Suite de l'exemple 1.7.
La matrice de codage pour coder la séquence d'entrée précédente (composée de 4 mots d'information) est :

$$F = \begin{pmatrix} 1\,1 & 0\,1 & 1\,1 & \\ & 1\,1 & 0\,1 & 1\,1 \\ & & 1\,1 & 0\,1 \\ & & & 1\,1 \end{pmatrix}$$

La séquence d'entrée sera :

$$\mathbf{m} = \begin{pmatrix} m_1(0) & m_1(1) & m_1(2) & m_1(3) \end{pmatrix}$$

et la séquence de sortie :

$$\mathbf{c} = \begin{pmatrix} c_1(0) & c_2(0) & c_1(1) & c_2(1) & c_1(2) & c_2(2) & c_1(3) & c_2(3) \end{pmatrix}$$

L'expression des bits de sortie est récapitulée dans le tableau suivant :

t	$c_1(t)$	$c_2(t)$
0	$m_1(0)$	$m_1(0)$
1	$m_1(1)$	$m_1(0) + m_1(1)$
2	$m_1(0) + m_1(2)$	$m_1(0) + m_1(1) + m_1(2)$
3	$m_1(1) + m_1(3)$	$m_1(1) + m_1(2) + m_1(3)$

1.4.3.2 Représentation polynomiale

A partir de fonction génératrice ou de transformation en z, il est possible d'associer à une suite d'éléments binaires une série formelle.

Définition 1.5. *Une fonction génératrice ou une transformation en z est une application qui, à une suite* $(\mathbf{a}(t))_{t\geq 0}$ *d'éléments de F, associe la série formelle :*

$$a(z) = \sum_{t\geq 0} \mathbf{a}(t).z^{-t} \tag{1.26}$$

L'ensemble des séries formelles à coefficients sur F constitue un anneau commutatif noté $F\,[[Z]]$.

Dans le cas des codes convolutifs, les éléments binaires de la suite sont émis au cours du temps, l'indéterminée en z^{-1} sera alors notée par convention D, pour "*delay*". De plus, l'ensemble de ces séries formelles peuvent être prolongées dans le corps des séries de Laurent de la forme :

$$a(D) = \sum_{t\geq 0} \mathbf{a}(t).D^t \tag{1.27}$$

Ainsi, les fonctions génératrices de la séquence des mots d'information $\mathbf{m} = \{\mathbf{m}(t)\}_{t\geq 0}$ et de la séquence des mots de code $\mathbf{c} = \{\mathbf{c}(t)\}_{t\geq 0}$ seront, respectivement :

$$\begin{aligned} m(D) &= \sum_{t\geq 0} \mathbf{m}(t).D^t = \left(\sum_{t\geq 0} m_1(t).D^t, \cdots, \sum_{t\geq 0} m_k(t).D^t \right) \\ c(D) &= \sum_{t\geq 0} \mathbf{c}(t).D^t = \left(\sum_{t\geq 0} c_1(t).D^t, \cdots, \sum_{t\geq 0} c_n(t).D^t \right) \end{aligned} \tag{1.28}$$

En notant $g_{i,j}(D)$ les polynômes générateurs du codeur tels que :

$$g_{i,j}(D) = g_{i,j}(0) + g_{i,j}(1).D + \cdots + g_{i,j}(K-1).D^{K-1} \tag{1.29}$$

on obtient la matrice génératrice, notée $G(D)$, définie par :

$$G(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,n}(D) \\ \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,n}(D) \end{bmatrix} \tag{1.30}$$

Il est alors possible d'écrire la relation liant les séquences d'entrée et de sortie :

$$c(D) = m(D).G(D) \tag{1.31}$$

Exemple 1.9.

La figure 1.10 représente le schéma d'implémentation du code convolutif présenté dans l'exemple 1.7. Ce codeur est composé de 1 entrée (notée m_1), de 2 sorties (notées respectivement c_1 et c_2) et d'une longueur de contrainte de 3 (représentée à l'aide des 2 bascules à décalage (D)).

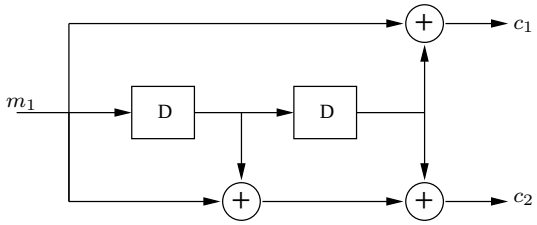


Figure 1.10 — Schéma d'implémentation d'un codeur NRNSC

La matrice génératrice de ce codeur peut être représentée sous trois formes différentes :
- Matrice génératrice en octal : $G = \begin{pmatrix} \mathbf{g}_{1,1} & \mathbf{g}_{1,2} \end{pmatrix} = \begin{pmatrix} 5 & 7 \end{pmatrix}$
- Matrice génératrice en binaire : $G = \begin{pmatrix} \mathbf{g}_{1,1} & \mathbf{g}_{1,2} \end{pmatrix} = \begin{pmatrix} (1\ 0\ 1) & (1\ 1\ 1) \end{pmatrix}$
- Matrice génératrice polynomiale : $G(D) = \begin{bmatrix} g_{1,1}(D) & g_{1,2}(D) \end{bmatrix} = \begin{bmatrix} 1+D^2 & 1+D+D^2 \end{bmatrix}$

Ce codeur étant composé d'une seule entrée, la séquence des mots d'information, $m(D)$, est telle que :

$$m(D) = \sum_{t \geq 0} m_1(t).D^t = m_1(0) + m_1(1).D + m_1(2).D^2 + m_1(3).D^3 + \cdots$$

et la séquence de sortie :

$$c(D) = (c_1(D), c_2(D)) = \left(\sum_{t \geq 0} c_1(t).D^t, \sum_{t \geq 0} c_2(t).D^t \right)$$

Reprenons l'exemple d'une séquence d'entrée composée de 4 mots d'information, soit $L = 4$. Les séquences des 2 sorties seront :

$$\begin{aligned} c_1(D) &= m(D).[1+D^2] \\ &= m_1(0) + m_1(1).D + (m_1(0) + m_1(2)).D^2 + (m_1(1) + m_1(3)).D^3 + m_1(2).D^4 + m_1(3).D^5 \end{aligned}$$

$$\begin{aligned} c_2(D) &= m(D).[1+D+D^2] \\ &= m_1(0) + (m_1(0) + m_1(1)).D + (m_1(0) + m_1(1) + m_1(2)).D^2 + (m_1(1) + m_1(2) + m_1(3)).D^3 \\ &\quad + (m_1(2) + m_1(3)).D^4 + m_1(3).D^5 \end{aligned}$$

Comme précédemment, on ne tiendra pas compte des $K-1$ derniers mots de code. Ces mots de code correspondent ici aux coefficients de degré supérieur ou égale à L. Nous pouvons vérifier que ces séquences représentées sous forme polynomiale correspondent aux séquences binaires de l'exemple 1.8.

1.4.3.3 La mémoire d'un code convolutif

Par la suite, nous noterons $C(n, k, K)$ un codeur de paramètres n, k et K. Nous noterons $\deg g_{i,j}(D)$, le degré du polynôme générateur $g_{i,j}(D)$ et μ_i le degré de la i-ème ligne de la matrice génératrice $G(D)$ comme étant le degré maximum de ses polynômes :

$$\mu_i = \max_{j=1,\cdots,n} (\deg g_{i,j}(D)) \tag{1.32}$$

Le degré d'une ligne, μ_i, correspond en fait à la mémoire de la i-ème entrée du codeur. Nous avons vu précédemment que chaque mot de code dépendait du mot d'information venant d'entrer dans le codeur mais également des $K-1$ mots ayant été introduits précédemment. La mémoire du codeur qui correspond au nombre de mots d'information gardés en mémoire sera notée μ et définie par :

$$\mu = K - 1 = \max_{i=1,\cdots,k} (\mu_i) \tag{1.33}$$

Dans la suite du document, nous définirons chaque polynôme générateur d'une matrice génératrice, notée $\mathbf{g}_{i,j}$, par un vecteur composé de $\mu_i + 1$ éléments :

$$\mathbf{g}_{i,j} = \begin{pmatrix} g_{i,j}(0) & g_{i,j}(1) & \cdots & g_{i,j}(\mu_i) \end{pmatrix} \tag{1.34}$$

Si une entrée est de degré inférieur à la mémoire du codeur, les coefficients des polynômes générateurs de cette entrée, qui sont compris entre $\mu_i + 1$ et μ, seront considérés comme nul. Soit :

$$g_{i,j}(l) = 0 \quad \forall l = \mu_i + 1, \cdots, \mu \tag{1.35}$$

Exemple 1.10.

Prenons l'exemple d'un $C(3, 2, 3)$ code défini par la matrice génératrice $G(D)$ suivante :

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

La mémoire de chaque entrée et celle du codeur sont telles que :

$$\mu_1 = \max_{j=1,\cdots,n} (\deg g_{1,j}(D)) = 1$$
$$\mu_2 = \max_{j=1,\cdots,n} (\deg g_{2,j}(D)) = 2$$
$$\mu = (\max\{\mu_1, \mu_2\}) = 2$$

D'après la matrice génératrice $G(D)$, les polynômes générateurs représentés en binaire et en octal sont :

$$\mathbf{g}_{1,1} = (0\ 1) = 1 \quad \mathbf{g}_{1,2} = (1\ 0) = 2 \quad \mathbf{g}_{1,3} = (1\ 1) = 3$$
$$=> \mathbf{g}_{1,j} = (g_{1,j}(0)\ g_{1,j}(1))$$

$$\mathbf{g}_{2,1} = (1\ 0\ 0) = 4 \quad \mathbf{g}_{2,2} = (0\ 0\ 1) = 1 \quad \mathbf{g}_{2,3} = (1\ 1\ 1) = 7$$
$$=> \mathbf{g}_{2,j} = (g_{2,j}(0)\ g_{2,j}(1)\ g_{2,j}(2))$$

La première entrée étant de degré inférieur à la mémoire du codeur, les coefficients $g_{1,j}(2)$ de cette entrée seront considérés comme étant à zéros. D'après l'équation (1.22), les K sous-matrices de codage sont :

$$F_0 = \begin{pmatrix} g_{1,1}(0) & g_{1,2}(0) & g_{1,3}(0) \\ g_{2,1}(0) & g_{2,2}(0) & g_{2,3}(0) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$F_1 = \begin{pmatrix} g_{1,1}(1) & g_{1,2}(1) & g_{1,3}(1) \\ g_{2,1}(1) & g_{2,2}(1) & g_{2,3}(1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$F_2 = \begin{pmatrix} g_{1,1}(2) & g_{1,2}(2) & g_{1,3}(2) \\ g_{2,1}(2) & g_{2,2}(2) & g_{2,3}(2) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

et le mot de code reçu à l'instant t sera :

$$\begin{aligned} &\mathbf{c}(t) = \begin{bmatrix} m_1(t) & m_2(t) \end{bmatrix}.F_0 + \begin{bmatrix} m_1(t-1) & m_2(t-1) \end{bmatrix}.F_1 + \begin{bmatrix} m_1(t-2) & m_2(t-2) \end{bmatrix}.F_2 \\ &= \begin{pmatrix} m_2(t) + m_1(t-1) & m_1(t) + m_2(t-2) & m_1(t) + m_2(t) + m_1(t-1) + m_2(t-1) + m_2(t-2) \end{pmatrix} \end{aligned}$$

1.4.4 Les codeurs de forme RSC

Un codeur de forme RSC est un codeur qui est systématique et récursif. Le terme systématique signifie que les k bits d'entrée du codeur se retrouvent en sortie du codeur, donc que la matrice génératrice d'un codeur RSC sera composée d'une matrice identité de taille k. Le terme récursif signifie que les sorties du codeur peuvent dépendre des sorties précédentes, ce qui implique que certaines sorties seront rebouclées en entrée. Il en découle que les polynômes générateurs d'un codeur RSC ne seront pas de simple polynômes, mais des fractions rationnelles polynomiales. Avant d'introduire ces codeurs de forme RSC, nous allons tout d'abord voir qu'il existera toujours un codeur NRNSC équivalent à un codeur RSC.

1.4.4.1 Les codeurs équivalents

Le terme de codeur *équivalent* ne signifie pas que deux matrices génératrices de codeurs équivalents engendreront la même séquence codée mais que ces séquences codées appartiendront au même code.

Définition 1.6. *Deux matrices génératrices $G(D)$ et $G'(D)$ sont équivalentes si elles engendrent le même code. Deux codes convolutifs sont équivalents si leurs matrices génératrices sont équivalentes.*

Théorème 1.1. *Deux codeurs de rendement $r = k/n$ et de matrices génératrices $G(D)$ et $G'(D)$ sont équivalents si et seulement si il existe une $k \times k$ matrice inversible $L(D)$ tel que :*

$$G(D) = L(D).G'(D) \tag{1.36}$$

Par définition, une famille de code contient au minimum un codeur de forme NRNSC et un codeur équivalent de forme RSC. De ce fait, il est toujours possible de passer de la représentation d'un codeur NRNSC vers son RSC équivalent et inversement.

1.4.4.2 Passage d'un codeur NRNSC sous sa forme RSC équivalente

Dans le cas d'un codeur de rendement $1/n$, la relation liant la matrice génératrice du code NRNSC et de son RSC équivalent est :

$$G_{RSC}(D) = \frac{G_{NRNSC}(D)}{g_{1,1}(D)} \tag{1.37}$$

Exemple 1.11.

La figure 1.10 de l'exemple 1.9 représentait le schéma d'implémentation d'un $C(2,1,3)$ codeur de forme NRNSC. Nous avons représenté ci-dessous le schéma de son codeur équivalent sous forme

RSC.

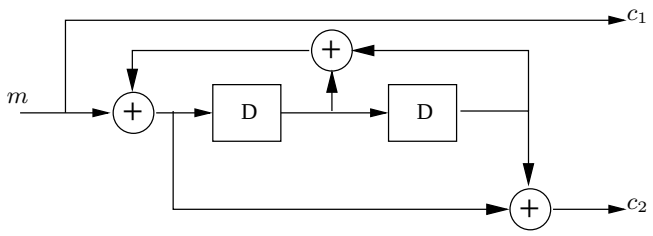


Figure 1.11 — Schéma d'implémentation d'un codeur RSC

La matrice génératrice du codeur NRNSC était :

$$G_{NRNSC}(D) = \begin{bmatrix} 1 + D^2 & 1 + D + D^2 \end{bmatrix}$$

D'après l'équation (1.37), la matrice génératrice du codeur RSC équivalent est :

$$G_{RSC}(D) = \begin{bmatrix} 1 & \frac{1+D+D^2}{1+D^2} \end{bmatrix}$$

Pour les codeurs ayant un nombre d'entrées strictement supérieur à 1, la relation entre les deux matrices génératrices est plus complexe. La partie ci-dessous va nous permettre de décrire le lien entre ces deux matrices qui a été proposé dans [JZ99].

La matrice génératrice d'un codeur NRNSC est définie par ses $k \times n$ polynômes générateurs :

$$G_{NRNSC}(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) & \cdots & g_{1,n}(D) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) & \cdots & g_{k,n}(D) \end{bmatrix} \tag{1.38}$$

Nous noterons $L(D)$, la matrice qui contient les $k \times k$ premiers polynômes générateurs de $G_{NRNSC}(D)$, telle que :

$$L(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) \\ \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) \end{bmatrix} \tag{1.39}$$

Afin d'obtenir la matrice équivalente du codeur RSC, il faut multiplier la matrice inverse de $L(D)$ par la matrice génératrice du codeur NRNSC :

$$G_{RSC}(D) = L^{-1}(D).G_{NRNSC}(D) = \frac{1}{\det L(D)}.\text{adj } L(D).G_{NRNSC}(D) \tag{1.40}$$

avec $\det L(D)$ qui correspond au déterminant de la matrice $L(D)$ et adj $L(D)$ la matrice adjointe (ou comatrice transposée) de $L(D)$. La matrice génératrice du codeur RSC équivalent, obtenue avec cette formule, est une matrice composée de la matrice identité de taille k et de polynômes qui sont des fractions rationnelles polynomiales. Elle est de la forme :

$$G_{RSC}(D) = \begin{bmatrix} 1 & & \frac{f_{1,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{1,n}(D)}{f_{1,1}(D)} \\ & \ddots & \vdots & \cdots & \vdots \\ & & 1 & \frac{f_{k,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{k,n}(D)}{f_{1,1}(D)} \end{bmatrix} \tag{1.41}$$

où les $f_{i,j}(D)$ ($\forall i = 1, \cdots, k$ et $\forall j = k+1, \cdots, n$) représentent les polynômes générateurs et

$f_{1,1}(D)$ le polynôme de *feedback*. Pour qu'un codeur RSC soit réalisable, il est nécessaire que le bit de poids faible du polynôme de feedback, $f_{1,1}(0)$, soit égale à 1. De ce fait, chaque ligne de la matrice génératrice d'un codeur RSC sera de degré identique. Nous noterons $\mu^{\perp}$ la mémoire d'un codeur de forme RSC et $\mu_i^{\perp}$ la mémoire correspondant à la i-ème entrée, nous avons :

$$\mu^{\perp} = \mu_1^{\perp} = \cdots = \mu_k^{\perp} \tag{1.42}$$

Dans la section 1.4.5, nous montrerons que cette mémoire, $\mu^{\perp}$, correspond à la mémoire du code dual.

Les polynômes générateurs d'un codeur de forme RSC correspondent également aux mineurs d'ordre k de la matrice génératrice du codeur NRNSC, que nous noterons $\Delta_i^j(D)$. Le lecteur intéressé pourra se référer à l'annexe A où une démonstration détaillée des liens entre les mineurs $\Delta_i^j(D)$ et les polynômes $f_{i,j}(D)$ est exposée.

Exemple 1.12.

Reprenons le codeur NRNSC défini dans l'exemple 1.10 de matrice génératrice :

$$G_{NRNSC}(D) = \begin{bmatrix} D & 1 & 1+D \\ 1 & D^2 & 1+D+D^2 \end{bmatrix}$$

La matrice $L(D)$ qui est composée des $k \times k$ premiers polynômes générateurs est telle que :

$$L(D) = \begin{bmatrix} D & 1 \\ 1 & D^2 \end{bmatrix}$$

D'après l'équation (1.40), la matrice du codeur RSC équivalent est :

$$G_{RSC}(D) = \frac{1}{1+D^3} \cdot \begin{bmatrix} 1+D^3 & 0 & 1+D+D^3 \\ 0 & 1+D^3 & 1+D^2+D^3 \end{bmatrix}$$

1.4.5 Les codes duaux

Lors de la présentation des codes en blocs, nous avons mis en évidence l'importance de la matrice génératrice du code dual, appelée matrice de parité du code, pour le détection et/ou la correction des erreurs. La matrice de parité d'un code en bloc est une matrice composée de $n-k$ mots de code qui permet de vérifier, en calculant le syndrome, si un mot appartient ou non au code. Un code convolutif peut également être représenté par la matrice génératrice du code dual [For73], mais de par la mémoire du code cette matrice n'est pas comme dans le cas d'un code en bloc constituée des mots de code de n bits.

Définition 1.7. *Soit $G(D)$ une matrice génératrice du code C. Une matrice $H(D)$ de rang $(n-k)$ composée de $(n-k) \times n$ polynômes est une matrice de parité du code C si et seulement si :*

$$G(D).H^T(D) = 0 \tag{1.43}$$

Corollaire 1.2. *Soit $H(D)$ une matrice de parité du code C. Une séquence de mots de code $c(D)$ appartient au code C si et seulement si :*

$$c(D).H^T(D) = 0 \tag{1.44}$$

La matrice de parité d'un code convolutif sous sa forme systématique est une $(n-k) \times n$ matrice

polynomiale, telle que :

$$H(D) = \begin{bmatrix} h_{1,1}(D) & \cdots & h_{1,k}(D) & h_0(D) & & \\ \vdots & \cdots & \vdots & & \ddots & \\ h_{n-k,1}(D) & \cdots & h_{n-k,k}(D) & & & h_0(D) \end{bmatrix} \tag{1.45}$$

où les $h_{i,j}(D)$ correspondent aux polynômes générateurs de la matrice de parité du code.

Rappelons que les polynômes générateurs du codeur RSC, $f_{l,m}(D)$, et les mineurs d'ordre k de la matrice génératrice du codeur NRNSC équivalent, $\Delta_l^m(D)$, sont égaux. En prenant les polynômes générateurs de la matrice de parité tels que :

$$\begin{cases} h_{i,j}(D) = f_{j,i+k}(D) = \Delta_j^{i+k}(D) & \forall i = 1, \cdots, n-k \text{ et } \forall j = 1, \cdots, k \\ h_0(D) = f_{1,1}(D) = \Delta_1^1(D) \end{cases} \tag{1.46}$$

la définition 1.7 est respectée. Le lecteur intéressé en trouvera une démonstration en annexe B.

Exemple 1.13.

Reprenons l'exemple du $C(2,1,3)$ code de l'exemple 1.9. La matrice génératrice et la matrice de parité sont telles que :

$$G(D) = \begin{bmatrix} 1+D^2 & 1+D+D^2 \end{bmatrix} \quad \text{et} \quad H(D) = \begin{bmatrix} 1+D+D^2 & 1+D^2 \end{bmatrix}$$

La matrice $H(D)$ est une matrice de parité du code puisque $G(D).H^T(D) = 0$. Les séquences de la première et de la seconde sortie sont :

$$c_1(D) = m(D).[1+D^2] = \sum_{t\geq 0} m_1(t).D^t + \sum_{t\geq 0} m_1(t).D^{t+2}$$
$$c_2(D) = m(D).[1+D+D^2] = \sum_{t\geq 0} m_1(t).D^t + \sum_{t\geq 0} m_1(t).D^{t+1} + \sum_{t\geq 0} m_1(t).D^{t+2}$$

D'après le corollaire 1.2, une séquence codée appartient au code, si et seulement si $c(D).H^T(D) = 0$.

$$c(D).H^T(D) = \begin{bmatrix} c_1(D) & c_2(D) \end{bmatrix} . \begin{bmatrix} 1+D+D^2 \\ 1+D^2 \end{bmatrix} = c_1(D).[1+D+D^2] + c_2(D).[1+D^2]$$

En séparant les pistes de codées, nous obtenons :

$$\begin{aligned} c_1(D).[1+D+D^2] &= \sum_{t\geq 0} c_1(t).D^t + \sum_{t\geq 0} c_1(t).D^{t+1} + \sum_{t\geq 0} c_1(t).D^{t+2} \\ &= \sum_{t\geq 0} \left(m_1(t).D^t + m_1(t).D^{t+2} + m_1(t).D^{t+1} + m_1(t).D^{t+3} + m_1(t).D^{t+2} + m_1(t).D^{t+4} \right) \end{aligned}$$

$$\begin{aligned} c_2(D).[1+D^2] &= \sum_{t\geq 0} c_2(t).D^t + \sum_{t\geq 0} c_2(t).D^{t+2} \\ &= \sum_{t\geq 0} \left(m_1(t).D^t + m_1(t).D^{t+1} + m_1(t).D^{t+2} + m_1(t).D^{t+2} + m_1(t).D^{t+3} + m_1(t).D^{t+4} \right) \end{aligned}$$

En additionnant les deux séquences, nous obtenons $c(D).H^T(D) = 0$.

Nous avons vu précédemment que, par nature, un code convolutif était adapté pour coder des séquences infinies mais que, en pratique, ces séquences étaient tronquées afin d'obtenir des

trames indépendantes. De ce fait, le corollaire 1.2 est respecté aux effets de bord près, c-à-d en ne tenant pas compte des coefficients de puissance supérieure ou égale à L (avec L le nombre de mots d'information).

Comme pour la matrice génératrice du code, il est possible d'écrire la matrice de parité d'un code sous forme matricielle à coefficients binaires. Les polynômes générateurs de la matrice de parité étant égaux aux polynômes générateurs de la matrice du codeur RSC équivalent, ces polynômes sont de degré $\mu^{\perp}$. Nous définirons $\mu^{\perp}+1$ sous-matrices de parité, notée H_i ($\forall i = 0, \cdots, \mu^{\perp}$), de taille $n-k \times n$ par :

$$H_i = \begin{pmatrix} h_{1,1}(i) & \cdots & h_{1,k}(i) & h_0(i) & & \\ \vdots & \cdots & \vdots & & \ddots & \\ h_{n-k,1}(i) & \cdots & h_{n-k,k}(i) & & & h_0(i) \end{pmatrix} \tag{1.47}$$

La matrice de parité sera composée de ces $\mu^{\perp}+1$ matrices H_i :

$$H = \begin{pmatrix} H_0 & & & & & \\ H_1 & H_0 & & & & \\ \vdots & \ddots & \ddots & & & \\ H_{\mu^{\perp}} & \cdots & H_1 & H_0 & & \\ & H_{\mu^{\perp}} & \cdots & H_1 & H_0 & \\ & & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \tag{1.48}$$

Comme pour la matrice de codage, afin de ne pas tenir compte de la partie vidange des registres, la matrice H sera tronquée. En notant L le nombre de mots d'information à coder, la matrice F sera de taille $k.L \times n.L$ et la matrice de parité H de taille $(n-k).L \times n.L$.

$$F = \begin{pmatrix} F_0 & \cdots & F_{\mu-1} & F_{\mu} & & \\ & \ddots & \vdots & \vdots & \ddots & \\ & & F_0 & \vdots & \ddots & F_{\mu} \\ & & & F_0 & \ddots & \vdots \\ & & & & \ddots & \vdots \\ & & & & & F_0 \end{pmatrix} \quad H^T = \begin{pmatrix} H_0^T & \cdots & H_{\mu^{\perp}-1}^T & H_{\mu^{\perp}}^T & & \\ & \ddots & \vdots & \vdots & \ddots & \\ & & H_0^T & \vdots & \ddots & H_{\mu^{\perp}}^T \\ & & & H_0^T & \ddots & \vdots \\ & & & & \ddots & \vdots \\ & & & & & H_0^T \end{pmatrix} \tag{1.49}$$

La matrice H est telle que :

$$F.H^T = 0 \tag{1.50}$$

Nous avons vu qu'un codeur RSC avait toujours un codeur de forme NRNSC qui lui était équivalent et que deux codeurs équivalents engendraient un même Code. De par ces définitions, même si nous ne pouvons obtenir la matrice de codage F d'un code de forme RSC, la matrice de parité H sera la même pour le code NRNSC et le code RSC équivalent.

Définition 1.8. *Soit H une matrice de parité du code C de taille $(n-k).L \times n.L$. Une séquence codée* $\mathbf{c}$ *de longueur $n.L$ appartient au code C si et seulement si :*

$$\mathbf{c}.H^T = 0 \tag{1.51}$$

Exemple 1.14.

Suite de l'exemple 1.13.

Les sous-matrices de parité H_i de ce codeur sont :

$$H_0 = \begin{pmatrix} 1 & 1 \end{pmatrix} \quad H_1 = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad H_2 = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

En posant $L = 4$, les matrices de codage et de parité sont respectivement de taille 4×8 et 8×4 :

$$F = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & & \\ & & 1 & 1 & 0 & 1 & 1 & 1 \\ & & & & 1 & 1 & 0 & 1 \\ & & & & & & 1 & 1 \end{pmatrix} \quad H^T = \begin{pmatrix} 1 & 1 & 1 & \\ 1 & 0 & 1 & \\ & 1 & 1 & 1 \\ & 1 & 0 & 1 \\ & & 1 & 1 \\ & & 1 & 0 \\ & & & 1 \\ & & & 1 \end{pmatrix}$$

Les matrices génératrices du codeur NRNSC et du codeur RSC équivalent sont :

$$G_{NRNSC}(D) = \begin{bmatrix} 1 + D^2 & 1 + D + D^2 \end{bmatrix} \quad G_{RSC}(D) = \begin{bmatrix} 1 & \frac{1+D+D^2}{1+D^2} \end{bmatrix}$$

Soit $\mathbf{m} = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix}$, la séquence codée par le codeur NRNSC et celle par le RSC seront respectivement :

$$\mathbf{c}_{NRNSC} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$
$$\mathbf{c}_{RSC} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Les deux codeurs étant équivalents, la matrice de parité définie précédemment vérifie la définition 1.8 pour les deux séquences.

1.4.6 Représentation sous forme de matrices d'état

Les polynômes générateurs d'un codeur de forme RSC ne sont plus de simples polynômes mais des fractions rationnelles polynomiales. De ce fait, il n'est plus possible de représenter le codage par de simples multiplications. Pour un code RSC, nous ne pouvons obtenir la matrice de codage F sous la forme que nous avions décrite à l'équation (1.24). Nous allons introduire une nouvelle représentation qui nous permettra de traiter indifféremment les codeurs de forme RSC et NRNSC. Nous passerons par une représentation sous forme de matrices d'état qui est adaptée pour représenter des systèmes ayant des retours d'information ("feedback"). Cette représentation d'état qui est bien connue dans le domaine de l'automatisme a été adaptée au cas des codes convolutifs dans [MS67].

1.4.6.1 Le codage

Chaque entrée d'un codeur est composée d'une mémoire interne de longueur μ_i. Nous appellerons un vecteur d'état, un vecteur composé des éléments qui sont gardés en mémoire par le codeur. Nous noterons $\mathbf{s}_i(t)$, le vecteur d'état de la i-ème entrée à l'instant t défini par :

$$\mathbf{s}_i(t) = \begin{pmatrix} s_i(t,0) & \cdots & s_i(t,\mu_i - 1) \end{pmatrix} \tag{1.52}$$

La mémoire globale du codeur sera représentée par un vecteur d'état, noté $\mathbf{s}(t)$, composé des vecteurs d'état de chaque entrée tel que :

$$\mathbf{s}(t) = \begin{pmatrix} \mathbf{s}_1(t) & \mathbf{s}_2(t) & \cdots & \mathbf{s}_k(t) \end{pmatrix} \tag{1.53}$$

Nous pouvons décrire formellement un codeur convolutif par les équations de description de "l'espace d'état" connues dans le domaine de l'automatique que nous avons réécrit sous forme

matricielle :

$$\begin{cases} \mathbf{s}(t+1) = [\mathbf{s}_1(t)\ m_1(t)\ \cdots\ \mathbf{s}_k(t)\ m_k(t)]\,.G_D \\ \mathbf{c}(t) = [\mathbf{s}_1(t)\ m_1(t)\ \cdots\ \mathbf{s}_k(t)\ m_k(t)]\,.G_N \end{cases} \tag{1.54}$$

avec G_D et G_N qui sont des matrices binaires. La matrice G_N correspond à la partie numérateur des polynômes générateurs et G_D à la partie dénominateur.

Le lecteur intéressé trouvera en annexe C une démonstration détaillée sur la construction des matrices d'état d'un codeur de forme NRNSC et RSC.

1.4.6.2 Matrices d'état d'un codeur NRNSC

La matrice d'état qui correspond à la partie numérateur, notée $G_{N_{(NRNSC)}}$, est une matrice de taille $\left(\sum_{i=1}^{k} \mu_i + k\right) \times n$ qui est composée des bits correspondant aux coefficients des $k \times n$ polynômes générateurs :

$$G_{N_{(NRNSC)}} = \begin{pmatrix} g_{1,1}(\mu_1) & g_{1,2}(\mu_1) & \cdots & g_{1,n}(\mu_1) \\ \vdots & \vdots & \cdots & \vdots \\ g_{1,1}(0) & g_{1,2}(0) & \cdots & g_{1,n}(0) \\ \vdots & \vdots & \cdots & \vdots \\ g_{k,1}(\mu_k) & g_{k,2}(\mu_k) & \cdots & g_{k,n}(\mu_k) \\ \vdots & \vdots & \cdots & \vdots \\ g_{k,1}(0) & g_{k,2}(0) & \cdots & g_{k,n}(0) \end{pmatrix} \tag{1.55}$$

La matrice d'état correspondant à la partie dénominateur, notée $G_{D_{(NRNSC)}}$, est une matrice bloc-diagonale de taille $\left(\sum_{i=1}^{k} \mu_i + k\right) \times \sum_{i=1}^{k} \mu_i$ composée de k matrices G_{D_i} :

$$G_{D_{(NRNSC)}} = \begin{pmatrix} G_{D_1} & & & \\ & G_{D_2} & & \\ & & \ddots & \\ & & & G_{D_k} \end{pmatrix} \quad \text{avec} \quad G_{D_i} = \begin{pmatrix} \mathbf{0}_{1,\mu_i} \\ \mathbf{I}_{\mu_i} \end{pmatrix},\ \forall i = 1, \cdots, k \tag{1.56}$$

Les matrices G_{D_i} sont de taille $(\mu_i + 1) \times \mu_i$ et composées d'un vecteur ligne nul de taille μ_i, noté $\mathbf{0}_{1,\mu_i}$ et d'une matrice identité de taille μ_i, notée $\mathbf{I}_{\mu_i}$.

Exemple 1.15.

Reprenons l'exemple du codeur $C(3, 2, 3)$ qui a été présenté dans l'exemple 1.10. Les polynômes générateurs représentés sous leurs formes vectorielles sont tels que :

$$\begin{array}{ccc} \mathbf{g}_{1,1} = (0\ 1) & \mathbf{g}_{1,2} = (1\ 0) & \mathbf{g}_{1,3} = (1\ 1) \\ \mathbf{g}_{2,1} = (1\ 0\ 0) & \mathbf{g}_{2,2} = (0\ 0\ 1) & \mathbf{g}_{2,3} = (1\ 1\ 1) \end{array}$$

Les matrices d'état pour la partie numérateur et dénominateur sont respectivement :

$$G_{N_{(NRNSC)}} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \quad \text{et} \quad G_{D_{(NRNSC)}} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Les vecteurs d'état pour la première et la seconde sortie sont tels que :

$$\begin{cases} \mathbf{s}_1(t) = \begin{pmatrix} s_1(t,0) \end{pmatrix} \\ \mathbf{s}_2(t) = \begin{pmatrix} s_2(t,0) & s_2(t,1) \end{pmatrix} \end{cases}$$

et le vecteur d'état représentant l'ensemble de la mémoire du codeur est :

$$\mathbf{s}(t) = \begin{pmatrix} s_1(t,0) & s_2(t,0) & s_2(t,1) \end{pmatrix}$$

D'après l'équation (1.54), nous obtenons :

$$\begin{cases} \mathbf{s}(t+1) = \begin{bmatrix} s_1(t,0) & m_1(t) & s_2(t,0) & s_2(t,1) & m_2(t) \end{bmatrix} . G_{D_{(NRNSC)}} \\ \mathbf{c}(t) = \begin{bmatrix} s_1(t,0) & m_1(t) & s_2(t,0) & s_2(t,1) & m_2(t) \end{bmatrix} . G_{N_{(NRNSC)}} \end{cases}$$

Le vecteur d'état à l'instant $t+1$ peut être représenté en fonction des entrées :

$$\begin{cases} s_1(t+1,0) = m_1(t) \\ s_2(t+1,0) = s_2(t,1) = m_2(t-1) \\ s_2(t+1,1) = m_2(t) \end{cases}$$

Les 3 sorties du codeur peuvent être représentées en fonction des bits d'entrée et des bits du vecteur d'état :

$$\begin{cases} c_1(t) = s_1(t,0) + m_2(t) \\ c_2(t) = m_1(t) + s_2(t,0) \\ c_3(t) = s_1(t,0) + m_1(t) + s_2(t,0) + s_2(t,1) + m_2(t) \end{cases}$$

et enfin en fonction des bits d'entrée du codeur :

$$\begin{cases} c_1(t) = m_1(t-1) + m_2(t) \\ c_2(t) = m_1(t) + m_2(t-2) \\ c_3(t) = m_1(t) + m_1(t-1) + m_2(t) + m_2(t-1) + m_2(t-2) \end{cases}$$

Nous pouvons vérifier que le lien entre les entrées et les sorties est le même que celui obtenu précédemment (exemple 1.10).

1.4.6.3 Matrices d'état d'un codeur RSC

Nous avons vu précédemment que la matrice génératrice d'un codeur RSC était composée de fractions rationnelles polynomiales. Le lien entre les polynômes générateurs du codeur NRNSC et de son RSC équivalent a été explicité dans l'annexe A. La matrice du codeur RSC est de la forme :

$$G_{RSC}(D) = \begin{bmatrix} 1 & & & \frac{f_{1,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{1,n}(D)}{f_{1,1}(D)} \\ & \ddots & & \vdots & \cdots & \vdots \\ & & 1 & \frac{f_{k,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{k,n}(D)}{f_{1,1}(D)} \end{bmatrix} \tag{1.57}$$

avec les polynômes $f_{i,j}(D)$ qui correspondent aux mineurs d'ordre k de la matrice génératrice du codeur NRNSC et $f_{1,1}(D)$ qui correspond au polynôme de feedback.

De par le rebouclage des sorties en entrée du codeur, les matrices d'état d'un codeur RSC sont un peu différentes de celles des codeurs NRNSC. Rappelons que les degrés des lignes de la matrice génératrice d'un codeur RSC sont tous identiques et que nous l'avons noté $\mu^{\perp}$. La matrice d'état de la partie dénominateur, $G_{D_{(RSC)}}$, est une matrice de taille $(k.(\mu^{\perp}+1) \times k.\mu^{\perp})$ qui peut être définie par k matrices, G_{D_1}, de taille $(\mu^{\perp}+1) \times \mu^{\perp}$. Cette matrice est composée de deux vecteurs nuls de taille $\mu^{\perp}-1$ séparés par une matrice identité de taille $\mu^{\perp}-1$ et des bits du polynôme de feedback $(f_{1,1}(D))$.

$$G_{D_{(RSC)}} = \begin{pmatrix} G_{D_1} & & & \\ & G_{D_1} & & \\ & & \ddots & \\ & & & G_{D_1} \end{pmatrix} \quad \text{avec} \quad G_{D_1} = \begin{pmatrix} 0 & \cdots & 0 & f_{1,1}(\mu^{\perp}) \\ 1 & & & f_{1,1}(\mu^{\perp}-1) \\ & \ddots & & \vdots \\ & & 1 & f_{1,1}(1) \\ 0 & \cdots & 0 & f_{1,1}(0) \end{pmatrix} \tag{1.58}$$

La matrice d'état correspondant à la partie numérateur, notée $G_{N_{(RSC)}}$, est de taille $k.(\mu^{\perp}+1) \times n$.

$$G_{N_{(RSC)}} = \begin{pmatrix} 0 & & q_{1,k+1}(\mu^{\perp}) & \cdots & q_{1,n}(\mu^{\perp}) \\ \vdots & & \vdots & \cdots & \vdots \\ 0 & & q_{1,k+1}(1) & \cdots & q_{1,n}(1) \\ 1 & & f_{1,k+1}(0) & \cdots & f_{1,n}(0) \\ & \ddots & \vdots & \cdots & \vdots \\ & 0 & q_{k,k+1}(\mu^{\perp}) & \cdots & q_{k,n}(\mu^{\perp}) \\ & \vdots & \vdots & \cdots & \vdots \\ & 0 & q_{k,k+1}(1) & \cdots & q_{k,n}(1) \\ & 1 & f_{k,k+1}(0) & \cdots & f_{k,n}(0) \end{pmatrix} \tag{1.59}$$

avec les coefficients $q_{i,j}(l)$ qui dépendent de la valeur de $f_{i,j}(0)$ ($\forall i = 1, \cdots, k$ et $\forall j = k+1, \cdots, n$).

– Si $f_{i,j}(0) = 1$:

$$q_{i,j}(l) = f_{i,j}(l) + f_{1,1}(l) \qquad \forall l = 1, \cdots, \mu^{\perp} \tag{1.60}$$

– Si $f_{i,j}(0) = 0$:

$$q_{i,j}(l) = f_{i,j}(l) \qquad \forall l = 1, \cdots, \mu^{\perp} \tag{1.61}$$

Exemple 1.16.

Reprenons le $C(3,2,3)$ code dont la matrice génératrice du codeur RSC a été présentée dans l'exemple 1.15. Les polynômes de ce codeur sont :

$$\begin{cases} \mathbf{f}_{1,1} = 11 = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} \\ \mathbf{f}_{1,3} = 15 = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} \\ \mathbf{f}_{2,3} = 13 = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix} \end{cases}$$

Le bit de poids faible de chaque polynôme étant égale à 1, les coefficients $q_{i,j}(l)$ seront :

$$q_{i,3}(l) = f_{i,3}(l) + f_{1,1}(l) \qquad \forall l = 1, \cdots, 3 \text{ et } \forall i = 1, 2$$

D'après les équations (1.59) et (1.58), les matrices d'état de ce codeur sont telles que :

$$G_{N_{(RSC)}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \qquad G_{D_{(RSC)}} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1.5 Propriétés des codes convolutifs

La théorie algébrique des matrices génératrices va nous permettre de donner les propriétés des codes convolutifs et des codes qu'ils engendrent. Ces propriétés vont tout d'abord nous permettre de différencier les codeurs catastrophiques des bons codeurs, puis de définir les caractéristiques des codeurs optimaux. Les différentes propriétés que nous introduirons ici ont principalement été tirées des articles [For71] et [McE98].

1.5.1 Le degré d'un code convolutif

Nous avons défini précédemment la notion de degré d'une ligne de la matrice génératrice, telle que :

$$\mu_i = \max_{j=1,\cdots,n} (\deg g_{i,j}(D)) \tag{1.62}$$

ainsi que le degré de la matrice génératrice qui était également appelé mémoire du codeur, défini par :

$$\mu = \max_{i=1,\cdots,k} (\mu_i) \tag{1.63}$$

Voici quelques définitions qui vont permettre de définir les degrés interne et externe des codes, ainsi que le degré d'un code convolutif.

Définition 1.9. *Soit une matrice de dimension $k \times n$. On appelle mineurs d'ordre i, avec $i \leq min(k, n)$, les déterminants de ses sous-matrices carrées de taille $i \times i$.*

Nous avons démontré précédemment (voir annexe A) que les mineurs d'ordre k de la matrice génératrice d'un code correspondaient également aux polynômes générateurs de la matrice équivalente RSC. La matrice génératrice du codeur RSC est telle que :

$$G_{RSC}(D) = \frac{1}{\Delta_1^1(D)} \cdot \begin{bmatrix} \Delta_1^1(D) & & & \Delta_1^{k+1}(D) & \cdots & \Delta_1^n(D) \\ & \ddots & & \vdots & \cdots & \vdots \\ & & \Delta_1^1(D) & \Delta_k^{k+1}(D) & \cdots & \Delta_k^n(D) \end{bmatrix} \tag{1.64}$$

avec les $\Delta_i^j(D)$ qui correspondent aux mineurs d'ordre k.

Définition 1.10. *Soit $G(D)$ une matrice génératrice et $\Delta_i^j(D)$, $\forall i = 1, \cdots, k$ et $j = 1, \cdots, n$, le (i, j)-ème mineur d'ordre k de $G(D)$. On appelle degré interne, l'entier positif noté intdeg $G(D)$,*

défini par :

$$integ\ G(D) = \max_{i=1,\cdots,k;j=1,\cdots,n} \left(\deg \Delta_i^j(D)\right)$$

Définition 1.11. *Soit $G(D)$ une matrice génératrice, on appelle degré externe, l'entier positif noté extdeg $G(D)$, défini par :*

$$extdeg\ G(D) = \sum_{i=1}^{k} \mu_i$$

Définition 1.12. *On appelle degré du code convolutif C, noté $\deg C$, le degré de la matrice génératrice qui a le degré externe minimal parmi l'ensemble des matrices génératrices qui l'engendrent.*

1.5.2 Les matrices génératrices catastrophiques

Parmi l'ensemble des matrices génératrices d'un code, seules quelques-unes permettent une correction des erreurs à la réception. Les autres, appelées matrices catastrophiques, vont propager l'erreur à l'infini au moment du décodage.

Définition 1.13. *Une matrice génératrice $G(D)$, de taille $k \times n$, est dite catastrophique si il existe un vecteur d'entrée $m(D)$ de poids infini, telle que :*

$$c(D) = m(D).G(D) \tag{1.65}$$

soit de poids fini.

En 1968, Massey et al. [MS68] ont démontré un théorème permettant de savoir si une matrice génératrice était catastrophique ou non.

Théorème 1.2. *Soit $G(D)$, une matrice génératrice d'un (n, k, K) code convolutif. $G(D)$ est non catastrophique si l'une des conditions suivantes est vérifiée.*

1. *Aucune entrée $m(D)$ de poids fini ne peut produire une sortie $c(D)$ de poids infini.*
2. *Le plus grand commun diviseur (PGCD) des mineurs d'ordre k de $G(D)$ est une puissance de D.*
3. *$G(D)$ possède une inverse à droite dont les coefficients sont de poids fini.*

Exemple 1.17.

Prenons l'exemple d'un $C(2, 1, 3)$ code qui aurait pour matrice génératrice :

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

Les mineurs d'ordre 1 de $G(D)$ sont : $\Delta_1(D) = 1 + D^2$ et $\Delta_2(D) = D + D^2$. Le PGCD des mineurs d'ordre 1 vaut $1 + D$. D'après le théorème précédent, la matrice de ce code est une matrice catastrophique puisque le PGCD de ces mineurs d'ordre k n'est pas une puissance de D.

En 1997, Planquette et al. ont explicité dans [PV97] une nouvelle caractérisation des codeurs catastrophiques. Notons G_l la matrice de Sylvester de taille $(K + (l-1)).k \times l.n$ (composée des

matrices définies à l'équation (1.22)) :

$$G_l = \begin{pmatrix} F_{K-1} & & & \\ F_{K-2} & F_{K-1} & & \\ \vdots & F_{K-2} & \ddots & \\ F_0 & \vdots & \ddots & F_{K-1} \\ & F_0 & \ddots & F_{K-2} \\ & & \ddots & \vdots \\ & & & F_0 \end{pmatrix} \tag{1.66}$$

En notant r le rang de la matrice G_1 et $p = \sum_{i=1}^{k}(\mu_i + 1) - r$, le codeur est non catastrophique si G_{p+1} est de rang plein ou est de rang déficient et ne possède aucun vecteur de contrôle à gauche persistant.

On appel vecteur de contrôle à gauche persistant un vecteur d'entrée qui vérifie :

$$\begin{pmatrix} \mathbf{m}_0 & \mathbf{m}_1 & \cdots & \mathbf{m}_{K-1+p} \end{pmatrix} . G_{p+1} = \mathbf{0} \tag{1.67}$$

où le vecteur d'entrée contient au moins un élément non nul.

Exemple 1.18.

Reprenons le codeur que nous avons introduit dans l'exemple 1.17. La matrice génératrice du codeur est $G = \begin{pmatrix} 3 & 5 \end{pmatrix}$. La matrice de Sylvester pour $l = 1$ est :

$$G_1 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

soit $r = \text{rang}\ (G_1) = 2$ et $p = 1$. La matrice G_{p+1} est une matrice de taille 4×4 telle que :

$$G_{p+1} = \begin{pmatrix} 1 & 1 & & \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ & & 0 & 1 \end{pmatrix}$$

Cette matrice est de rang déficient, rang $(G_{p+1}) = 3$ et le vecteur d'entrée $\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$ est un vecteur de contrôle à gauche persistant. Nous pouvons en conclure que la matrice de ce code est bien une matrice catastrophique.

1.5.3 Les codes convolutifs optimaux

L'une des propriétés d'une matrice génératrice d'un code convolutif est qu'elle doit être canonique. McEliece a démontré des théorèmes permettant de définir les propriétés d'une matrice génératrice canonique, le lecteur voulant des démonstrations sur ces théorèmes pourra se référer à [McE98].

Définition 1.14. *Soit $G(D)$ une matrice génératrice de taille $k \times n$ avec $k < n$ et Λ_i le PGCD des mineurs d'ordre i de $G(D)$. Par convention, on posera $\Lambda_0 = 1$. Soit $\gamma_i = \left(\frac{\Lambda_i}{\Lambda_{i-1}}\right)$. Les γ_i pour i de 1 à k sont appelés facteurs invariants de $G(D)$.*

Théorème 1.3. *Une matrice génératrice, $G(D)$, de taille $k \times n$ est basique si et seulement si une des conditions suivantes est vérifiée.*

1. *Les facteurs invariants de $G(D)$ valent 1.*
2. *Le plus grand commun diviseur des mineurs d'ordre k de $G(D)$ vaut 1.*
3. *$G(a)$ est de rang k pour tout a dans la clôture algébrique de F.*
4. *$G(D)$ possède une inverse à droite à coefficients dans $F[D]$ ($G(D).H(D) = \mathbf{I}_k$).*
5. *Si $c(D) = m(D).G(D)$ avec $c(D) \in F[D]^n$, alors $m(D) \in F[D]^k$ ("une sortie polynomiale implique une entrée polynomiale").*
6. *$G(D)$ est une sous-matrice d'une matrice uni-modulaire.*

Théorème 1.4. *Une matrice génératrice, $G(D)$, de taille $k \times n$ est réduite si et seulement si une des conditions suivantes est vérifiée.*

1. *Si l'on définit la matrice indicatrice de plus haut degré de chaque ligne, $\overline{G}$, par $\overline{G_{i,j}}$ comme étant le coefficient de D^{μ_i} de $G_{i,j}(D)$ où $\mu_i = deg\ G_i(D)$. Alors $\overline{G}$ est de rang k.*

$$\overline{G_{i,j}} = coeff_{D^{\mu_i}}\left(g_{i,j}(D)\right)$$

2. *$exdeg\ G(D) = integ\ G(D)$.*
3. *$\forall m(D) \in F[D]^k$*

$$deg\ m(D)G(D) = \max_{i=1,\cdots,k}\left(deg\ m_i(D) + deg\ G_i(D)\right)$$

Théorème 1.5. *Une matrice génératrice est canonique si et seulement si elle est à la fois basique et réduite.*

Exemple 1.19.

Reprenons l'exemple du $C(3,2,3)$ code présenté dans l'exemple 1.15 et de matrice génératrice :

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

Les mineurs d'ordre 2 de cette matrice sont :

$$\begin{pmatrix}\Delta_1(D) & \Delta_2(D) & \Delta_3(D)\end{pmatrix} = \begin{pmatrix}1+D+D^3 & 1+D^2+D^3 & 1+D^3\end{pmatrix}$$

Le PGCD des mineurs d'ordre 2, $PGCD\left(\Delta_1(D), \Delta_2(D), \Delta_3(D)\right)$ est égale à 1. D'après le théorème 1.4, la matrice génératrice $G(D)$ est basique.

Le degré des deux lignes de la matrice génératrice sont tels que : $\mu_1 = 1$ et $\mu_2 = 2$. La matrice indicatrice définie dans le théorème 1.4 est :

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

Cette matrice est de rang 2, ce qui signifie d'après le théorème 1.4 que la matrice génératrice est réduite.

D'après le théorème 1.5, la matrice génératrice $G(D)$ est une matrice canonique.

Exemple 1.20.

Prenons maintenant l'exemple de deux matrices génératrices, $G'_1(D)$ et $G'_2(D)$, qui sont équivalentes à la matrice $G(D)$ présentée dans l'exemple 1.19 :

$$G'_1(D) = \begin{bmatrix} 0 & 1 \\ 1+D & 0 \end{bmatrix}.G(D) = \begin{bmatrix} 1 & D^2 & 1+D+D^2 \\ D+D^2 & 1+D & 1+D^2 \end{bmatrix}$$

et

$$G'_2(D) = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} . G(D) = \begin{bmatrix} 1 & D^2 & 1+D+D^2 \\ 1+D & 1+D^2 & D^2 \end{bmatrix}$$

Dans le tableau présenté ci-dessous, nous avons représenté les différentes propriétés de ces trois matrices génératrices, $G(D)$, $G'_1(D)$ et $G'_2(D)$, à l'aide des théorèmes 1.3, 1.4 et 1.5.

Matrice	Basique ?	Réduite ?	Canonique ?
$G(D)$	$\surd$	$\surd$	$\surd$
$G'_1(D)$	$-$	$\surd$	$-$
$G'_2(D)$	$\surd$	$-$	$-$

Tableau 1.1 — Propriétés de trois matrices équivalentes

Nous pouvons voir dans le tableau 1.1 que parmi les trois matrices génératrices équivalentes présentées, seule la matrice génératrice $G(D)$ est canonique. Par conséquent, les matrices $G'_1(D)$ et $G'_2(D)$ ne seront pas utilisées en pratique puisqu'elles ne sont pas canoniques.

Pour un rendement donné, il existe plusieurs matrices génératrices canoniques. Afin d'en sélectionner l'optimale, c'est à dire celle qui permettra de corriger le plus d'erreurs, nous allons introduire la notion de *distance libre*. Cette distance correspond à la plus petite distance entre deux mots de code non nuls.

Définition 1.15. *On appelle distance libre d'un (n, k, K) code convolutif C, l'entier $d_{libre}(C)$ défini par*

$$d_{libre}(C) = \min_{\mathbf{c} \neq \mathbf{c}'} \left(d_H(\mathbf{c}, \mathbf{c}') \right),$$

avec $\mathbf{c}$ et $\mathbf{c}'$ deux mots du code C et d_H la distance de Hamming entre ces deux mots de code.

Le code convolutif sera dit optimal si pour un rendement donné, la matrice canonique est celle qui engendre la plus grande *distance libre*. Une liste non-exhaustive de codes convolutifs optimaux est proposée en annexe D.

1.6 Conclusions

Dans ce chapitre, nous avons tout d'abord montré l'importance de mettre en place des récepteurs intelligents afin de rendre les chaînes de transmission beaucoup plus souples et évolutives en terme de changement de standard. Dans ce but, nous avons tout d'abord introduit le principe de fonctionnement d'une chaîne de transmission numérique puis la signification du contexte non coopératif.

L'objectif de ces travaux étant de développer des méthodes capables d'identifier en aveugle les paramètres du code correcteur d'erreur, en particulier ceux des codes convolutifs, la seconde partie de ce chapitre était vouée à une étude sur la théorie algébrique des codes convolutifs. Cette étude nous a permis d'obtenir les propriétés indispensables pour la mise en place de nos méthodes de reconnaissance qui vous seront présentées dans le prochain chapitre.

CHAPITRE 2

Reconnaissance aveugle d'un code convolutif

2.1 Introduction

Dans ce chapitre, nous nous intéresserons à la reconnaissance aveugle des codes convolutifs. Nous développerons des méthodes qui permettront, à partir de la seule connaissance du train binaire issu du démodulateur, d'identifier l'ensemble des paramètres d'un code convolutif. A ce jour, la littérature concernant cette thématique reste peu riche, même si elle a tendance ces derniers temps à se développer. Les premiers travaux ont été initiés par Rice [Ric95]. Dans cet article, une première approche permettant d'identifier des codes convolutifs de rendement 1/2 a été développée lorsque le train binaire reçu était non-entaché d'erreurs. Cette méthode a ensuite été étendue au cas des codes de rendement $1/n$ par Filiol [Fil97], puis aux codes de rendement $(n-1)/n$ dans [Fil01]. Dans [Bar05], le même principe a été repris dans le cadre de la reconnaissance aveugle des turbocodes composés de codes convolutifs de rendement k/n. Les premiers travaux permettant d'identifier quelques paramètres d'un code convolutif dans le cadre d'une communication bruitée ont été initiés par Barbier et al. dans [BSH06]. Puis dans [DH07], une approche différente basée sur l'algorithme EM (pour Expectation-Maximization) a été proposé par Dingel et al., dans le cadre de l'identification aveugle de code convolutif de rendement $1/n$ en présence d'erreurs de transmission. Une méthode basée sur l'algorithme d'Euclide a été proposée par Wang et al. [WHZ07]. Cette approche permet dans un contexte bruité d'identifier un code de rendement 1/2.

En parallèle avec ces avancées sur l'identification des codes convolutifs, des travaux concernant la reconnaissance des codes en bloc, avec ou sans bruit, ont été mené par Valembois [Val00], par Burel et al. [BG03] et par Cluzeau [Clu04]. Des travaux portant sur la reconnaissance de l'utilisation ou non d'un code linéaire ont également été effectués par Chabot [Cha07]. Afin de lutter efficacement contre les erreurs de type burst, un code correcteur d'erreurs est généralement suivi d'un entrelaceur. Des travaux permettant de reconnaître la taille de l'entrelaceur et quelques paramètres du code utilisé avant l'entrelacement (le rendement du code ou encore une base du code dual) ont également été mené par Sicot et al. [SH05a], [SH05b], [BSH06] et [SHB09] dans le cadre d'une transmission bruitée.

Les différentes normes et/ou standards qui utilisent des codes convolutifs dans le but de protéger les données du bruit introduit par le canal de transmission, utilisent des codes convolutifs dits optimaux. En effet, ce sont ces codes optimaux qui ont en réception le plus fort pouvoir de correction. Nos algorithmes de reconstruction seront basés sur les propriétés algébriques des codes convolutifs optimaux que nous avons défini dans le chapitre 1. Dans ce chapitre, nous allons présenter différents algorithmes qui permettront de reconnaître le codeur qui a été utilisé avec la seule connaissance du train binaire issu du démodulateur.

Nous développerons dans un premier temps des algorithmes de reconnaissance aveugle en faisant l'hypothèse que le train binaire reçu est propre, c'est-à-dire non entaché d'erreurs. Lors de communications radio-mobiles, le bruit généré par le canal de transmission est généralement de type

burst, c'est-à-dire que la perturbation des bits se produit par rafales. Ce type de bruit peut-être très gênant lors du décodage. En effet, si la rafale de bruit est très longue, même si le code utilisé a un très bon pouvoir de correction, le décodeur aura des difficultés à retrouver le message émis. En revanche, si ce type de bruit n'est pas idéal pour le décodage, dans le cadre de la reconnaissance aveugle d'un code il offre l'avantage de conserver des plages de données propres. Ces plages propres nous permettraient d'appliquer cet algorithme de reconnaissance aveugle. De plus, en se plaçant assez près de l'émetteur, de tels algorithmes pourraient également être utilisés, puisque dans cette configuration les données seront propres. Mais, dans le but de proposer un algorithme de reconnaissance fonctionnant dans le plus de situations possibles, nous développerons dans la seconde partie de ce chapitre un algorithme d'identification aveugle dans le cadre de l'interception d'une communication bruitée.

2.2 Reconnaissance aveugle d'un code convolutif dans le cas non-bruité

Nous présenterons dans cette partie une méthode permettant d'identifier un code convolutif avec la seule connaissance du train binaire issu du démodulateur. Nous ferons l'hypothèse que le train binaire reçu est propre, c-à-d que le canal de transmission ainsi que les parties modulation/démodulation n'ont engendré aucune erreur dans la séquence de données utilisée par nos algorithmes. Dans un premier temps, nous ferons également l'hypothèse que le train binaire reçu est synchronisé. Cette synchronisation signifie que le premier bit de la trame reçue correspond au premier bit d'un mot de code. Nous présenterons par la suite une méthode qui permettra d'effectuer une synchronisation aveugle afin de lever cette hypothèse.

Nous découperons notre méthode d'identification aveugle en quatre étapes. La première étape consistera dans un premier temps, à détecter la présence d'un code convolutif. Il serait effectivement inutile d'appliquer un algorithme de reconnaissance sur une trame non codée. Cette première étape nous permettra, dans l'hypothèse qu'un code convolutif a été détecté, d'identifier les paramètres du code. L'objectif de la deuxième étape sera de réaliser une synchronisation aveugle. La troisième étape nous permettra d'identifier la matrice du code dual, soit la matrice de parité du code. Et enfin, l'identification d'une matrice génératrice du code sera réalisée en dernière étape.

Dans le premier chapitre, nous avons montré qu'il existait différentes familles de code convolutif, en particulier les codes RSC et NRNSC. Nous avons également montré que ces deux types de code étaient équivalents. En effet il existe toujours une forme RSC de la matrice génératrice d'un codeur NRNSC. De plus, nous savons que la matrice de parité d'un code de forme NRNSC est composée des polynômes générateurs du codeur RSC équivalent. Ainsi, nous traiterons de manière identique les deux types de code. En sortie de l'algorithme de reconnaissance nous obtiendrons le codeur sous sa forme NRNSC. Nous serons en mesure, à partir de cette matrice de retrouver le codeur équivalent sous sa forme RSC, mais sans aucune connaissance a priori sur la nature du code il sera impossible de distinguer si il s'agissait au départ d'un code de forme RSC ou NRNSC.

2.2.1 Détection et identification de paramètres

Nous présentons ici une méthode qui nous permettra de détecter la présence d'un code, et dans l'hypothèse qu'un code ait été détecté, d'identifier ses paramètres. Cette méthode est basée sur le calcul du rang de matrices construites à partir des données issue du démodulateur. Ce critère du rang a été initialement proposé dans [Val00] afin de détecter la présence d'un code en bloc. Cette méthode a ensuite été reprise dans [BG03] afin d'identifier le rendement d'un code en bloc et la taille de l'entrelaceur, lorsque le train binaire était codé et entrelacé. Par la suite, nous avons adapté cette méthode dans [MGB09a] au cas d'un train binaire simplement codé par un code convolutif

afin d'identifier les paramètres du code.

N'ayant aucune erreur de transmission et ayant fait l'hypothèse que nous étions synchronisé, le train binaire reçu correspond simplement à la séquence des mots de code, notée $\mathbf{c}$, telle que :

$$\mathbf{c} = \begin{pmatrix} c_1(t) & \cdots & c_n(t) & \cdots & c_1(t+1) & \cdots & c_n(t+1) & \cdots \end{pmatrix}, \quad \forall t \in \mathbb{N} \tag{2.1}$$

Le principe de la méthode est de réorganiser ce vecteur sous la forme de matrices de taille $M \times l$, que nous noterons R_l. Nous ferons varier le nombre de colonnes l, de 2 à une valeur l_{max} et nous fixerons le nombre de lignes M tel que $M \geq 2.l_{max}$. La figure 2.1 présente un exemple de réorganisation des données interceptées sous la forme d'une matrice de taille $M \times l$, avec $l = 3$. Afin de détecter la présence d'un code et d'identifier ses paramètres, le rang dans $GF(2)$ de chaque matrice R_l sera calculé.

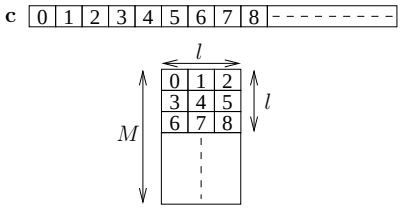


Figure 2.1 — Réorganisation des données interceptées sous forme de matrice

Le rang d'une matrice correspond au nombre de colonnes (ou de lignes) qui sont linéairement indépendantes. Un code en bloc ou convolutif introduit de la redondance au message, cette redondance engendrera une dépendance entre les bits des mots de code. De ce fait, il existera des matrices qui présenteront des déficiences de rang. En revanche, si aucun codage n'a été utilisé et que le train binaire est iid, les bits de la séquence interceptée seront indépendant les uns des autres donc les matrices seront de rang plein. Le critère le plus simple pour détecter la présence d'un code est de décider qu'il n'y a pas eu de codage si toute les matrices sont de rang plein et qu'il y a eu codage si certaines matrices présentent des chutes de rang.

Faisons l'hypothèse que le train binaire n'a pas été codé et calculons le rang des matrices obtenues avec les données interceptées. N'ayant eu aucun codage, aucune redondance n'a été introduite et nous vérifions sur la figure 2.2 que les matrices sont toutes de rang plein.

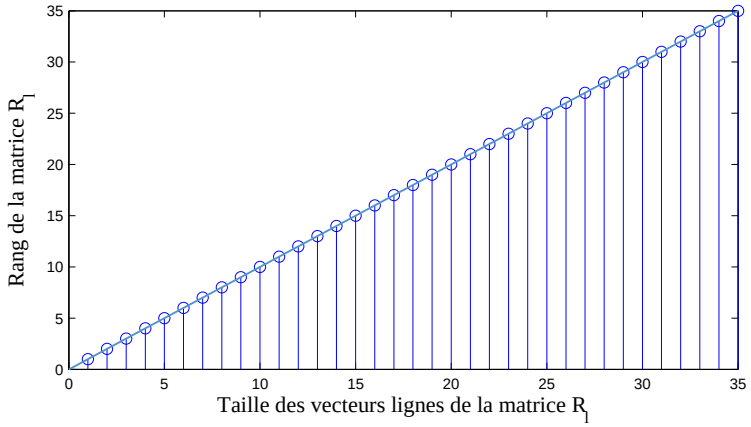


Figure 2.2 — Rang des matrices R_l dans le cas de données non codées

2.2.1.1 Code en bloc

Un code en bloc de rendement $r = k/n$ est décrit par une matrice génératrice de taille $k \times n$. Dans le premier chapitre, nous avons montré qu'un code C pouvait être décrit par plusieurs matrices génératrices équivalentes. Parmi l'ensemble des matrices génératrices, il existe toujours une forme systématique de cette matrice, soit une matrice de la forme :

$$G = \begin{pmatrix} 1 & & & g_{1,k+1} & \cdots & g_{1,n} \\ & \ddots & & \vdots & \cdots & \vdots \\ & & 1 & g_{k,k+1} & \cdots & g_{k,n} \end{pmatrix} \tag{2.2}$$

Chaque mot de code de n bits codé avec une matrice génératrice systématique sera composé des k bits du mot d'information et de $(n-k)$ bits de parité, où ces $(n-k)$ bits de parité dépendent des k bits du mot d'information. Ainsi, la matrice R_l pour $l = n$ sera telle que :

$$R_n = \begin{pmatrix} m_1(t) & \cdots & m_k(t) & c_{k+1}(t) & \cdots & c_n(t) \\ m_1(t+1) & \cdots & m_k(t+1) & c_{k+1}(t+1) & \cdots & c_n(t+1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}, \quad \forall t \in \mathbb{N} \tag{2.3}$$

Les $(n-k)$ bits de parité, noté $\{c_{k+1}(t), \cdots, c_n(t)\}$, dépendent des k bits du mot d'information, noté $\{m_1(t), \cdots, m_k(t)\}$. De par cette dépendance entre les bits de parité et les bits du mot d'information, cette matrice R_n présentera une déficience de rang de $(n-k)$. En revanche, la matrice R_{n+1} sera telle que :

$$R_{n+1} = \begin{pmatrix} m_1(t) & \cdots & m_k(t) & c_{k+1}(t) & \cdots & c_n(t) & m_1(t+1) \\ m_2(t+1) & \cdots & c_{k+1}(t+1) & c_{k+2}(t+1) & \cdots & m_1(t+2) & m_2(t+2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{2.4}$$

Nous pouvons voir qu'il n'existe aucune dépendance entre les colonnes de cette matrice, elle sera donc de rang plein.

Le rang des matrices R_l aura deux comportements distincts en fonction de l :

– Si $l \neq \alpha.n$ $(\forall \alpha \in \mathbb{N})$:

$$rg(R_l) = l \tag{2.5}$$

– Si $l = \alpha.n$ $(\forall \alpha \in \mathbb{N})$:

$$rg(R_l) = l.\frac{k}{n} = l - \frac{l}{n}.(n-k) < l \tag{2.6}$$

De par la présence de matrices de rang déficient, cette méthode nous permet de détecter la présence du code. De plus, d'après l'équation (2.6), nous pourrons également identifier le rendement du code.

Exemple 2.21.

Reprenons l'exemple du code de Hamming de paramètres $n = 7$ et $k = 4$ que nous avons vu dans le chapitre précédent. En prenant ce codeur sous sa forme systématique, un mot de code est constitué des 4 bits du mot d'information et de 3 bits de parité :

$$\mathbf{c}(t) = \begin{pmatrix} m_1(t) & m_2(t) & m_3(t) & m_4(t) & c_5(t) & c_6(t) & c_7(t) \end{pmatrix}$$

avec les 3 bits de parité qui sont tels que :

$$\begin{cases} c_5(t) = m_2(t) + m_3(t) + m_4(t) \\ c_6(t) = m_1(t) + m_3(t) + m_4(t) \\ c_7(t) = m_1(t) + m_2(t) + m_4(t) \end{cases}$$

D'après l'équation (2.6), les matrices de taille $\alpha.n$ construites avec les mots de code présentent une déficience de rang de $\alpha.(n-k)$. Prenons l'exemple de la matrice de taille $n = 7$:

$$R_7 = \begin{pmatrix} m_1(0) & m_2(0) & m_3(0) & m_4(0) & c_5(0) & c_6(0) & c_7(0) \\ m_1(1) & m_2(1) & m_3(1) & m_4(1) & c_5(1) & c_6(1) & c_7(1) \\ m_1(2) & m_2(2) & m_3(2) & m_4(2) & c_5(2) & c_6(2) & c_7(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Cette matrice est composée de 3 colonnes qui sont linéairement dépendantes puisque les bits de parité ($c_5(t)$, $c_6(t)$ et $c_7(t)$) dépendent des bits du mot d'information ($m_1(t)$, $m_2(t)$, $m_3(t)$ et $m_4(t)$). De ce fait, le rang de cette matrice présentera une déficience de 3, soit de $(n-k)$.

Nous avons représenté sur la figure 2.3 le rang des matrices R_l construites avec les mots de code en fonction de l, où $l = 1, \cdots, 35$ et $M = 70$.

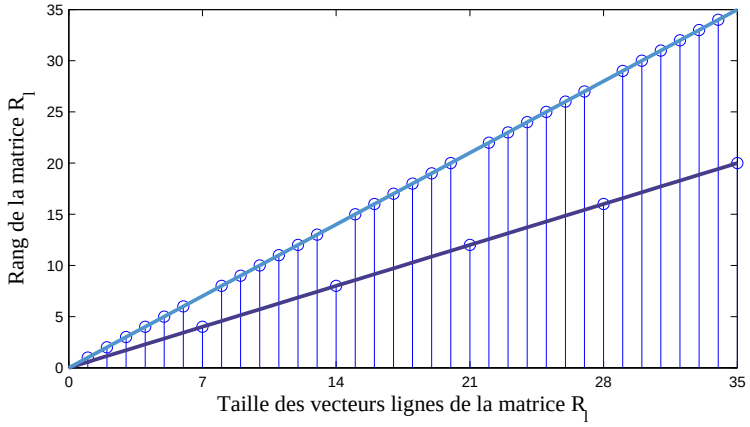


Figure 2.3 — Rang des matrices R_l dans le cas d'un code de Hamming

Nous vérifions que seule les matrices de taille $\alpha.n$ ne sont pas de rang plein et que la droite qui passe par ces chutes de rang vérifie l'équation (2.6) :

$$rg(R_l) = l.\frac{4}{7} = l - l.\frac{3}{7} \qquad \forall l = \alpha.n$$

2.2.1.2 Code convolutif

De la même manière qu'un code en bloc, un code convolutif introduit également de la redondance et cette redondance engendrera également une dépendance entre les bits des mots de code. En revanche, la différence entre ces deux codes est la mémoire générée par un code convolutif. Cette mémoire va influer sur la valeur du rang des matrices, mais également sur la taille de la première matrice de rang déficient. Pour un code en bloc, la première matrice de rang déficient est de taille n. Pour un code convolutif, nous noterons n_a la taille de la première matrice de rang déficient, tel que $n_a = \alpha.n > n$.

Exemple 2.22.

Prenons l'exemple du $C(2,1,3)$ code convolutif. Nous avons vu dans le premier chapitre que l'expression des 2 sorties à l'instant t en fonction des bits d'entrée successifs était :

$$\begin{cases} c_1(t) = m_1(t) + m_1(t-2) \\ c_2(t) = m_1(t) + m_1(t-1) + m_1(t-2) \end{cases}$$

La matrice R_l pour $l = n$ serait telle que :

$$R_n = \begin{pmatrix} c_1(t) & c_2(t) \\ c_1(t+1) & c_2(t+1) \\ c_1(t+2) & c_2(t+2) \\ \vdots & \vdots \end{pmatrix} = \begin{pmatrix} m_1(t) + m_1(t-2) & m_1(t) + m_1(t-1) + m_1(t-2) \\ m_1(t+1) + m_1(t-1) & m_1(t+1) + m_1(t) + m_1(t-1) \\ m_1(t+2) + m_1(t) & m_1(t+2) + m_1(t+1) + m_1(t) \\ \vdots & \vdots \end{pmatrix}$$

Nous pouvons vérifier qu'il n'existe aucune combinaison linéaire entre les colonnes de cette matrice puisque $c_1(t) \neq c_2(t)$, donc elle sera de rang plein.

Dans cette partie, nous présenterons uniquement les résultats de notre méthode d'identification aveugle des paramètres d'un code convolutif. Les origines des expression de cette méthode seront présentées dans la partie suivante. Le rang des matrices R_l aura, comme précédemment, deux comportements différents en fonction de la valeur de l :

– Si $l \neq \alpha.n$ ou $l < n_a$ $(\forall \alpha \in \mathbb{N})$:

$$rg(R_l) = l \tag{2.7}$$

– Si $l = \alpha.n$ et $l \geq n_a$ $(\forall \alpha \in \mathbb{N})$:

$$rg(R_l) = l.\frac{k}{n} + \mu^{\perp} < l \tag{2.8}$$

où $\mu^{\perp}$ correspond à la mémoire du code dual.

Le calcul du rang de ces matrices permet dans un premier temps de vérifier la présence d'un code et d'identifier les paramètres du code détecté. D'après l'équation (2.8), connaissant le rang de deux matrices consécutives de rang déficient, nous serons capable d'identifier les paramètres listés ci-dessous.

1. Identification de n :
   La différence entre la taille de deux matrices de rang déficient correspond à la taille des mots de code :
   $$n = (n_a + (\alpha + 1).n) - (n_a + \alpha.n) \quad \forall \alpha \in \mathbb{N} \tag{2.9}$$
2. Identification de k :
   La différence entre le rang de deux matrices de rang déficient correspond au nombre d'entrée du codeur :
   $$k = rg(R_{n_a+(\alpha+1).n}) - rg(R_{n_a+\alpha.n}) \quad \forall \alpha \in \mathbb{N} \tag{2.10}$$
3. Identification de $\mu^{\perp}$:
   La mémoire du code dual est identifiée à partir de la connaissance du rendement du code et du rang d'une matrice de rang déficient :
   $$\mu^{\perp} = rg(R_{n_a+\alpha.n}) - (n_a + \alpha.n).\frac{k}{n} \quad \forall \alpha \in \mathbb{N} \tag{2.11}$$
4. Identification de μ :
   Le lien entre la mémoire du code et celle du code dual est :
   $$\mu^{\perp} = \sum_{i=1}^{k} \mu_i \tag{2.12}$$
   où μ_i représente la mémoire de la i-ème entrée du codeur. La mémoire du codeur μ qui correspond à la mémoire maximale des entrées est telle que :
   $$\mu = \left\lceil \frac{\mu^{\perp}}{k} \right\rceil \tag{2.13}$$

où $\lceil x \rceil$ est l'arrondi supérieur de x.

Nous venons de présenter une méthode qui permet de détecter la présence d'un code et d'en identifier ses paramètres. Cette même méthode nous permettra également de différencier un code en bloc d'un code convolutif. D'après les équations (2.6) et (2.8), le rang des matrices R_l pour un code en bloc et un code convolutif est :

– Code en bloc :

$$rg(R_l) = l.\frac{k}{n} < l \quad \forall l = \alpha.n \tag{2.14}$$

– Code convolutif :

$$rg(R_l) = l.\frac{k}{n} + \mu^{\perp} < l \quad \forall l = \alpha.n \text{ et } l \geq n_a \tag{2.15}$$

D'après ces deux équations, nous pourrons décider que le code identifié est un code convolutif si la mémoire, $\mu^{\perp}$, identifiée est non-nulle. En revanche, si cette mémoire est nulle, nous déciderons que le code identifié est un code en bloc.

Exemple 2.23.

Suite de l'exemple 2.22.

Prenons l'exemple du $C(2,1,3)$ code de matrice génératrice $G(D) = [1+D^2 \;\; 1+D+D^2]$ et de matrice de parité $H(D) = [1+D+D^2 \;\; 1+D^2]$. La figure 2.4 représente le rang des matrices obtenues à partir de la séquence des mots de code pour $l = 1, \cdots, 35$ et $M = 70$.

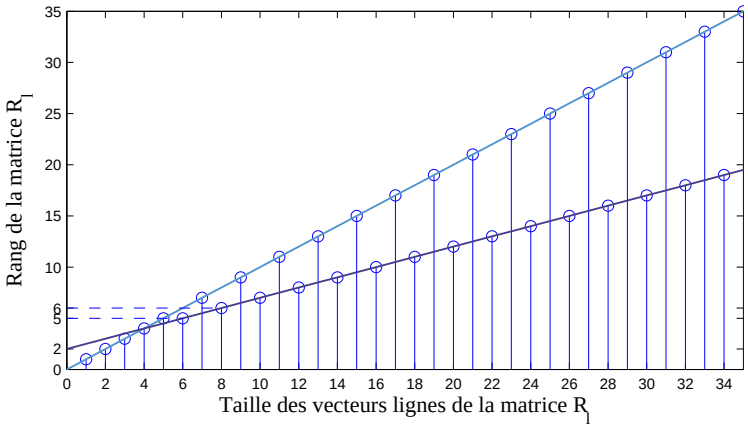


Figure 2.4 — Rang des matrices R_l du $C(2,1,3)$ code

Pour ce codeur, la première matrice de rang déficient est de taille $n_a = \alpha.n = 6$ et nous vérifions que les matrices de taille $l = \alpha.n \geq 6$ présentent des déficiences de rang.

D'après les équations (2.9), (2.10), (2.11) et (2.13), en prenant 2 matrices de rang déficient consécutives, par exemple R_6 et R_8, nous obtenons :

$$\begin{cases} n = 8 - 6 = 2 \\ k = rg(R_8) - rg(R_6) = 1 \\ \mu^{\perp} = rg(R_6) - 6.\frac{1}{2} = 2 \\ \mu = 2 \end{cases}$$

La mémoire du code dual étant non nulle, nous pouvons en déduire que le code est un code convolutif et les paramètres identifiés correspondent bien aux paramètres du $C(2,1,3)$ code convolutif.

Cette méthode publiée dans [GMB08, MGB09a] permet d'identifier l'ensemble des paramètres d'un code convolutif. Nous allons nous intéresser plus en détails sur les propriétés du rang des matrices et en particuliers sur la taille de la première matrice qui est de rang déficient, soit la valeur de n_a.

2.2.2 Étude du rang des matrices R_l

D'après l'équation (2.8), nous savons que les matrices de taille $\alpha.n$, avec $\alpha \in \mathbb{N}$, présentent des déficiences de rang telles que :

$$rg(R_{\alpha.n}) = \alpha.n.\frac{k}{n} + \mu^{\perp} < \alpha.n \quad \text{pour } \alpha.n \geq n_a \tag{2.16}$$

soit :

$$\alpha.n > n.\frac{\mu^{\perp}}{n-k} \tag{2.17}$$

Par conséquent, la première matrice de rang déficient sera de taille :

$$n_a = n.\left\lfloor \frac{\mu^{\perp}}{n-k} + 1 \right\rfloor \tag{2.18}$$

où $\lfloor . \rfloor$ désigne la partie entière.

2.2.2.1 Code de rendement $(n-1)/n$

Pour un code de rendement $(n-1)/n$, la matrice génératrice est composée de $(n-1) \times n$ polynômes générateurs et la matrice de parité est un vecteur composé de n polynômes générateurs :

$$G(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,n-1}(D) & g_{1,n}(D) \\ \vdots & \cdots & \vdots & \vdots \\ g_{n-1,1}(D) & \cdots & g_{n-1,n-1}(D) & g_{n-1,n}(D) \end{bmatrix}, \quad H(D) = \begin{bmatrix} h_{1,1}(D) & \cdots & h_{1,n-1}(D) & h_0(D) \end{bmatrix} \tag{2.19}$$

En ne tenant pas compte de la partie remplissage des registres, la matrice de parité sous sa forme binaire est :

$$H = \begin{pmatrix} H_{\mu^{\perp}} & H_{\mu^{\perp}-1} & \cdots & H_1 & H_0 & & \\ & H_{\mu^{\perp}} & H_{\mu^{\perp}-1} & \cdots & H_1 & H_0 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \tag{2.20}$$

où les sous-matrices de parité H_i, $\forall i = 0, \cdots, \mu^{\perp}$, sont de simples vecteurs de taille n définies par :

$$H_i = \begin{pmatrix} h_{1,1}(i) & \cdots & h_{1,n-1}(i) & h_0(i) \end{pmatrix} \tag{2.21}$$

La matrice de parité H (2.20) est composée du même vecteur ligne qui est simplement décalé de n colonnes entre chaque ligne. Nous noterons $\mathbf{h}$ ce vecteur qui correspond à une relation de parité et qui est défini par :

$$\mathbf{h} = \begin{pmatrix} H_{\mu^{\perp}} & H_{\mu^{\perp}-1} & \cdots & H_0 \end{pmatrix} \tag{2.22}$$

Cette relation de parité est composée des $(\mu^{\perp}+1)$ sous-matrices de parité du code qui sont chacune de taille $1 \times n$. Cette relation de parité $\mathbf{h}$ est donc un vecteur de taille $n.(\mu^{\perp}+1)$. D'après la définition de la valeur de n_a (2.18), pour un code de rendement $(n-1)/n$, la taille de la première matrice présentant une chute de rang correspond également à la taille de la relation de parité du code.

Dans la partie 1.4.4 du chapitre précédent, nous avons vu que les polynômes de la matrice de parité étaient tels que :

$$\begin{cases} h_{i,j}(D) = f_{j,i+k}(D) & \forall i = 1, \cdots, n \text{ et } \forall j = 1, \cdots, k \\ h_0(D) = f_{1,1}(D) \end{cases} \tag{2.23}$$

où les polynômes $f_{l,m}(D)$ correspondent aux polynômes générateurs du codeur sous sa forme RSC.

Nous avons également montré qu'un codeur RSC était réalisable si le bit de poids faible du polynôme de feedback ($f_{1,1}(D)$) était égal à "1", soit $f_{1,1}(0) = 1$. De par cette propriété, le polynôme générateur $h_0(D)$ est de la forme :

$$h_0(D) = 1 + h_0(1).D + \cdots + h_0(\mu^{\perp} - 1).D^{\mu^{\perp}-1} + h_0(\mu^{\perp}).D^{\mu^{\perp}} \tag{2.24}$$

D'après l'équation (2.22), le dernier élément du vecteur **h** correspond au bit $h_0(0)$. Ainsi, la relation de parité **h** est un vecteur de taille n_a qui a pour dernier élément un bit à "1". Cette relation de parité est donc de degré $(n_a - 1)$.

Nous savons que les matrices R_l, pour $l < n_a$, sont de rang plein et que la matrice R_{n_a} est la première matrice de rang déficient. De ce fait, la relation de parité **h** du code qui est de degré $(n_a - 1)$ correspond à la relation de parité de plus petit degré du code.

Exemple 2.24.

Suite de l'exemple 2.23.

D'après les paramètres du code et l'équation (2.18), nous obtenons $n_a = 6$. La matrice R_l pour $l = n_a$ est telle que :

$$R_6 = \begin{pmatrix} c_1(0) & c_2(0) & c_1(1) & c_2(1) & c_1(2) & c_2(2) \\ c_1(3) & c_2(3) & c_1(4) & c_2(4) & c_1(5) & c_2(5) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

En notant une ligne de cette matrice :

$$\begin{pmatrix} c_1(i) & c_2(i) & c_1(i+1) & c_2(i+1) & c_1(i+2) & c_2(i+2) \end{pmatrix}$$

avec $i = [0 : 3 : M]$, on remarque qu'il existe une combinaison linéaire entre les colonnes de cette matrice. En effet, en additionnant les colonnes (1,2,3,5,6) de cette matrice, nous obtenons :

$$c_1(i) + c_2(i) + c_1(i+1) + c_1(i+2) + c_2(i+2) = 0$$

Cette matrice présentera une déficience de rang égale à 1. Nous pouvons vérifier sur la figure 2.4 de l'exemple 2.23 que la matrice R_6 présente une déficience de rang de 1 puisque $rg(R_6) = 5$.

2.2.2.2 Code de rendement k/n

Dans le cas général, pour un codeur de rendement k/n, la première matrice de rang déficient est de taille n_a :

$$n_a = n.\left\lfloor \frac{\mu^{\perp}}{n-k} + 1 \right\rfloor \tag{2.25}$$

Nous noterons $Q(l)$, la déficience de rang de la matrice R_l qui est définie par :

$$Q(l) = l - rg(R_l) = l - l.\frac{k}{n} - \mu^{\perp} \tag{2.26}$$

D'après la valeur de n_a, la matrice de taille n_a aura une déficience de rang égale à :

$$Q(n_a) = (n-k) - \Big(\mu^{\perp} \bmod (n-k)\Big) \tag{2.27}$$

où $\big(\mu^{\perp} \bmod (n-k)\big)$ représente le reste dans $(0, \cdots, n-k-1)$ de la division euclidienne de deux nombres entiers : $\frac{\mu^{\perp}}{n-k}$.

La première matrice de rang déficient présentera une déficience de rang qui sera comprise entre

1 et $(n-k)$, soit :

$$1 \leq Q(n_a) \leq (n-k) \tag{2.28}$$

Pour un code de rendement $(n-1)/n$, la matrice R_{n_a} aura une déficience de rang de 1. En revanche pour un code de rendement k/n, cette déficience dépendra du rendement et de la mémoire du code dual.

D'après l'équation (2.26), la différence entre deux déficiences de rang est :

$$Q(n_a + (\alpha+1).n) - Q(n_a + \alpha.n) = n - k \tag{2.29}$$

Exemple 2.25.

Prenons l'exemple d'un code de rendement 1/3 avec $K = 3, \cdots, 10$. Le tableau 2.1 représente la valeur de n_a et la déficience de rang de la matrice R_{n_a} pour ces différentes valeurs de K.

K	n_a	$Q(n_a)$
3	6	2
4	6	1
5	9	2
6	9	1
7	12	2
8	12	1
9	15	2
10	15	1

Tableau 2.1 — Étude de la première matrice de rang déficient pour un code de rendement 1/3

Prenons maintenant le $C(3,1,4)$ code de matrice génératrice $G = \begin{pmatrix} 13 & 15 & 17 \end{pmatrix}$ et calculons le rang des matrices R_l pour $l = 1, \cdots, 35$ et $M = 70$. Le rang de ces matrices est représenté sur la figure 2.5.

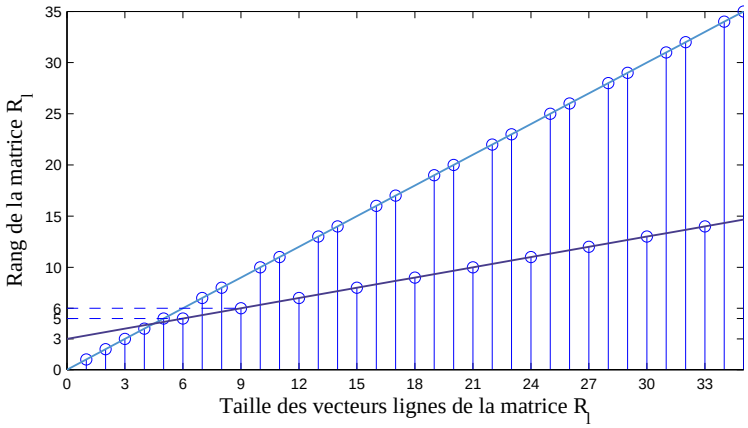


Figure 2.5 — Rang des matrices R_l du $C(3,1,4)$ code

Nous vérifions que la première matrice de rang déficient est de taille 6 et que cette déficience de rang est de 1, comme indiqué dans le tableau 2.1.

D'après les équations (2.9), (2.10), (2.11) et (2.13), en prenant le rang des matrices R_6 et R_9,

nous obtenons les paramètres suivants :

$$\begin{cases} n = 9 - 6 = 3 \\ k = rg(R_9) - rg(R_6) = 1 \\ \mu^{\perp} = rg(R_6) - 6.\frac{1}{3} = 3 \\ \mu = 3 \end{cases}$$

Nous vérifions que les paramètres identifiés correspondent aux paramètres du $C(3, 1, 4)$ code.

2.2.2.3 Quelques cas particuliers

Parmi l'ensemble des codes convolutifs, il existe quelques cas particuliers tels que les codes de rendement k/n avec $n = \alpha.2$ et $k \neq n - 1$. Pour ces codes, les matrices R_l pour $l = \alpha.n$ présentent également une déficience de rang mais il existe également des matrices de taille $l \neq \alpha.n$ qui présenteront des chutes de rang. Mais, ces dernières chutes de rang seront plus faible que celles obtenues pour les matrices de taille $l = \alpha.n$. En effet, pour $l = \alpha.n$ les chutes de rang observées seront "maximales". Par conséquent, il sera possible de distinguer les chutes de rang maximales obtenues pour $l = \alpha.n$ et d'identifier les paramètres du code en utilisant uniquement ces chutes de rang maximales.

Exemple 2.26.

Prenons l'exemple d'un code de rendement 1/4 et de longueur de contrainte égale à 6. La figure 2.6 représente le rang des matrices R_l avec $l = 1, \cdots, 35$ et $M = 70$.

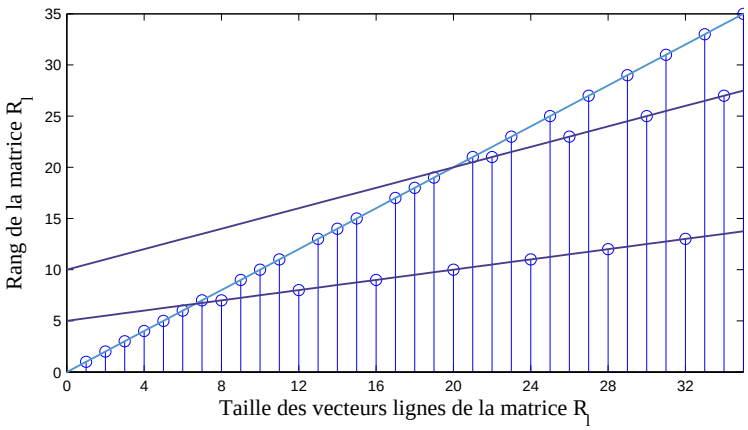


Figure 2.6 — Rang des matrices R_l du $C(4, 1, 6)$ code

Sur cette figure, nous remarquons la présence de chutes de rang maximales pour les matrices R_l avec $l = [8, 12, 16, 20, 24, 28, 32]$. D'après les paramètres du code, la taille de la première matrice de rang déficient n_a est :

$$n_a = n.\left\lfloor \frac{\mu^{\perp}}{n-k} + 1 \right\rfloor = 8$$

Cette valeur correspond bien à la première matrice présentant une déficience de rang observée sur la figure 2.6. En revanche, sur cette même figure, nous remarquons que les matrices de taille $l = [22, 26, 30, 34]$ sont également de rang déficient. Mais les déficiences de rang pour ces matrices sont plus petite que celles obtenues avec les matrices de taille $l = \alpha.n \geq n_a$. En utilisant uniquement les matrices présentant des chutes de rang maximales, nous pourrons identifier les

paramètres du code. Prenons les matrices R_8 et R_{12} :

$$\begin{cases} n = 12 - 8 = 4 \\ k = rg(R_{12}) - rg(R_8) = 1 \\ \mu^{\perp} = rg(R_8) - 8.\frac{1}{4} = 5 \\ \mu = 5 \end{cases}$$

Nous pouvons vérifier que les paramètres identifiés correspondent à ceux du $C(4, 1, 6)$ code.

Le calcul du rang des matrices construites avec les mots de code nous a permis de développer une méthode permettant d'identifier en aveugle les paramètres d'un code convolutif. Jusqu'ici, nous avons fait l'hypothèse que nous étions synchronisé, c-à-d que la trame reçue commençait par le premier bit d'un mot de code. Or, lors de l'interception d'une trame il faudra réaliser une synchronisation aveugle afin de trouver le début d'un mot de code.

2.2.3 Synchronisation aveugle

Nous ferons maintenant l'hypothèse que la trame reçue n'est pas synchronisée. Nous devrons effectuer une synchronisation aveugle qui consistera à trouver le début d'un mot de code, soit le premier bit d'un mot de code $c_1(t)$. La méthode que nous venons de présenter pour identifier les paramètres d'un code convolutif restera efficace pour identifier le rendement d'un code, même si la trame n'est pas synchronisée. En revanche, sans synchronisation, il sera impossible d'identifier la mémoire du code et du code dual. En effet, cette désynchronisation implique que la première matrice présentant des déficiences de rang ne sera pas nécessairement de taille n_a. Par contre, le pas entre les matrices de rang déficient sera comme précédemment de n. Nous noterons n_{a_1} la taille de la première de rang déficient obtenue avec la trame non synchronisée. A partir de la matrice de taille n_{a_1}, notée $R_{n_{a_1}}$, nous proposons une méthode afin d'identifier le début d'un mot de code. Cette méthode a été initialement proposée dans [BG03] afin d'effectuer une synchronisation aveugle sur une trame codée par un code en bloc et entrelacée.

Nous noterons $\tilde{\mathbf{c}}$ la séquence interceptée qui est une version décalée de t_0 bits de la séquence des mots de code $\mathbf{c}$. Les matrices R_l seront construites avec cette version décalée des mots de code et le rang dans $GF(2)$ de ces matrices sera calculé. Connaissant deux matrices de rang déficient, nous serons en mesure d'identifier la taille des mots de code et la taille de la matrice $R_{n_{a_1}}$.

Afin d'identifier le paramètre de synchronisation, que nous noterons d, nous définirons les matrices $R_{n_{a_1},d}$. Ces matrices qui sont de taille $M \times n_{a_1}$ sont construites avec la trame $\tilde{\mathbf{c}}$ décalée de d bits. Nous ferons varier le paramètre d de 0 à $n-1$ et nous calculerons le rang normalisé de ces n matrices. Nous noterons $\rho(n_{a_1}, d)$ le rang normalisé de la matrice $R_{n_{a_1},d}$ définie par :

$$\rho(n_{a_1}, d) = \frac{rg(R_{n_{a_1},d})}{n_{a_1}} \tag{2.30}$$

Nous verrons apparaître une chute de rang maximal lorsque le bon nombre de bits aura été supprimé. Une fois le paramètre d identifié, nous supprimerons les d premiers bits de la trame $\tilde{\mathbf{c}}$ et nous appliquerons la méthode précédente sur cette nouvelle trame afin d'identifier l'ensemble des paramètres du code.

Exemple 2.27.

Prenons l'exemple d'un code convolutif de paramètres $C(3, 1, 4)$. La séquence des mots de code $\mathbf{c}$ est :

$$\mathbf{c} = \begin{pmatrix} c_1(0) & c_2(0) & c_3(0) & c_1(1) & c_2(1) & c_3(1) & \cdots \end{pmatrix}$$

En introduisant une désynchronisation de paramètre $t_0 = 4$, la trame reçue, notée $\tilde{\mathbf{c}}$, est telle

que :

$$\tilde{\mathbf{c}} = \begin{pmatrix} c_2(1) & c_3(1) & c_1(2) & c_2(2) & c_3(2) & \cdots \end{pmatrix}$$

La figure 2.10 représente le rang des matrices R_l construites avec les données $\tilde{\mathbf{c}}$.

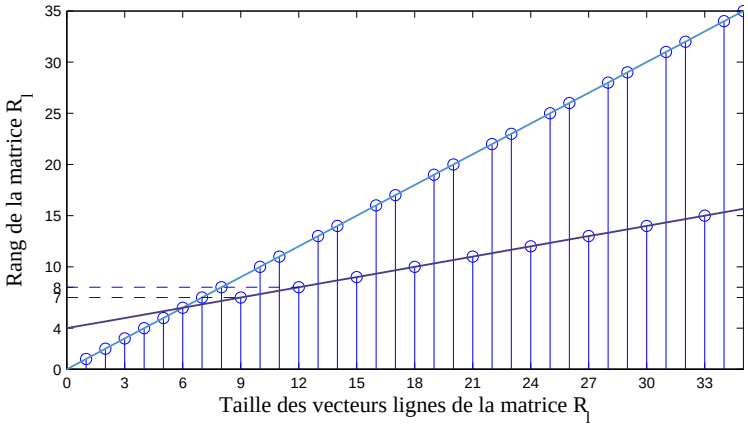


Figure 2.7 — Rang des matrices R_l du $C(3,1,4)$ code avec un décalage de $t_0 = 4$

Sur la figure 2.5 de l'exemple 2.25, nous avons vu que pour ce codeur la première matrice de rang déficient était de taille 6 et que la droite passant par les chutes de rang coupait l'axe des ordonnées en la valeur de 3. De par la désynchronisation, nous remarquons sur la figure 2.10 que la première matrice de rang déficient est une matrice de taille $n_{a_1} = 9$ et que la droite coupe l'axe des ordonnées en la valeur de 4. En appliquant la méthode d'identification aveugle des paramètres, on obtiendrait les paramètres suivants :

$$\begin{cases} n = 3 \\ k = 1 \\ \mu^{\perp} = 4 \end{cases}$$

Le rendement du code identifié est correct mais de par la désynchronisation, la mémoire du code dual ne correspond pas à celle du code initialement utilisé. Nous allons réaliser une synchronisation aveugle afin de pouvoir identifier en aveugle les bons paramètres du code. Sur la figure 2.8, le rang normalisé (2.30) des matrices de taille $n_{a_1} = 9$ est représenté pour des valeurs de désynchronisation variant de 0 à $n - 1$.

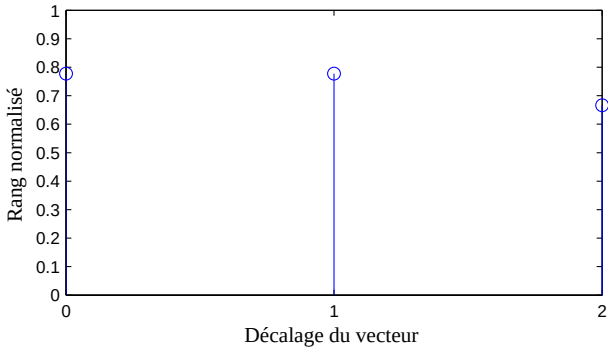


Figure 2.8 — Synchronisation aveugle du $C(3,1,4)$ code

Nous remarquons sur cette figure que pour $d = 2$, soit en supprimant les 2 premiers bits de la trame $\tilde{\mathbf{c}}$, nous obtenons une chute de rang maximale. En effet, si nous supprimons les 2 premiers

bits de $\tilde{\mathbf{c}}$, la trame obtenue est :

$$\begin{pmatrix} c_1(2) & c_2(2) & c_3(2) & c_1(3) & c_2(3) & c_3(3) & \cdots \end{pmatrix}$$

Nous pouvons vérifier que cette trame débute par le premier bit d'un mot de code. En recalculant le rang des matrices R_l avec cette trame synchronisée, nous obtiendrons le même résultat qui était présenté dans l'exemple 2.25.

Maintenant que la trame est synchronisée et que les paramètres du code ont été identifiés, nous allons pouvoir nous intéresser à l'identification de la matrice génératrice du code dual.

2.2.4 Identification du code dual

Les paramètres du code ayant été identifiés, à partir de la trame synchronisée nous proposerons dans cette partie une méthode permettant d'identifier la matrice de parité du code. Nous verrons tout d'abord cette méthode dans le cas des codeurs de rendement $(n-1)/n$, puis nous la généraliserons aux codeurs de rendement k/n.

2.2.4.1 Code de rendement $(n-1)/n$

D'après les propriétés des codes convolutifs (voir chapitre 1), si $H(D)$ est une matrice de parité du code C, une séquence de mots de code, $c(D)$, appartient au code C si et seulement si :

$$c(D).H^T(D) = 0 \tag{2.31}$$

Avec $c(D)$ la séquence des mots de code et $H(D)$ la matrice de parité composée de n polynômes :

$$\begin{bmatrix} c_1(D) & \cdots & c_{n-1}(D) & c_n(D) \end{bmatrix} . \begin{bmatrix} h_{1,1}(D) \\ \vdots \\ h_{1,n-1}(D) \\ h_0(D) \end{bmatrix} = 0 \tag{2.32}$$

Nous obtenons une équation de la forme suivante :

$$\sum_{i=1}^{n-1} c_i(D).h_{1,i}(D) + c_n(D).h_0(D) = 0 \tag{2.33}$$

En pratique, les données traitées ne sont pas sous forme polynomiale mais sous forme binaire. La matrice de parité sous sa forme binaire, notée H (2.20), est une matrice composée du même vecteur ligne $\mathbf{h}$ (2.22). Avec $\mathbf{h}$ la relation de parité du code de plus petit degré qui est définie par :

$$\mathbf{h} = \begin{pmatrix} h_{1,1}(\mu^\perp), \cdots, h_{1,n-1}(\mu^\perp), h_0(\mu^\perp), \cdots, h_{1,1}(1), \cdots, h_{1,n-1}(1), h_0(1), h_{1,1}(0), \cdots, h_{1,n-1}(0), h_0(0) \end{pmatrix} \tag{2.34}$$

En notant R_{n_a} la matrice de taille $M \times n_a$ construite avec les mots de code, nous obtenons le système homogène suivant :

$$R_{n_a}.\mathbf{h}^T = \mathbf{0} \tag{2.35}$$

Dans le but d'identifier la relation de parité du code, il sera nécessaire de construire la matrice R_{n_a} et de résoudre le système homogène précédent. Afin d'obtenir une solution non triviale lors de la résolution de ce système, il est nécessaire que le rang de la matrice R_{n_a} soit inférieur ou égal à $n_a - 1$. Par conséquent, pour que le problème ne soit pas sous dimensionné, il est nécessaire de disposer d'une trame composée d'au minimum n_a^2 bits.

En réorganisant les bits des mots de code différemment, seuls $n.(n_a + \mu^{\perp})$ bits seront nécessaire pour construire un système homogène. Nous noterons B une matrice de taille $n_a \times n_a$ telle que :

$$B = \begin{bmatrix} B_0 & B_1 & \cdots & B_{\mu^{\perp}-1} & B_{\mu^{\perp}} \end{bmatrix} \tag{2.36}$$

où les $\mu^{\perp} + 1$ matrices B_j sont de taille $n_a \times n$ et définies par :

$$B_j = \begin{pmatrix} c_1(j) & \cdots & c_n(j) \\ c_1(j+1) & \cdots & c_n(j+1) \\ \vdots & \cdots & \vdots \\ c_1(j+n_a-1) & \cdots & c_n(j+n_a-1) \end{pmatrix}, \quad \forall j = d, \cdots, d+\mu^{\perp} \tag{2.37}$$

où d correspond au paramètre de synchronisation obtenu avec la méthode de synchronisation aveugle. Pour identifier la relation de parité $\mathbf{h}$, il suffit de construire la matrice B et de résoudre le système homogène suivant :

$$B.\mathbf{h}^T = \mathbf{0} \tag{2.38}$$

Pour un code de rendement $(n-1)/n$, nous avons montré que la matrice de taille $l = n_a$ présentait une déficience de rang égale à 1. De ce fait, le nombre de vecteurs solutions du système (2.38) sera peu élevé. Nous savons également que le dernier élément de la relation de parité que nous voulons identifier est un élément à "1" puisqu'il correspond au bit $h_0(0)$. Nous pouvons donc lors de la résolution du système faire l'hypothèse que le vecteur solution que nous recherchons se termine par un bit à "1".

Exemple 2.28.

Prenons la suite de l'exemple 2.22 du $C(2,1,3)$ code. La matrice de parité sous sa forme polynomiale est :

$$H(D) = \begin{bmatrix} 1 + D + D^2 & 1 + D^2 \end{bmatrix}$$

Nous avons vu que la première matrice de rang déficient était de taille $n_a = 6$. La matrice B (2.36) de taille $n_a \times n_a$ construite avec les mots de code est telle que :

$$B = \begin{pmatrix} c_1(t) & c_2(t) & c_1(t+1) & c_2(t+1) & c_1(t+2) & c_2(t+2) \\ c_1(t+1) & c_2(t+1) & c_1(t+2) & c_2(t+2) & c_1(t+3) & c_2(t+3) \\ c_1(t+2) & c_2(t+2) & c_1(t+3) & c_2(t+3) & c_1(t+4) & c_2(t+4) \\ c_1(t+3) & c_2(t+3) & c_1(t+4) & c_2(t+4) & c_1(t+5) & c_2(t+5) \\ c_1(t+4) & c_2(t+4) & c_1(t+5) & c_2(t+5) & c_1(t+6) & c_2(t+6) \\ c_1(t+5) & c_2(t+5) & c_1(t+6) & c_2(t+6) & c_1(t+7) & c_2(t+7) \end{pmatrix}, \quad \forall t \in \mathbb{N}$$

En résolvant le système (2.38), nous obtenons un unique vecteur solution :

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

Notons $B^{(i)}$ une ligne de la matrice B telle que :

$$B^{(i)} = \begin{pmatrix} c_1(i) & c_2(i) & c_1(i+1) & c_2(i+1) & c_1(i+2) & c_2(i+2) \end{pmatrix} \quad \forall i = t, \cdots, t+n_a-1$$

Nous avons montré que les bits des mots de code étaient tels que :

$$\begin{cases} c_1(i) = m_1(i) + m_1(i-2) \\ c_2(i) = m_1(i) + m_1(i-1) + m_1(i-2) \end{cases}$$

Alors, d'après l'expression de ces mots de code, nous pouvons vérifier que le vecteur $\mathbf{h}$ identifié

correspond à une relation de parité du code puisque :

$$B^{(i)}.\mathbf{h}^T = c_1(i) + c_2(i) + c_1(i+1) + c_1(i+2) + c_2(i+2) = 0$$

2.2.4.2 Code de rendement k/n

Pour un code de rendement k/n, la matrice de parité est une $(n-k) \times n$ matrice définie par :

$$H(D) = \begin{bmatrix} h_{1,1}(D) & \cdots & h_{1,k}(D) & h_0(D) & & \\ \vdots & \cdots & \vdots & & \ddots & \\ h_{n-k,1}(D) & \cdots & h_{n-k,k}(D) & & & h_0(D) \end{bmatrix} \tag{2.39}$$

Comme précédemment, nous savons qu'une séquence de mots de code $c(D)$ appartient au code C si :

$$c(D).H^T(D) = \begin{bmatrix} c_1(D) & \cdots & c_k(D) & c_{k+1}(D) & \cdots & c_n(D) \end{bmatrix} . \begin{bmatrix} h_{1,1}(D) & \cdots & h_{n-k,1}(D) \\ \vdots & \cdots & \vdots \\ h_{1,k}(D) & \cdots & h_{n-k,k}(D) \\ h_0(D) & & \\ & \ddots & \\ & & h_0(D) \end{bmatrix} = 0 \tag{2.40}$$

La matrice de parité sous sa forme binaire est composée de $(n-k)$ relations de parité. Notons H_i les sous matrices de parité qui sont de taille $(n-k) \times n$:

$$H_i = \begin{pmatrix} h_{1,1}(i) & \cdots & h_{1,k}(i) & h_0(i) & & \\ \vdots & \cdots & \vdots & & \ddots & \\ h_{n-k,1}(i) & \cdots & h_{n-k,k}(i) & & & h_0(i) \end{pmatrix}, \quad \forall i = 0, \cdots, \mu^{\perp} \tag{2.41}$$

Nous définirons $H_i^{(j)}$, un vecteur de taille n qui correspond à la j-ème ligne de la matrice H_i :

$$H_i^{(j)} = \begin{pmatrix} h_{j,1}(i) & \cdots & h_{j,k}(i) & \mathbf{0}_{j-1} & h_0(i) & \mathbf{0}_{n-k-j} \end{pmatrix} \tag{2.42}$$

où $\mathbf{0}_l$ est un vecteur nul de taille l. Alors, les $(n-k)$ relations de parité du code, noté $\mathbf{h}_j$ ($\forall j = 1, \cdots, n-k$), sont définies par :

$$\mathbf{h}_j = \begin{pmatrix} H_{\mu^{\perp}}^{(j)} & H_{\mu^{\perp}-1}^{(j)} & \cdots & H_1^{(j)} & H_0^{(j)} \end{pmatrix} \tag{2.43}$$

Afin d'identifier ces $(n-k)$ relations de parité qui sont des vecteurs de taille $n.(\mu^{\perp}+1)$, nous proposerons deux méthodes.

Méthode 1

Précisons tout d'abord que ces $(n-k)$ relations de parité sont des vecteurs de taille $n.(\mu^{\perp}+1)$ composés de $(n-k-1).(\mu^{\perp}+1)$ bits à "0" et qu'elles sont chacune composées des bits du polynôme $h_0(D)$. Nous noterons n_{a_1} la taille de ces relations de parité (2.43), soit :

$$n_{a_1} = n.(\mu^{\perp}+1) \tag{2.44}$$

Pour identifier ces relations de parité, il est nécessaire de construire une matrice de taille $n_{a_1} \times n_{a_1}$. Comme précédemment, nous pouvons soit construire directement la matrice $R_{n_{a_1}}$, soit construire la matrice B. Pour la matrice $R_{n_{a_1}}$, le nombre minimum de bits nécessaires est de $n_{a_1}^2$ alors que seuls $n.(n_{a_1} + \mu^{\perp})$ bits sont nécessaires pour la matrice B. Cette dernière est composée de $(\mu^{\perp} + 1)$ matrices B_j :

$$B = \begin{bmatrix} B_0 & B_1 & \cdots & B_{\mu^{\perp}-1} & B_{\mu^{\perp}} \end{bmatrix} \tag{2.45}$$

avec les matrices B_j de taille $n_{a_1} \times n$:

$$B_j = \begin{pmatrix} c_1(j) & \cdots & c_n(j) \\ c_1(j+1) & \cdots & c_n(j+1) \\ \vdots & \cdots & \vdots \\ c_1(j+n_{a_1}-1) & \cdots & c_n(j+n_{a_1}-1) \end{pmatrix}, \quad \forall j = d, \cdots, d+\mu^{\perp} \tag{2.46}$$

Les $(n-k)$ relations de parité du code peuvent être identifiées en résolvant le système suivant :

$$B.\mathbf{h}^T = \mathbf{0} \tag{2.47}$$

D'après l'équation (2.26), la déficience de rang de la matrice $R_{n_{a_1}}$, qui est équivalente à celle de la matrice B, est :

$$Q(n_{a_1}) = (\mu^{\perp} + 1).(n-k) - \mu^{\perp} \tag{2.48}$$

Pour un codeur dont le nombre d'entrées est inférieur à $(n-1)$, la déficience de rang sera donc strictement supérieure à $(n-k)$. Il est évident, que plus la déficience de rang sera grande plus le nombre de vecteurs solutions sera élevé. Mais, nous savons que chaque vecteur solution doit être composé d'au minimum $(n-k-1).(\mu^{\perp}+1)$ bits à "0" et des mêmes bits du polynôme $h_0(D)$. Ainsi, il sera possible de faire un tri parmi l'ensemble des vecteurs candidats et d'obtenir au moins un groupe de $(n-k)$ vecteurs solutions qui vérifie ces propriétés.

Lors de la résolution du système, la phase la plus coûteuse en terme de complexité reste la triangularisation de la matrice B à l'aide du pivot de Gauss. Le coût de cette phase est de l'ordre de $\mathcal{O}(\big(n.(\mu^{\perp}+1)\big)^3)$. Nous allons proposer une seconde méthode qui nous permettra de réduire ce coût.

Méthode 2

Afin de simplifier le système à résoudre, nous allons découper le système de l'équation (2.40) en $(n-k)$ sous systèmes :

$$\begin{bmatrix} c_1(D) & \cdots & c_k(D) & c_{k+s}(D) \end{bmatrix} . \begin{bmatrix} h_{s,1}(D) \\ \vdots \\ h_{s,k}(D) \\ h_0(D) \end{bmatrix} = \sum_{i=1}^{k} c_i(D).h_{s,i}(D) + c_{k+s}(D).h_0(D) = 0 \tag{2.49}$$

où $s = 1, \cdots, n-k$.

Nous remarquons que le polynôme $h_0(D)$ est commun au $(n-k)$ systèmes, ce qui signifie que lors de la résolution des systèmes nous devrons vérifier que l'on obtient ce même polynôme pour les $(n-k)$ systèmes. Sous forme binaire, ce système peut s'écrire en posant $(n-k)$ matrice $B^{(s)}$ $(\forall s = 1, \cdots, n-k)$ de taille $(k+1).(\mu^{\perp}+1) \times (k+1).(\mu^{\perp}+1)$. Nous noterons $B_j^{(s)}$ les sous-matrices

de taille $(k+1).(\mu^{\perp}+1) \times (k+1)$:

$$B_j^{(s)} = \begin{pmatrix} c_1(j) & \cdots & c_k(j) & c_{k+s}(j) \\ c_1(j+1) & \cdots & c_k(j+1) & c_{k+s}(j+1) \\ \vdots & \cdots & \vdots & \vdots \\ c_1(j+n_{a_2}-1) & \cdots & c_k(j+n_{a_2}-1) & c_{k+s}(j+n_{a_2}-1) \end{pmatrix}, \quad \forall j = d, \cdots, d+\mu^{\perp} \quad (2.50)$$

où $n_{a_2} = (k+1).(\mu^{\perp}+1)$. Les $(n-k)$ matrices $B^{(s)}$ sont définie par :

$$B^{(s)} = \begin{bmatrix} B_0^{(s)} & B_1^{(s)} & \cdots & B_{\mu^{\perp}-1}^{(s)} & B_{\mu^{\perp}}^{(s)} \end{bmatrix}, \quad \forall s = 1, \cdots, n-k \quad (2.51)$$

Nous noterons $\mathbf{h}_s$ les $(n-k)$ relations de parité qui sont des vecteurs de taille $(k+1).(\mu^{\perp}+1)$:

$$\mathbf{h}_s = \begin{pmatrix} h_{s,1}(\mu^{\perp}) & \cdots & h_{s,k}(\mu^{\perp}) & h_0(\mu^{\perp}) & \cdots & h_{s,1}(0) & \cdots & h_{s,k}(0) & h_0(0) \end{pmatrix} \quad (2.52)$$

Les éléments mis en couleur plus claire correspondent aux bits du polynômes $h_0(D)$ qui sont communs aux $(n-k)$ relations de parité. Afin d'identifier ces $(n-k)$ relations de parité, il faudra construire avec les mots de code les $(n-k)$ matrice $B^{(s)}$ et résoudre ces $(n-k)$ systèmes homogènes :

$$B^{(s)}.\mathbf{h}_s^T = \mathbf{0} \quad (2.53)$$

En résolvant ces $(n-k)$ systèmes, nous devrons vérifier que les bits du polynôme $h_0(D)$ sont communs à chaque système. Nous savons également que comme les relations de parité $\mathbf{h}_s$ ont pour dernier élément le bit $h_0(0)$, les $(n-k)$ vecteurs que nous recherchons se termineront par un bit à "1".

Le coût en terme de complexité de résolution d'un système est de l'ordre de $\mathcal{O}(\big((k+1).(\mu^{\perp}+1)\big)^3)$, soit une complexité totale pour les $(n-k)$ système de l'ordre de $\mathcal{O}((n-k).\big((k+1).(\mu^{\perp}+1)\big)^3)$. Nous remarquons que le coût de cette méthode est plus faible que celle de la méthode 1. De plus, avec la méthode 1, nous avons vu que la déficience de rang de la matrice B était de $(\mu^{\perp}+1).(n-k)-\mu^{\perp}$. Or, avec la méthode 2 la déficience de rang des matrices $B^{(s)}$ est de 1, ce qui implique que le nombre de vecteurs candidats sera moins élevé avec cette seconde méthode. Par conséquent, lors d'une reconnaissance aveugle des code convolutifs nous préférerons utiliser cette seconde méthode.

Exemple 2.29.

Reprenons la suite du $C(3,1,4)$ code de l'exemple 2.27.

L'expression des 3 sorties du codeur en fonction des bits des mots d'information est :

$$\begin{cases} c_1(t) = m_1(t) + m_1(t-2) + m_1(t-3) \\ c_2(t) = m_1(t) + m_1(t-1) + m_1(t-3) \\ c_3(t) = m_1(t) + m_1(t-1) + m_1(t-2) + m_1(t-3) \end{cases}$$

La calcul du rang des matrices construites avec les mots reçus nous a permis d'identifier l'ensemble des paramètres du code ainsi que le paramètre de désynchronisation $d = 2$. D'après ces paramètres, nous allons construire les 2 matrices $B^{(s)}$ (2.51) afin d'identifier la matrice de parité

du code.

$$B^{(s)} = \begin{pmatrix} c_1(2) & c_{k+s}(2) & c_1(3) & c_{k+s}(3) & c_1(4) & c_{k+s}(4) & c_1(5) & c_{k+s}(5) \\ c_1(3) & c_{k+s}(3) & c_1(4) & c_{k+s}(4) & c_1(5) & c_{k+s}(5) & c_1(6) & c_{k+s}(6) \\ c_1(4) & c_{k+s}(4) & c_1(5) & c_{k+s}(5) & c_1(6) & c_{k+s}(6) & c_1(7) & c_{k+s}(7) \\ c_1(5) & c_{k+s}(5) & c_1(6) & c_{k+s}(6) & c_1(7) & c_{k+s}(7) & c_1(8) & c_{k+s}(8) \\ c_1(6) & c_{k+s}(6) & c_1(7) & c_{k+s}(7) & c_1(8) & c_{k+s}(8) & c_1(9) & c_{k+s}(9) \\ c_1(7) & c_{k+s}(7) & c_1(8) & c_{k+s}(8) & c_1(9) & c_{k+s}(9) & c_1(10) & c_{k+s}(10) \\ c_1(8) & c_{k+s}(8) & c_1(9) & c_{k+s}(9) & c_1(10) & c_{k+s}(10) & c_1(11) & c_{k+s}(11) \\ c_1(9) & c_{k+s}(9) & c_1(10) & c_{k+s}(10) & c_1(11) & c_{k+s}(11) & c_1(12) & c_{k+s}(12) \end{pmatrix}, \forall s = 1, 2$$

En résolvant les deux systèmes (2.53), nous obtenons uniquement une solution non-triviale pour chacun des systèmes :

$$\begin{cases} \mathbf{h}_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \\ \mathbf{h}_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \end{cases}$$

Nous pouvons vérifier que les bits du polynôme $h_0(D)$ sont communs à ces deux vecteurs.

2.2.5 Identification d'une matrice génératrice

2.2.5.1 Code de rendement $1/n$

Pour un code de rendement $1/n$, les polynômes générateurs de la matrice de parité $H(D)$ correspondent aux polynômes générateurs de la matrice génératrice. De ce fait, pour ces codeurs l'identification de la matrice de parité nous permet d'obtenir directement la matrice génératrice du code. D'après les propriétés énoncées dans le chapitre 1, le lien entre les polynômes de $H(D)$ et de $G(D)$ est :

$$\begin{cases} h_{i,j}(D) = g_{1,i+1}(D) \\ h_0(D) = g_{1,1}(D) \end{cases} \tag{2.54}$$

Les $(n-k)$ relations de parité définies à l'équation (2.52) sont telles que :

$$\begin{aligned} \mathbf{h}_s &= \begin{pmatrix} h_{s,1}(\mu^\perp) & h_{\mu^\perp}(0) & \cdots & h_{s,1}(0) & h_0(0) \end{pmatrix} \\ &= \begin{pmatrix} g_{1,s+1}(\mu^\perp) & g_{1,1}(\mu^\perp) & \cdots & g_{1,s+1}(0) & g_{1,1}(0) \end{pmatrix} \end{aligned} \tag{2.55}$$

Il suffit de réordonner les bits des $(n-k)$ relations de parité afin d'obtenir la matrice génératrice du code.

Exemple 2.30.

Suite de l'exemple 2.22 du $C(2,1,3)$.

La relation de parité identifiée par la méthode précédente est :

$$\begin{aligned} \mathbf{h} &= \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} h_{1,1}(2) & h_0(2) & h_{1,1}(1) & h_0(1) & h_{1,1}(0) & h_0(0) \end{pmatrix} \end{aligned}$$

D'après (2.55), les deux polynômes de la matrice génératrice sont :

$$\begin{cases} g_{1,1} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} = 5 \\ g_{1,2} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = 7 \end{cases}$$

Ces polynômes correspondent aux polynômes de la matrice génératrice du $C(2,1,3)$ code utilisé.

Exemple 2.31.

Suite de l'exemple 2.27 du $C(3,1,4)$.

Les 2 relations de parité du code que nous avons identifiées sont :

$$\begin{cases} \mathbf{h}_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \\ \mathbf{h}_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \end{cases}$$

D'après (2.55), les trois polynômes de la matrice génératrice sont :

$$\begin{cases} g_{1,1} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix} = 13 \\ g_{1,2} = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} = 15 \\ g_{1,3} = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} = 17 \end{cases}$$

Ces polynômes correspondent aux polynômes de la matrice génératrice du $C(3, 1, 4)$ code utilisé.

2.2.5.2 Code de rendement k/n

Dans le cas d'un code de rendement $r = k/n$, l'identification de la matrice génératrice du code est plus complexe que celle des codes de rendement $1/n$. En effet, il existe un lien entre les polynômes de la matrice de parité et ceux de la matrice génératrice mais ce lien n'est pas aussi direct que dans le cas précédent. Dans cette partie, nous parlerons de l'identification d'*une* matrice génératrice et non de *la* matrice génératrice. Nous avons montré dans le premier chapitre qu'un code pouvait être décrit par plusieurs matrices génératrices équivalentes. Il sera donc possible lors de l'identification aveugle de cette matrice de trouver non pas la matrice du code mais une matrice équivalente.

D'après les propriétés des codes et des codes duaux, une matrice $H(D)$ est une matrice de parité du code C si :

$$G(D).H^T(D) = 0 \tag{2.56}$$

soit :

$$\begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,n}(D) \\ \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,n}(D) \end{bmatrix} \cdot \begin{bmatrix} h_{1,1}(D) & \cdots & h_{n-k,1}(D) \\ \vdots & \cdots & \vdots \\ h_{1,k}(D) & \cdots & h_{n-k,k}(D) \\ h_0(D) & & \\ & \ddots & \\ & & h_0(D) \end{bmatrix} = 0 \tag{2.57}$$

Dans le but de simplifier la résolution du système, nous poserons $(n-k)$ systèmes définis par :

$$\begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) & g_{1,k+s}(D) \\ \vdots & \cdots & \vdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) & g_{k,k+s}(D) \end{bmatrix} \cdot \begin{bmatrix} h_{s,1}(D) \\ \vdots \\ h_{s,k}(D) \\ h_0(D) \end{bmatrix} = 0, \quad \forall s = 1, \cdots, n-k \tag{2.58}$$

soit :

$$\sum_{i=1}^{k} g_{l,i}(D).h_{s,i}(D) + g_{l,k+s}(D).h_0(D) = 0 \tag{2.59}$$

pour $l = 1, \cdots, k$ et $s = 1, \cdots, n-k$.

Afin d'écrire ces $(n-k)$ systèmes sous forme binaire, nous poserons $k.(n-k)$ matrices $D_i^{(s)}$

composées des éléments binaires des polynômes $h_{s,i}(D)$ et qui seront de taille $(k+1).(\mu+1)\times(\mu+1)$:

$$D_i^{(s)} = \begin{pmatrix} h_{s,i}(\mu) & \cdots & h_{s,i}(0) \\ h_{s,i}(\mu+1) & \cdots & h_{s,i}(1) \\ \vdots & \cdots & \vdots \\ h_{s,i}(k.(\mu+1)+2.\mu+1) & \cdots & h_{s,i}(k.(\mu+1)+\mu) \end{pmatrix} \tag{2.60}$$

où $i = 1, \cdots, k$ et $s = 1, \cdots, n-k$. Les polynômes $h_{s,i}(D)$ sont tels que :

$$h_{s,i}(D) = h_{s,i}(0) + h_{s,i}(1).D + \cdots + h_{s,i}(\mu^{\perp}-1).D^{\mu^{\perp}-1} + h_{s,i}(\mu^{\perp}).D^{\mu^{\perp}} \tag{2.61}$$

de ce fait, les éléments $h_{i,j}(l)$ des matrices de l'équation (2.60) pour $l > \mu^{\perp}$ seront considérés comme des éléments à "0". Nous noterons D_0 la $(k+1).(\mu+1) \times (\mu+1)$ matrice composée des éléments binaires du polynôme $h_0(D)$, qui sera commune aux $(n-k)$ systèmes :

$$D_0 = \begin{pmatrix} h_0(\mu) & \cdots & h_0(0) \\ h_0(\mu+1) & \cdots & h_0(1) \\ \vdots & \cdots & \vdots \\ h_0(k.(\mu+1)+2.\mu+1) & \cdots & h_0(k.(\mu+1)+\mu) \end{pmatrix} \tag{2.62}$$

Nous définirons $(n-k)$ matrices, notées $D^{(s)}$, de taille $(k+1).(\mu+1) \times (k+1).(\mu+1)$ telles que :

$$D^{(s)} = \begin{bmatrix} D_1^{(s)} & \cdots & D_k^{(s)} & D_0 \end{bmatrix} \tag{2.63}$$

En notant $\mathbf{g}_{i,j}$ un vecteur de taille $(\mu+1)$ composé des éléments binaires du polynôme générateur $g_{i,j}(D)$, tel que :

$$\mathbf{g}_{i,j} = \begin{pmatrix} g_{i,j}(0) & \cdots & g_{i,j}(\mu) \end{pmatrix} \tag{2.64}$$

nous obtenons les $(n-k)$ systèmes suivant :

$$D^{(s)}.\begin{bmatrix} \mathbf{g}_{i,1} & \cdots & \mathbf{g}_{i,k} & \mathbf{g}_{i,k+s} \end{bmatrix}^T = \mathbf{0}, \quad \forall i = 1, \cdots, k \; \forall s = 1, \cdots, n-k \tag{2.65}$$

Les matrices $D^{(s)}$ construites avec les bits des polynômes générateurs de la matrice du code dual sont des matrices creuses. Ainsi, le nombre de vecteurs solutions des systèmes (2.65) sera élevé. Mais, nous remarquons que les bits des polynômes $g_{i,j}(D)$, $\forall i,j = 1, \cdots, k$ sont communs aux $(n-k)$ systèmes. Nous pourrons donc réduire le nombre de vecteurs solutions en nous basant sur cette propriété. De plus, nous savons que le code utilisé est un code optimal, par conséquent nous pourrons nous baser sur les propriétés de ces codes (énoncées au chapitre 1) afin de réduire le nombre de solutions.

Pour les codes de rendement $1/n$, l'unique solution identifiée correspondra à la matrice génératrice du code. En revanche, pour les codes composés de plusieurs entrées, $k > 1$, nous pourrons, selon le codeur, identifier non pas la matrice génératrice du code, mais un ensemble de matrices génératrices équivalentes. La matrice génératrice du code se trouvera parmi l'ensemble des matrices identifiées, mais il sera mathématiquement impossible de la distinguer des autres puisque ces matrices sont équivalentes. De plus, pour ces codes il existera toujours en sortie de notre algorithme une indéterminée sur l'ordre des lignes de la matrice. En effet, nous aurons les k lignes de la matrice génératrice mais, sans hypothèse sur le code, il nous sera impossible de mettre ces lignes dans le bon ordre.

Au début de ce chapitre, nous avons précisé que notre algorithme de reconnaissance traitait indifféremment les codes RSC et NRNSC. En effet, nous avons montré qu'il existait toujours une forme RSC équivalente à un code NRNSC. De ce fait, en sortie de notre algorithme nous obtien-

drons systématiquement le code sous sa forme NRNSC. Si le code utilisé était un code RSC, nous l'identifierons sous sa forme NRNSC équivalente. D'après les propriétés énoncées au chapitre 1, connaissant la forme NRNSC du code, nous serons capable d'identifier le RSC équivalent. En revanche, sans connaissance à priori sur la nature du code, RSC ou NRNSC, il nous sera impossible de connaître lequel des deux a été utilisé pour coder les mots de code. Effectivement, ces deux codes étant équivalent, il est mathématiquement impossible de les distinguer.

2.2.6 Reconnaissance aveugle du $C(3,2,3)$ code convolutif

Dans cette partie, nous allons reprendre l'exemple du $C(3,2,3)$ code convolutif présenté dans la premier chapitre.

2.2.6.1 Présentation des paramètres du code

Les matrices génératrices du code et du code dual sont telles que :

$$G(D) = \begin{bmatrix} D & 1 & 1+D \\ 1 & D^2 & 1+D+D^2 \end{bmatrix} \quad H(D) = \begin{bmatrix} 1+D+D^3 & 1+D^2+D^3 & 1+D^3 \end{bmatrix} \tag{2.66}$$

Les différents paramètres de ce code (rendement, mémoires, longueur de contrainte, ...) sont :

$$\begin{cases} r = 2/3 \\ \mu_1 = 1,\ \mu_2 = 2,\ \mu = 2 \\ K = 3 \\ \mu^{\perp} = \sum_{i=1}^{k} \mu_i = 3 \\ n_a = 12 \end{cases} \tag{2.67}$$

D'après la matrice génératrice, l'expression des sorties en fonction des entrées à l'instant t est :

$$\begin{cases} c_1(t) = m_2(t) + m_1(t-1) \\ c_2(t) = m_1(t) + m_2(t-2) \\ c_3(t) = m_1(t) + m_2(t) + m_1(t-1) + m_2(t-1) + m_2(t-2) \end{cases} \tag{2.68}$$

Si nous introduisons une désynchronisation de paramètre $t_0 = 11$, alors le train binaire reçu, noté $\tilde{\mathbf{c}}$, sera :

$$\tilde{\mathbf{c}} = \begin{pmatrix} c_3(3) & c_1(4) & c_2(4) & c_3(4) & c_1(5) & \cdots \end{pmatrix} \tag{2.69}$$

A partir de la seule connaissance de cette trame $\tilde{\mathbf{c}}$, nous allons appliquer notre algorithme de reconnaissance afin d'identifier l'ensemble des paramètres du code et une matrice génératrice du code.

2.2.6.2 Reconnaissance aveugle du code

La première étape de reconnaissance consiste à calculer le rang des matrices R_l construites avec la trame $\tilde{\mathbf{c}}$. Le rang des matrices R_l est représenté sur la figure 2.9.

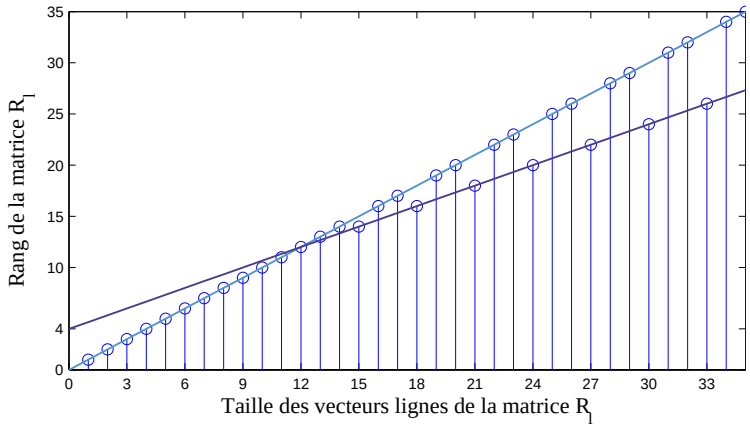


Figure 2.9 — Rang des matrices R_l du $C(3,2,3)$ code avec un décalage de $t_0 = 11$

Nous pouvons voir sur cette figure que la première matrice de rang déficient est de taille $n_{a_1} = 15$ et que la différence entre la taille de deux matrices consécutives de rang déficient nous permet d'identifier la taille des mots de code : $n = 18 - 15 = 3$.

Connaissant n_{a_1} et n, nous allons réaliser une synchronisation aveugle. Pour cela, nous allons construire les matrices $R(n_{a_1}, d)$ avec $d = 0, \cdots, n-1$. Les équations ci-dessous représentent les premières lignes de chaque matrices $R(n_{a_1}, d)$:

$$\begin{aligned}
R(15,0) &= (c_3(3)\ c_1(4)\ c_2(4)\ c_3(4)\ c_1(5)\ c_2(5)\ c_3(5)\ c_1(6)\ c_2(6)\ c_3(6)\ c_1(7)\ c_2(7)\ c_3(7)\ c_1(8)\ c_2(8)) \\
R(15,1) &= (c_1(4)\ c_2(4)\ c_3(4)\ c_1(5)\ c_2(5)\ c_3(5)\ c_1(6)\ c_2(6)\ c_3(6)\ c_1(7)\ c_2(7)\ c_3(7)\ c_1(8)\ c_2(8)\ c_3(8)) \\
R(15,2) &= (c_3(4)\ c_1(5)\ c_2(5)\ c_3(5)\ c_1(6)\ c_2(6)\ c_3(6)\ c_1(7)\ c_2(7)\ c_3(7)\ c_1(8)\ c_2(8)\ c_3(8)\ c_1(9)\ c_2(9))
\end{aligned} \tag{2.70}$$

Nous avons calculer le rang de chacune de ces matrices et nous avons reporté sur la figure 2.10 leurs rangs normalisés. Nous notons que pour $d = 1$, nous obtenons une chute de rang maximale et nous pouvons vérifier d'après (2.70) que pour $d = 1$, le premier bit de la matrice $R(n_{a_1}, 1)$ correspond au premier bit d'un mot de code.

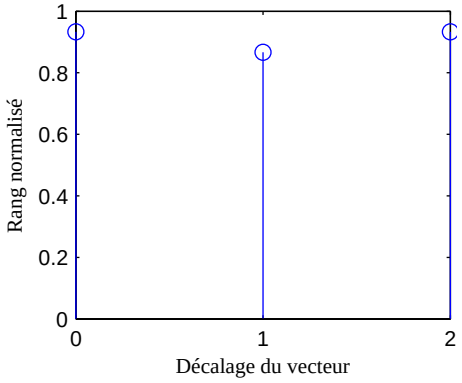


Figure 2.10 — Synchronisation aveugle du $C(3,2,3)$

De par ce paramètre de synchronisation, $d = 1$, nous obtenons le vecteur synchronisé $\mathbf{c}$:

$$\mathbf{c} = \begin{pmatrix} c_1(4) & c_2(4) & c_3(4) & c_1(5) & c_2(5) & c_3(5) & \cdots \end{pmatrix} \tag{2.71}$$

Afin d'identifier les paramètres du code, nous allons calculer le rang des matrices R_l construites avec le vecteur $\mathbf{c}$. Le rang de ces matrices est représenté sur la figure 2.11.

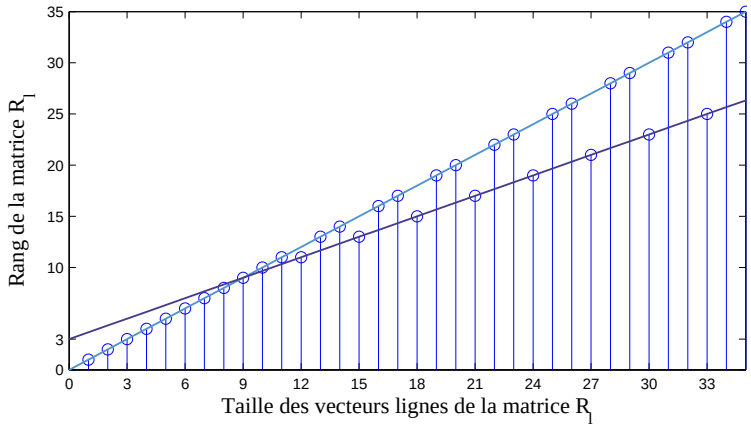


Figure 2.11 — Rang des matrices R_l du $C(3,2,3)$ code

D'après le rang des matrices, en appliquant la méthode d'identification des paramètres, nous obtenons :

$$\begin{cases} n_a = 12 \\ n = 3 \\ k = 2 \\ \mu^{\perp} = 3 \\ \mu = 2 \end{cases} \tag{2.72}$$

Avec ces paramètres, nous pouvons construire la matrice B (2.36), de taille $n_a \times n_a$:

$$B = \begin{pmatrix} c_1(4) & c_2(4) & c_3(4) & c_1(5) & c_2(5) & c_3(5) & c_1(6) & c_2(6) & c_3(6) & c_1(7) & c_2(7) & c_3(7) \\ c_1(5) & c_2(5) & c_3(5) & c_1(6) & c_2(6) & c_3(6) & c_1(7) & c_2(7) & c_3(7) & c_1(8) & c_2(8) & c_3(8) \\ c_1(6) & c_2(6) & c_3(6) & c_1(7) & c_2(7) & c_3(7) & c_1(8) & c_2(8) & c_3(8) & c_1(9) & c_2(9) & c_3(9) \\ c_1(7) & c_2(7) & c_3(7) & c_1(8) & c_2(8) & c_3(8) & c_1(9) & c_2(9) & c_3(9) & c_1(10) & c_2(10) & c_3(10) \\ c_1(8) & c_2(8) & c_3(8) & c_1(9) & c_2(9) & c_3(9) & c_1(10) & c_2(10) & c_3(10) & c_1(11) & c_2(11) & c_3(11) \\ c_1(9) & c_2(9) & c_3(9) & c_1(10) & c_2(10) & c_3(10) & c_1(11) & c_2(11) & c_3(11) & c_1(12) & c_2(12) & c_3(12) \\ c_1(10) & c_2(10) & c_3(10) & c_1(11) & c_2(11) & c_3(11) & c_1(12) & c_2(12) & c_3(12) & c_1(13) & c_2(13) & c_3(13) \\ c_1(11) & c_2(11) & c_3(11) & c_1(12) & c_2(12) & c_3(12) & c_1(13) & c_2(13) & c_3(13) & c_1(14) & c_2(14) & c_3(14) \\ c_1(12) & c_2(12) & c_3(12) & c_1(13) & c_2(13) & c_3(13) & c_1(14) & c_2(14) & c_3(14) & c_1(15) & c_2(15) & c_3(15) \\ c_1(13) & c_2(13) & c_3(13) & c_1(14) & c_2(14) & c_3(14) & c_1(15) & c_2(15) & c_3(15) & c_1(16) & c_2(16) & c_3(16) \\ c_1(14) & c_2(14) & c_3(14) & c_1(15) & c_2(15) & c_3(15) & c_1(16) & c_2(16) & c_3(16) & c_1(17) & c_2(17) & c_3(17) \\ c_1(15) & c_2(15) & c_3(15) & c_1(16) & c_2(16) & c_3(16) & c_1(17) & c_2(17) & c_3(17) & c_1(18) & c_2(18) & c_3(18) \end{pmatrix} \tag{2.73}$$

Afin d'identifier la relation de parité du code, nous allons résoudre le système homogène suivant :

$$B.\mathbf{h}^T = 0 \tag{2.74}$$

où $\mathbf{h}$ est un vecteur de taille 12. En résolvant ce système, nous obtenons une unique solution non nulle :

$$\mathbf{h} = (1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1) \tag{2.75}$$

D'après les paramètres du code identifiés, nous savons que la matrice de parité du code est composée de 3 polynômes générateurs $h_{1,1}(D)$, $h_{1,2}(D)$ et $h_0(D)$. D'après (2.52) et la relation de parité identifiée (2.75), ces trois polynômes sont tels que :

$$\begin{cases} \mathbf{h}_{1,1} = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} = 15 \\ \mathbf{h}_{1,2} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix} = 13 \\ \mathbf{h}_0 = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} = 11 \end{cases} \tag{2.76}$$

Avec ces polynômes générateurs de la matrice de parité que nous avons identifié, nous pouvons à présent construire la matrice D (2.63) qui nous permettra d'identifier une matrice génératrice du code :

$$\begin{aligned} D &= \begin{pmatrix} h_{1,1}(2) & h_{1,1}(1) & h_{1,1}(0) & h_{1,2}(2) & h_{1,2}(1) & h_{1,2}(0) & h_0(2) & h_0(1) & h_0(0) \\ h_{1,1}(3) & h_{1,1}(2) & h_{1,1}(1) & h_{1,2}(3) & h_{1,2}(2) & h_{1,2}(1) & h_0(3) & h_0(2) & h_0(1) \\ 0 & h_{1,1}(3) & h_{1,1}(2) & 0 & h_{1,2}(3) & h_{1,2}(2) & 0 & h_0(3) & h_0(2) \\ 0 & 0 & h_{1,1}(3) & 0 & 0 & h_{1,2}(3) & 0 & 0 & h_0(3) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{2.77}$$

Notons $\mathbf{f}$ un ensemble de vecteurs de taille 9 tel que :

$$D.\mathbf{f}^T = 0 \tag{2.78}$$

En résolvant ce système et en nous basant sur les propriétés des codes convolutifs optimaux que nous avons présentées dans le premier chapitre, nous obtenons au final deux matrices génératrices :

$$G_1 = \begin{pmatrix} 4 & 1 & 7 \\ 1 & 2 & 3 \end{pmatrix} \text{ et } G_2 = \begin{pmatrix} 1 & 2 & 3 \\ 6 & 5 & 1 \end{pmatrix} \tag{2.79}$$

Ces deux matrices étant équivalentes, il nous est impossible sans information supplémentaire de les différencier. Nous ne pourrons donc pas choisir entre ces deux matrices génératrices. De plus, l'ordre des lignes de ces matrices n'est pas forcément le bon et nous n'avons aucun critère permettant de trouver le bon ordre. En revanche, nous remarquons qu'en ne tenant pas compte de l'ordre des lignes, la matrice G_1 correspond à la matrice génératrice du code.

En résumant les différents paramètres identifiés par notre algorithme de reconnaissance, nous obtenons au final deux codes : le $C(3, 2, 3)$ code de matrice génératrice G_1 et le $C(3, 2, 3)$ code de matrice génératrice G_2.

2.3 Reconnaissance aveugle d'un code convolutif dans le cas bruité

La méthode de reconnaissance d'un code convolutif que nous venons de présenter est très efficace lorsque les mots de code reçus sont propres, mais en présence d'erreurs de transmission cette méthode deviendra très vite inefficace. Nous proposerons dans cette partie une nouvelle méthode qui permettra de reconnaître un code à partir d'une trame codée bruitée ou non-bruitée. Afin de modéliser les erreurs de transmission, nous ferons l'hypothèse que les parties modulation, canal de transmission et démodulation sont modélisées par un canal binaire symétrique (BSC). Ce type de canal n'est certes pas le plus réaliste, compte-tenu du fait que les bruits générés par une commu-

nication radio-mobile sont de type burst et que ce canal modélisera des erreurs isolées. Mais, avec des erreurs de type burst, nous savons qu'il existe des plages propres de longueur assez grande pour appliquer la méthode de reconnaissance précédente. En revanche, avec des erreurs isolées, il serait impossible de trouver une plage propre pour pouvoir appliquer cette même méthode. De ce fait, en optant pour cette modélisation, nous nous plaçons au final dans le cas le plus défavorable pour nos algorithmes de reconnaissance aveugle.

Nous noterons P_e, la probabilité d'erreur du canal binaire symétrique et $\mathbf{y}$, le vecteur des mots reçu après transmission, tel que :

$$\mathbf{y} = \mathbf{c} + \mathbf{e} \tag{2.80}$$

où le vecteur $\mathbf{e}$ contient les erreurs de transmission et $\mathbf{c}$ la séquence des mots de code.

En présence d'erreusr de transmission, il existe dans la littérature très peu de méthodes permettant d'identifier les paramètres et la matrice génératrice d'un code convolutif. Dans [BSH06] une approche algébrique permettant d'identifier une base du code dual, soit un ensemble de relations de parité du code est proposée. Une méthode basée sur l'algorithme d'Euclide a été développée dans [WHZ07] afin d'identifier un code convolutif de rendement 1/2. Enfin, le lecteur intéressé pourra se référer à [DH07] où une méthode d'identification aveugle d'un code convolutif de rendement $1/n$ est expliquée. Ce dernier est un algorithme probabiliste basé sur l'algorithme EM. En revanche, la littérature concernant la reconnaissance aveugle des codes en blocs est plus riche et ces algorithmes, tout du moins une partie, peuvent être adaptés dans le cadre de la reconnaissance aveugle de code convolutif. Nous pouvons cité la méthode présentée dans [CF09] qui permet d'identifier un code poinçonné en utilisant l'algorithme proposé dans [VF01] dans le cadre d'un code en bloc.

Nous présenterons dans cette section un algorithme qui nous permettra d'estimer les paramètres ainsi qu'une matrice génératrice d'un code convolutif optimal. Nous ferons l'hypothèse dans cette partie que la trame reçue est synchronisée et entachée d'erreurs. Nous avons présenté cette méthode, dans le cas d'un codeur de rendement $(n-1)/n$ dans [MGB09b].

2.3.1 Quelques notations et définitions

Nous noterons $\tilde{R}$ une matrice de taille $M \times l$ construite à partir de la séquence bruitée $\mathbf{y}$, $\tilde{\mathbf{r}}$ une ligne de cette matrice et $\mathbf{h}$ une relation de parité du code. Pour $l = \alpha.n \geq n_a$, sans erreur de transmission, chaque vecteur ligne de la matrice $\tilde{R}$ vérifie la relation $\tilde{\mathbf{r}}.\mathbf{h}^T = 0$. En revanche, en présence d'erreurs de transmission, certaines lignes de cette matrice vérifieront la relation $\tilde{\mathbf{r}}.\mathbf{h}^T = 1$. Cependant, en fonction du poids de Hamming de la relation de parité $\mathbf{h}$ et de la probabilité d'erreur du canal P_e, il a été montré dans [Clu06] qu'il était possible de déterminer la probabilité de ces deux évènements.

Proposition 2.3. *Soit $\mathbf{h} \in C^{\perp}$ et $\tilde{\mathbf{r}}$ un vecteur de mots de code bruité par un canal binaire symétrique de probabilité d'erreur P_e. Alors :*

$$Pr\left[\tilde{\mathbf{r}}.\mathbf{h}^T = 0\right] = \frac{1 + (1 - 2.P_e)^{w(\mathbf{h})}}{2}, \tag{2.81}$$

et

$$Pr\left[\tilde{\mathbf{r}}.\mathbf{h}^T = 1\right] = \frac{1 - (1 - 2.P_e)^{w(\mathbf{h})}}{2}. \tag{2.82}$$

où $w(\mathbf{h})$ est le poids de Hamming de la relation de parité $\mathbf{h}$.

D'après cette proposition, il est possible de déterminer le poids de Hamming moyen de la relation $\tilde{R}.\mathbf{h}^T$ en fonction du vecteur $\mathbf{h}$.

– Si $\mathbf{h} \in C^{\perp}$:

$$w\left(\tilde{R}.\mathbf{h}^T\right) \simeq M.\frac{1-(1-2.P_e)^{w(\mathbf{h})}}{2} \tag{2.83}$$

– Si $\mathbf{h} \notin C^{\perp}$:

$$w\left(\tilde{R}.\mathbf{h}^T\right) \simeq \frac{M}{2} \tag{2.84}$$

Les deux comportements du poids de la relation $w\left(\tilde{R}.\mathbf{h}^T\right)$ vont nous permettre d'établir un critère afin de déterminer, avec une certaine probabilité, si le vecteur $\mathbf{h}$ est ou n'est pas une relation de parité du code. D'après la proposition 2.3, il est nécessaire de connaître la probabilité d'erreur du canal afin de pouvoir distinguer les deux comportements du poids de $w\left(\tilde{R}.\mathbf{h}^T\right)$. Or, étant dans un contexte non coopératif, cette probabilité d'erreur est inconnue au niveau du récepteur. En conséquence, nous proposerons tout d'abord une méthode qui nous permettra d'identifier certaines matrices qui sont probablement de rang déficient. A partir de la connaissance de ces matrices, nous serons en mesure d'estimer le nombre de sorties du codeur et la probabilité d'erreur du canal. Nous pourrons par la suite estimer une base du code dual, soit un ensemble de relations de parité et en déduire les paramètres du code et une matrice génératrice. Avant de présenter notre algorithme de reconnaissance, nous présenterons tout d'abord quelques définitions nécessaires pour sa bonne compréhension.

2.3.2 Principe de notre méthode de reconnaissance aveugle

Pour un train binaire non entaché d'erreurs, le calcul du rang des matrices construites avec ces données nous a permis d'identifier les paramètres du code convolutif. Or, en présence d'erreurs, cette méthode deviendra très vite inefficace. En effet, les chutes de rang des matrices de taille $l = \alpha.n \geq n_a$ correspondent aux nombres de colonnes dépendantes. Or, les erreurs générées par le canal vont perturber certains bits, donc la dépendance entre les colonnes. Ainsi, la plupart des matrices seront de rang plein. Dans [SH05a], il a été proposé une méthode, dans le cas d'un train binaire codé par un code en bloc et entrelacé, permettant de pallier ce problème. Le principe de cette méthode est de ne plus rechercher les colonnes dépendantes mais les colonnes qui sont "presque dépendantes". Nous allons utiliser ce même principe afin de rechercher les matrices qui sont probablement de rang déficient.

Nous noterons $\tilde{R}_l$, les matrices de taille $M \times l$ construites avec les mots reçu $\mathbf{y}$. Le paramètre M sera fixe et l variera de 1 à l_{max}. Nous chercherons dans les matrices $\tilde{R}_l$ le nombre de colonnes "presque dépendantes", afin de déterminer si la matrice est de rang déficient ou non. Pour cela, on triangularise la matrice $\tilde{R}_l$ à l'aide du pivot de Gauss adapté à $GF(2)$:

$$A_l.\tilde{R}_l.B_l = T_l \tag{2.85}$$

où A_l et B_l sont des matrices de taille $M \times M$ et $l \times l$ respectivement. Les permutations sur les lignes réalisées durant le pivot de Gauss sont représentées dans A_l, les permutations et additions de colonnes sont représentées dans B_l et la matrice T_l est une $M \times l$ matrice triangulaire inférieure. Nous montrerons qu'en fonction du nombre d'éléments à "1" présent dans les colonnes des matrices T_l, il sera possible de déterminer les matrices $\tilde{R}_l$ qui sont probablement de rang déficient.

2.3.2.1 La matrice triangulaire inférieure

La figure 2.12 représente une matrice T_l qui est une matrice triangulaire inférieure de taille $M \times l$.

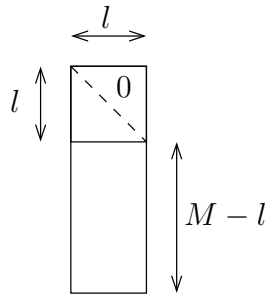


Figure 2.12 — Matrice triangulaire inférieure

Le rang de la matrice $\tilde{R}_l$ correspond au nombre de colonnes indépendantes, soit au nombre de "1" présents sur la diagonale de sa matrice triangulaire T_l. Pour les matrices de taille $l = \alpha.n \geq n_a$, nous avons montré qu'il existait des colonnes dépendantes, soit des combinaisons linéaires entre les colonnes de $\tilde{R}_l$. Nous définirons une combinaison linéaire par un ensemble d'indices noté $\mathcal{I} = \{i_1, \cdots, i_p\}$ tel que la somme des colonnes d'indice $\mathcal{I}$ de $\tilde{R}_l$ soit nulle :

$$\tilde{\mathbf{r}}_{i_1}^l + \cdots + \tilde{\mathbf{r}}_{i_p}^l = 0 \tag{2.86}$$

où $\tilde{\mathbf{r}}_i^l$ correspond à la i-ème colonne de la matrice $\tilde{R}_l$.

En présence d'erreurs de transmission, un unique bit erroné sur les colonnes d'indice $\mathcal{I}$ de la matrice $\tilde{R}_l$ implique que la combinaison linéaire entre ces colonnes n'existera plus. Nous n'allons plus rechercher les colonnes qui satisfont l'équation (2.86), mais des colonnes qui sont telles que :

$$\tilde{\mathbf{r}}_{i_1}^l + \cdots + \tilde{\mathbf{r}}_{i_p}^l \approx 0 \tag{2.87}$$

Nous montrerons qu'en fonction du nombre d'éléments à "1" présents dans les colonnes de la partie inférieure de T_l, il sera possible de distinguer certaines de ces combinaisons. Par partie inférieure, nous entendons les $(M - l)$ dernières lignes de la matrice T_l.

Par la suite, nous découperons le produit $A_l.\tilde{R}_l$ en deux sous-matrices. Nous noterons R_1^l les l premières lignes résultant de ce produit et R_2^l les $(M - l)$ dernières lignes, comme représenté sur la figure 2.13.

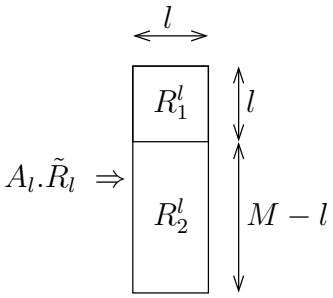


Figure 2.13 — Représentation du produit $A_l.\tilde{R}_l$

- Pour $l \neq \alpha.n$ ou $l < n_a$

Dans cette configuration, il n'existe aucune combinaison linéaire entre les colonnes de la matrice $\tilde{R}_l$. Cette matrice sera de rang plein et la diagonale de la matrice triangulaire T_l sera composée de l éléments à "1". Les éléments se trouvant au dessus de cette diagonale seront nuls et les éléments étant en dessous pourront être des "1" et des "0". La figure 2.14 représente une matrice T_l dans cette configuration.

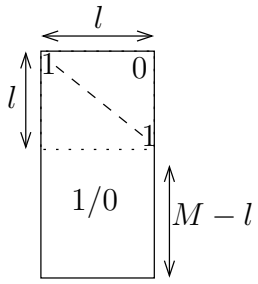


Figure 2.14 — Matrice triangulaire inférieure T_l pour $l \neq \alpha.n$ ou $l < n_a$

Notons $N_l(i)$ une variable contenant le nombre de bit à "1" présent dans la i-ème colonne de la partie inférieure de T_l. En notant $\mathbf{b}_i^l$ la i-ème colonne de la matrice B_l, la variable $N_l(i)$ correspond au poids de la relation $R_2^l.\mathbf{b}_i^l$:

$$N_l(i) = w\left(R_2^l.\mathbf{b}_i^l\right) \tag{2.88}$$

Dans cette configuration, nous savons que le vecteur $\mathbf{b}_i^l$ n'appartient pas au code dual, donc d'après l'équation (2.84), le poids moyen de la relation $R_2^l.\mathbf{b}_i^l$ sera proche de $\frac{M-l}{2}$. De ce fait, la variable $N_l(i)$ suivra une loi binomiale de paramètres $(M-l)$ et $1/2$ que nous noterons $\mathcal{B}(M-l, 1/2)$.

- Pour $l = \alpha.n \geq n_a$

Dans ce cas, nous savons qu'il existe au moins une combinaison linéaire entre les colonnes de la matrice $\tilde{R}_l$. Faisons tout d'abord l'hypothèse que le canal de transmission n'a généré aucune erreur, $P_e = 0$. Dans cette hypothèse, la matrice $\tilde{R}_l$ est de rang déficient. Nous noterons r le rang de $\tilde{R}_l$, tel que $r < l$. Alors la diagonale de la matrice T_l sera composée de r éléments à "1" et de $l - r$ éléments à "0". Cette matrice aura $l - r$ colonnes nulles et r colonnes qui seront composées (en dessous de la diagonale) d'éléments à "1" et "0". La figure 2.15 représente une matrice T_l dans cette configuration.

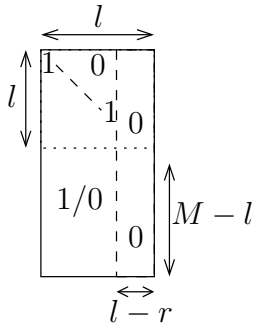


Figure 2.15 — Matrice triangulaire inférieure T_l pour $l = \alpha.n \geq n_a$

La variable $N_l(i)$ aura donc deux comportements en fonction de i. En effet, si le vecteur $\mathbf{b}_i^l$ n'appartient pas au code dual, la variable $N_l(i)$ sera telle que :

$$N_l(i) = w\left(R_2^l.\mathbf{b}_i^l\right) \approx \frac{M-l}{2} \tag{2.89}$$

En revanche, d'après l'équation (2.83) si le vecteur $\mathbf{b}_i^l$ appartient au code dual, le poids moyen sera :

$$N_l(i) = w\left(R_2^l.\mathbf{b}_i^l\right) \approx (M-l).P \tag{2.90}$$

avec P qui correspond à la probabilité définie par :

$$P = \frac{1-(1-2.P_e)^{w(\mathbf{b}_i^l)}}{2} \tag{2.91}$$

Dans cette configuration, la variable $N_l(i)$ suivra une loi Binomiale de paramètres $(M-l)$ et P : $\mathcal{B}(M-l,P)$.

En fonction du vecteur $\mathbf{b}_i^l$, la variable $N_l(i)$ a donc deux comportements différents :

– Si $\mathbf{b}_i^l \notin C^\perp$:

$$N_l(i) \rightarrow \mathcal{B}(M-l,\frac{1}{2}) \tag{2.92}$$

– Si $\mathbf{b}_i^l \in C^\perp$:

$$N_l(i) \rightarrow \mathcal{B}(M-l,P) \tag{2.93}$$

En définissant un seuil, nous serons en mesure de délimiter les deux comportements de la variable $N_l(i)$ et de décider si le vecteur $\mathbf{b}_i^l$ appartient ou non au code dual $C^\perp$. Nous définirons deux hypothèses $\mathcal{H}_0$ et $\mathcal{H}_1$ telles que $\mathcal{H}_0$ correspond à l'hypothèse où $\mathbf{b}_i^l$ appartient au code dual et $\mathcal{H}_1$ correspond à l'hypothèse où $\mathbf{b}_i^l$ n'appartient pas au code dual.

$$\mathcal{H}_0 \text{ si } \mathbf{b}_i^l \in C^\perp \quad \text{et} \quad \mathcal{H}_1 \text{ si } \mathbf{b}_i^l \notin C^\perp$$

En définissant un seuil γ, d'après les deux comportements de $N_l(i)$, nous déciderons que :

$$\begin{cases} \text{si } N_l(i) \leq \frac{M-l}{2}.\gamma \text{ alors } \mathcal{H}_0 \\ \\ \text{si } N_l(i) > \frac{M-l}{2}.\gamma \text{ alors } \mathcal{H}_1 \end{cases} \tag{2.94}$$

Afin d'optimiser les performances de notre algorithme, nous allons étudier le seuil γ.

2.3.2.2 Étude du seuil de détection γ

Afin d'étudier le seuil γ, nous devrons étudier les lois de probabilité de la variable $N_l(i)$. Nous ferons tout d'abord un bref rappel sur la loi Binomiale.

- Loi Binomiale

Notons $\mathcal{B}(n,p)$ une loi Binomiale de paramètres n et p. La variable aléatoire X suit une loi de probabilité définie par :

$$p(k) = Pr(X=k) = \binom{n}{k} p^k.(1-p)^{n-k} \tag{2.95}$$

Cette loi de probabilité correspond à l'expérience suivante :

On renouvelle n fois de manière indépendante une épreuve de Bernoulli de paramètre p. Une épreuve de Bernoulli correspond à une expérience aléatoire comportant deux issues : le succès de probabilité p et l'échec de probabilité $(1-p)$. La variable aléatoire X correspond au nombre de succès à l'issue des n épreuves.

- Loi de probabilité de la variable $N_l(i)$

Nous avons montré que la variable $N_l(i)$ avait deux comportements différents en fonction du vecteur $\mathbf{b}_i^l$. Nous allons étudier le comportement de cette variable en fonction des deux hypothèses $\mathcal{H}_0$ et $\mathcal{H}_1$.

Hypothèse $\mathcal{H}_0$:

Sous cette hypothèse, nous savons que la variable $N_l(i)$ suit une loi Binomiale $\mathcal{B}(M-l,P)$. Donc, la probabilité pour que la variable $N_l(i)$ soit inférieure ou égale à $\frac{M-l}{2}.\gamma$ est :

$$Pr\left(N_l(i) \leq \frac{M-l}{2}.\gamma|\mathcal{H}_0\right) = \sum_{j=0}^{\lfloor \frac{M-l}{2} \rfloor.\gamma} \binom{M-l}{j}.P^j.(1-P)^{M-l-j} \tag{2.96}$$

Hypothèse $\mathcal{H}_1$:

Dans ce cas, la variable $N_l(i)$ suit une loi Binomiale $\mathcal{B}(M-l,1/2)$. La probabilité pour que la variable $N_l(i)$ soit supérieure à $\frac{M-l}{2}.\gamma$ est :

$$Pr\left(N_l(i) > \frac{M-l}{2}.\gamma|\mathcal{H}_1\right) = \sum_{j=\lfloor \frac{M-l}{2} \rfloor.\gamma+1}^{M-l} \binom{M-l}{j}.\frac{1}{2}^{M-l} \tag{2.97}$$

D'après ces différentes probabilités nous allons chercher à calculer le seuil γ optimal dans le sens où il minimisera les probabilités de non détection.

- Le seuil optimal :

Notons P_{fa} la probabilité de fausse alarme qui correspond à la probabilité de décider que $\mathbf{b}_i^l \in C^\perp$ alors qu'en réalité $\mathbf{b}_i^l \notin C^\perp$. Cette probabilité correspond à :

$$P_{fa} = Pr\left(N_l(i) \leq \frac{M-l}{2}.\gamma|\mathcal{H}_1\right) = \sum_{j=0}^{\lfloor \frac{M-l}{2} \rfloor.\gamma} \binom{M-l}{j}.\frac{1}{2}^{M-l} \tag{2.98}$$

Nous noterons P_{nd} la probabilité de non détection qui correspond au cas où l'on décide que $\mathbf{b}_i^l \notin C^\perp$ alors que $\mathbf{b}_i^l \in C^\perp$. Cette probabilité est définie par :

$$P_{nd} = Pr\left(N_l(i) > \frac{M-l}{2}.\gamma|\mathcal{H}_0\right) = \sum_{j=\lfloor \frac{M-l}{2} \rfloor.\gamma+1}^{M-l} \binom{M-l}{j}.P^j.(1-P)^{M-l-j} \tag{2.99}$$

La probabilité de détection, notée P_{det}, est telle que :

$$P_{det} = 1 - P_{nd} = Pr\left(N_l(i) \leq \frac{M-l}{2}.\gamma|\mathcal{H}_0\right) = \sum_{j=0}^{\lfloor \frac{M-l}{2} \rfloor.\gamma} \binom{M-l}{j}.P^j.(1-P)^{M-l-j} \tag{2.100}$$

En prenant comme critère à minimiser la somme de la probabilité de fausse alarme et la probabilité de non détection, le seuil optimal γ_{opt} est définie par :

$$\begin{aligned} \gamma_{opt} &= \arg\min_{\gamma} \left(P_{nd} + P_{fa}\right) = \arg\min_{\gamma} \left(1 - P_{det} + P_{fa}\right) \\ &= \arg\min_{\gamma} \left(1 + \sum_{j=0}^{\lfloor \frac{M-l}{2} \rfloor.\gamma} \binom{M-l}{j} \left[2^{l-M} - P^j.(1-P)^{M-l-j}\right] \right) \end{aligned} \tag{2.101}$$

Nous remarquons que ce seuil optimal dépend des paramètres M, l et P. Or, dans un contexte non-coopératif, le paramètre P, qui dépend du poids de Hamming du vecteur $\mathbf{b}_i^l$ et de la probabilité d'erreur du canal, P_e, est inconnu. Il est donc impossible, au début de notre algorithme de calculer

ce seuil optimal. Par conséquent, nous devrons, dans un premier temps, fixer un seuil de départ.

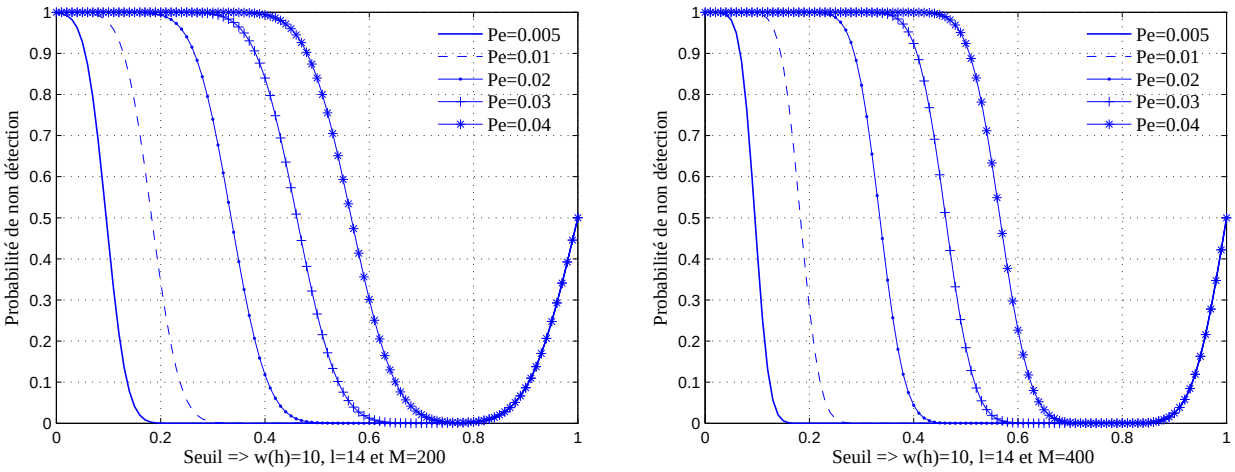
- Le choix du seuil de départ :

Dans [SH05b] (algorithme développé pour les codes en blocs), il a été proposé de fixer un seuil de départ à 0.6. Nous montrerons que dans notre contexte, en fonction des paramètres des codes, le choix de ce seuil de départ semble également judicieux. D'après (2.101), le seuil optimal dépend des paramètres M, l et P, avec le paramètre P qui est défini par :

$$P = \frac{1 - (1 - 2.P_e)^{w(\mathbf{h})}}{2} \tag{2.102}$$

Nous allons voir l'influence que ces différents paramètres ont sur la valeur du seuil qui permettra d'obtenir une probabilité de non détection nulle.

Regardons tout d'abord l'impact de la probabilité d'erreur du canal sur la valeur du seuil γ optimal. Nous fixerons le poids de Hamming de la relation $\mathbf{h}$ à 10 et le paramètre l à 14. Nous avons représenté sur les figures 2.16(a) et 2.16(b) l'évolution de la probabilité de non détection en fonction du seuil optimal pour différentes probabilité d'erreur du canal.

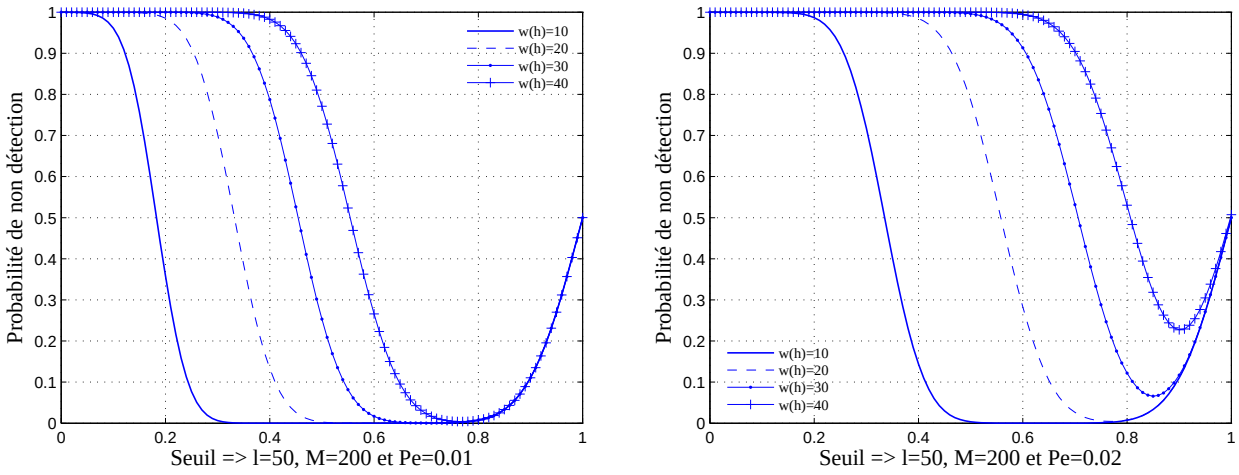


(a) $P_{nd} = f(\gamma, P_e)$ pour $M = 200$ (b) $P_{nd} = f(\gamma, P_e)$ pour $M = 400$

Figure 2.16 — Seuil de détection γ en fonction de P_e

Pour obtenir la figure 2.16(a), nous avons fixé la valeur de M à 200 et nous l'avons mise à 400 pour la figure 2.16(b). Tout d'abord, nous pouvons noter que la valeur de M a peu d'influence sur le choix du seuil. En effet, nous remarquons quasiment le même comportement sur les deux courbes. En revanche, la probabilité d'erreur est un facteur important dans le choix du seuil. L'objectif étant d'obtenir $P_{nd}\#0$, nous notons que, plus P_e sera élevé, plus l'intervalle du seuil permettant d'avoir $P_{nd}\#0$ sera faible. En effet, pour $P_e \leq 0.01$, un seuil compris entre 0.3 et 0.8 permet de garantir une probabilité de non détection quasi-nulle. En revanche pour $P_e \geq 0.03$, cet intervalle se réduit à des valeurs comprises entre 0.6 et 0.8. De ce fait, en fixant un seuil de départ à 0.6, nous remarquons que pour des probabilités d'erreur du canal inférieures à 0.03 nous serons en mesure de détecter quelques relations de parité du code.

Pour une probabilité d'erreur P_e fixé à 0.01 et à 0.02, nous avons représenté sur les figures 2.17(a) et 2.17(b) l'évolution de la probabilité de non détection en fonction du seuil, pour différents poids de la relation $\mathbf{h}$ (ici $l = 50$ et $M = 200$).



(a) $P_{nd} = f(\gamma, w(\mathbf{h}))$ pour $P_e = 0.01$

(b) $P_{nd} = f(\gamma, w(\mathbf{h}))$ pour $P_e = 0.02$

Figure 2.17 — Seuil de détection γ en fonction de $w(\mathbf{h})$

Nous remarquons sur ces deux figures que le poids de Hamming du vecteur $\mathbf{h}$ est également un facteur important. En effet, plus ce poids sera grand, plus il sera difficile de détecter cette relation de parité. Prenons l'exemple d'une relation de parité de poids de Hamming égal à 40. Nous pouvons voir que pour $P_e = 0.01$, le seuil devra être compris entre 0.7 et 0.8 pour obtenir $P_{nd}\#0$. Et pour $P_e = 0.02$, il sera impossible de trouver un seuil qui garantira une probabilité de non détection quasi-nulle. Mais, l'objectif de notre algorithme de reconnaissance est d'identifier les paramètres et une matrice génératrice du code. Or, nous avons montré que pour identifier une matrice génératrice, les relations de parité utiles correspondaient aux relations de parité de plus petit degré. Par conséquent, parmi l'ensemble des relations de parité d'un code nous chercherons les relations de parité qui sont de poids faible. Ainsi, un seuil de départ fixé à 0.6, nous permettra tout de même d'identifier les relations de parité de petit poids du code, même pour des probabilités d'erreur de canal allant jusqu'à 0.02.

Compte-tenu des paramètres du canal de transmission et des codes convolutifs, le choix de fixer un seuil de départ à 0.6 au début semble être un bon compromis. En effet, avec un seuil moins élevé le risque serait de ne détecter aucune des relations linéaires si elles sont de poids élevées ou si la probabilité d'erreur du canal est élevée. Et en choisissant un seuil trop élevé, nous risquerions de détecter des relations linéaires qui n'appartiennent pas au code dual.

2.3.3 Algorithme de reconnaissance

D'après les différentes propriétés que nous venons d'énoncer, nous allons pouvoir développer notre algorithme de reconnaissance aveugle. Nous allons découper cet algorithme en trois parties :

1. Estimation du nombre de sorties n et de la probabilité d'erreur du canal P_e
2. Estimation d'une base du code dual
3. Estimation des paramètres k et K et d'une matrice génératrice du code

2.3.3.1 Estimation de n et P_e

Nous savons que la taille des mots de code correspond à la différence entre la taille de deux matrices de rang déficient consécutives. En présence d'erreurs de transmission, nous n'allons pas calculer le rang des matrices mais réaliser une triangularisation afin de déterminer le nombre de

colonnes de chaque matrice $\tilde{R}_l$ qui sont probablement dépendantes :

$$T_l = A_l.\tilde{R}_l.B_l \tag{2.103}$$

Nous noterons $Q(l)$ le nombre de colonnes probablement dépendantes pour chaque matrice $\tilde{R}_l$ tel que :

$$Q(l) = \text{Card}\left\{i \in \{1, \cdots, l\} | N_l(i) \leq \frac{M-l}{2}.\gamma\right\} \tag{2.104}$$

Nous avons vu précédemment que la variable $N_l(i) = w\left(R_2^l.\mathbf{b}_i^l\right)$ avait deux comportements différents en fonction de l et que d'après les différentes propriétés énoncées plus haut, le seuil γ fixé à 0.6, devrait nous permettre de délimiter les deux comportements de la variable $N_l(i)$. En effet, si toute les colonnes de B_l sont indépendantes alors la variable $Q(l)$ sera nulle. En revanche, si des colonnes dépendantes sont détectées alors $Q(l)$ sera non-nulle.

En présence d'erreurs, il existera des matrices de taille $l = \alpha.n$ ou aucune colonne dépendante ne sera détectée, par conséquent la variable $Q(l)$ sera nulle. De plus, pour des matrices de taille $l \neq \alpha.n$, il est possible que notre algorithme détecte des colonnes dépendantes, ce qui signifie que la variable $Q(l)$ sera non nulle. Par conséquent, nous ne pourrons pas identifier la taille des mots de code en prenant simplement la distance entre deux pics de $Q(l)$ non-nulle. Nous noterons $\mathcal{I}$ l'ensemble des valeurs de l qui correspondent à :

$$\mathcal{I} = \{l = 1, \cdots, l_{max} | Q(l) \neq 0\} \tag{2.105}$$

Afin d'obtenir la meilleure estimation de la taille des mots de code, nous chercherons la distance dont l'occurrence est la plus élevée dans l'ensemble $\mathcal{I}$. Pour cela, nous définirons les fonctions diff($\mathbf{x}$) et mode($\mathbf{x}$).

- Fonction diff($\mathbf{x}$) :
  Si $\mathbf{x}$ est un vecteur de taille $(m+1)$, $\mathbf{x} = (x(0)\ x(1)\ \cdots\ x(m))$, alors diff($\mathbf{x}$) est un vecteur de taille m qui correspond à la différence entre deux éléments consécutifs :

$$\text{diff}(\mathbf{x}) = \begin{pmatrix} x(1)-x(0) & x(2)-x(1) & \cdots & x(m)-x(m-1) \end{pmatrix} \tag{2.106}$$

- Fonction mode($\mathbf{x}$) :
  L'opération mode($\mathbf{x}$) permet d'obtenir la valeur qui a la plus grande occurrence dans le vecteur $\mathbf{x}$.

De ce fait, la taille des mots de code estimée, notée $\hat{n}$, sera telle que :

$$\hat{n} = \text{mode}(\text{diff}(\mathcal{I})) \tag{2.107}$$

L'algorithme 1 représente les différentes étapes réalisées afin d'obtenir une première estimation de $\hat{n}$ et la taille de quelques matrices qui sont probablement de rang déficient.

Exemple 2.32.

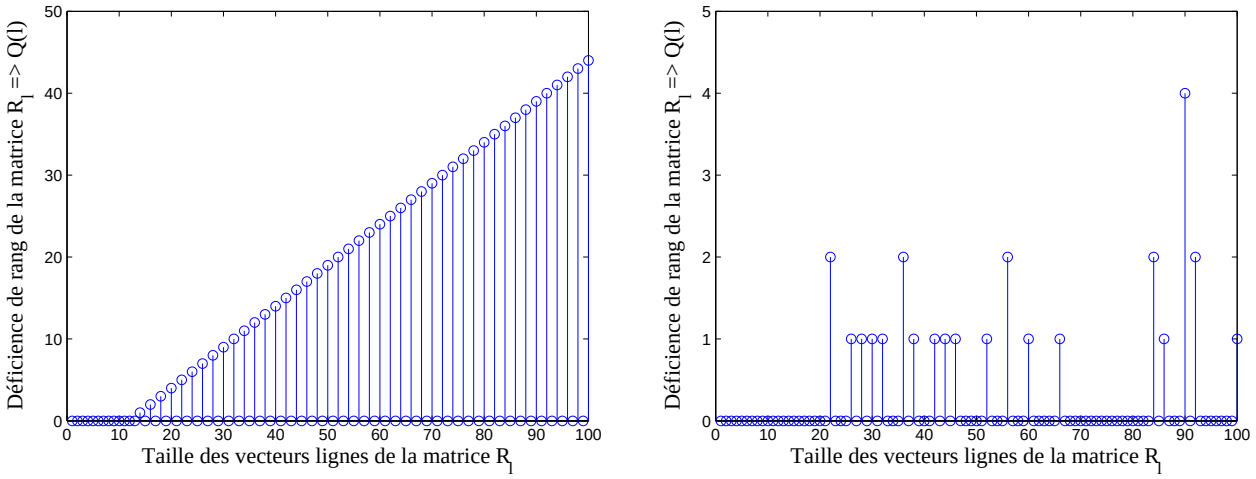
Prenons l'exemple du code $C(2, 1, 7)$ de matrice génératrice $G = \begin{pmatrix} 133 & 171 \end{pmatrix}$.

Sur la figure 2.18(a), nous avons représenté la déficience de rang des matrices R_l lorsque la probabilité d'erreur du canal $P_e = 0$. Afin d'obtenir ces déficiences de rang, nous avons utilisé l'algorithme 1 avec un seuil γ fixé à 0. Sur la figure 2.18(b), nous avons représenté pour différentes valeurs de l, le nombre de colonnes dépendantes qui ont été détectées lorsque $P_e = 0.02$ et en fixant un seuil γ à 0.6.

Algorithme 1 : Recherche de matrices de rang probablement déficient

Entrées : la trame reçue $\mathbf{y}$, la valeur de l_{max} et M et le seuil γ fixé à 0.6
Sorties : la taille des mots de code : $\hat{n}$

pour $l = 2$ *à* l_{max} **faire**
  Construire $\tilde{R}_l$ de taille $M \times l$ avec $\mathbf{y}$;
  Trianguliariser $\tilde{R}_l$ => $T_l = A_l.\tilde{R}_l.B_l$;
  pour $i = 1$ *à* l **faire**
    Calculer : $N_l(i) = w(R_2^l.\mathbf{b}_i^l)$;
    si $N_l(i) \leq \frac{M-l}{2}.\gamma$ **alors**
      $Q(l) = Q(l) + 1$;
    fin
  fin
fin
$\mathcal{I} = \{l = 1, \cdots, l_{max} | Q(l) \neq 0\}$;
$\hat{n} = \text{mode}(\text{diff}(\mathcal{I}))$;



(a) Déficience de rang du $C(2, 1, 7)$ code avec $P_e = 0$ (b) Déficience de rang du $C(2, 1, 7)$ code avec $P_e = 0.02$

Figure 2.18 — Déficiences de rang du $C(2, 1, 7)$ code pour $P_e = 0$ et $P_e = 0.02$

Nous pouvons vérifier que pour $P_e = 0$, les matrices de taille $l = \alpha.n \geq n_a$ présentent une déficience de rang puisque $Q(l) \neq 0$. En revanche, pour $P_e = 0.02$, nous pouvons voir que très peu de colonnes dépendantes ont été détectées. Pour cette probabilité d'erreur du canal, nous avons récapitulé dans le tableau 2.2, les tailles des matrices pour lesquelles des colonnes dépendantes ont été détectées (soit l'ensemble $\mathcal{I}$ défini à l'équation (2.105)) ainsi que le nombre de ces colonnes (soit la variable $Q(l)$ définie à l'équation (2.104)).

$\mathcal{I}$ => l	22	26	28	30	32	36	38	42	44	46	52	56	60	66	84	86	90	92	100
$Q(l)$	2	1	1	1	1	2	1	1	1	1	1	2	1	1	2	1	4	2	1
diff($\mathcal{I}$)		4	2	2	2	4	2	4	2	2	6	4	4	6	18	2	4	2	8

Tableau 2.2 — Déficiences de rang détectées du $C(2, 1, 7)$ code pour $P_e = 0.02$

D'après ces valeurs et l'équation (2.107), d'après l'algorithme 1, nous obtiendrons une première estimation de $\hat{n}$: $\hat{n} = 2$.

Ce premier algorithme nous a permis de réaliser une première estimation de la taille des mots de code. Nous pouvons dans un premier temps, supprimer les valeurs de l'ensemble $\mathcal{I}$ qui ne sont pas multiple de $\hat{n}$. Nous noterons $\mathcal{I}$ l'ensemble défini par :

$$\mathcal{I} = \{l = 1, \cdots, l_{max} | Q(l) \neq 0 \cap l = \alpha.\hat{n}\} \tag{2.108}$$

Dans le but d'obtenir une meilleure estimation des tailles des matrices de rang déficient, nous avons vu précédemment qu'il était nécessaire de calculer le seuil γ optimal (2.101) qui dépend de la probabilité P. D'après (2.89), nous savons que la variable $N_l(i)$ est telle que :

$$N_l(i) \approx (M - l).P, \quad \forall N_l(i) \leq \frac{M - l}{2}.\gamma \tag{2.109}$$

Connaissant la variable $N_l(i)$, il nous sera possible d'estimer la probabilité P. En revanche, lors de l'étude du seuil, nous avons montré qu'il était plus difficile de détecter les relations de parité du code qui étaient de poids fort. Par conséquent, il est plus judicieux, pour estimer la probabilité P et le seuil optimal, d'utiliser les variables $N_l(i)$ qui ont été obtenues avec des vecteurs $\mathbf{b}_i^l$ qui sont de poids faible. Nous noterons s, la taille de la première matrice qui présente une variable $Q(l)$ nulle et $\mathcal{J}$ l'ensemble des colonnes dépendantes qui ont été détectées :

$$\mathcal{J} = \left\{ i = 1, \cdots, s | N_s(i) \leq \frac{M - s}{2}.\gamma \right\} \tag{2.110}$$

Dans cette configuration, la probabilité P estimée est telle que :

$$\hat{P} = \frac{1}{Q(s)} \sum_{i \in \mathcal{J}} \left(\frac{N_s(i)}{M - s} \right) \tag{2.111}$$

Une fois cette probabilité estimée, nous serons en mesure, d'après l'équation (2.101) d'estimer le seuil optimal, γ_{opt}. Avec ce seuil, nous pourrons affiner notre recherche sur les matrices qui sont probablement de rang déficient. En effet, les variables $N_l(i)$ ayant été sauvegardées, nous pourrons effectuer une recherche et délimiter les deux comportements de $N_l(i)$ avec le seuil optimal.

$$Q(l) = \text{Card} \left\{ i \in \{1, \cdots, l\} | N_l(i) \leq \frac{M - l}{2}.\gamma_{opt} \right\} \tag{2.112}$$

Ces différentes étapes qui sont représentées dans l'algorithme 2 permettent en sortie de refaire une estimation de la taille des mots de code en prenant les valeurs de $Q(l)$ obtenues avec le seuil γ_{opt}.

La taille des mots de code et la probabilité $\hat{P}$ étant connues, nous pourrons d'après l'équation 2.102 estimer la probabilité d'erreur du canal. Cette probabilité d'erreur dépend également des poids de Hamming des vecteur $\mathbf{b}_i^l$. Par conséquent, nous utiliserons également les relations $\mathbf{b}_i^l$ qui sont de poids faible. Nous noterons s la taille de la première matrice qui présente une variable $Q(l)$ nulle. Alors, la probabilité d'erreur du canal estimée, notée $\hat{P}_e$, est :

$$\hat{P}_e = \frac{1}{Q(s)} \sum_{i \in \mathcal{J}} \left(0.5 - 0.5 \exp \left(\frac{\log(1 - 2.\hat{P})}{w(\mathbf{b}_i^s)} \right) \right) \tag{2.113}$$

avec $\mathcal{J}$ qui est l'ensemble défini par :

$$\mathcal{J} = \left\{ i = 1, \cdots, s | N_s(i) \leq \frac{M - s}{2}.\gamma_{opt} \right\} \tag{2.114}$$

Algorithme 2 : Estimation de $\hat{n}$, γ_{opt} et $\hat{P}_e$

Entrées : $N_l(i)$, $\mathcal{I}$
Sorties : la taille des mots de code $\hat{n}$, le seuil optimal γ_{opt} et la probabilité d'erreur du canal $\hat{P}_e$

Calculer $\hat{P}$ (2.111);
Calculer γ_{opt} (2.101);
pour $l = 2$ *à* l_{max} **faire**
  pour $i = 1$ *à* l **faire**
    si $N_l(i) \leq \frac{M-l}{2}.\gamma$ **alors**
      $Q(l) = Q(l) + 1$;
    fin
  fin
fin
$\mathcal{I} = \{l = 1, \cdots, l_{max} | Q(l) \neq 0\}$;
$\hat{n} = \text{mode}(\text{diff}(\mathcal{I}))$;
Calculer $\hat{P}_e$ (2.113);

A partir de la connaissance de la probabilité d'erreur du canal $\hat{P}_e$ et de la taille des mots de code $\hat{n}$, nous pourrons nous intéresser à l'estimation d'une base du code dual afin d'en déduire la relation de parité de plus petit degré.

2.3.3.2 Estimation d'une base du code dual

Dans cette partie, nous mettrons en place un algorithme qui nous permettra de construire une base du code dual, soit un ensemble de relations de parité du code. Nous avons montré que la variable $N_l(i)$ suivait deux lois différentes selon que le vecteur candidat $\mathbf{b}_i^l$ appartenait ou non au code dual. D'après les différentes définitions que nous avons vu, nous déciderons que :

$$\begin{cases} \mathbf{b}_i^l \in C^{\perp} \text{ si } N_l(i) \leq \frac{M-l}{2}.\gamma_{opt} \\ \\ \mathbf{b}_i^l \notin C^{\perp} \text{ si } N_l(i) > \frac{M-l}{2}.\gamma_{opt} \end{cases} \tag{2.115}$$

Nous avons montré précédemment que le seuil optimal variait en fonction du poids de Hamming de $\mathbf{b}_i^l$, donc pour chaque vecteur candidat nous calculerons le seuil optimal qui lui correspond. Connaissant la probabilité d'erreur du canal, $\hat{P}_e$, nous serons en mesure pour chaque vecteur candidat de calculer la probabilité $\hat{P}$ qui lui est associée (2.102) :

$$\hat{P} = 1 - \frac{1 + (1 - 2.\hat{P}_e)^{w(\mathbf{b}_i^l)}}{2} \tag{2.116}$$

et d'en déduire le seuil optimal γ_{opt}. En sortie de cet algorithme 3, nous obtiendrons un ensemble, noté D, de relations de parité du code.

Afin d'estimer une matrice génératrice du code, ainsi que le nombre d'entrées et la mémoire du code, nous devons identifier parmi l'ensemble D la relation de parité de plus petit degré. Nous noterons $\hat{\mathbf{h}}$ cette relation de parité et $\hat{n}_a$ sa taille.

2.3.3.3 Estimation des paramètres k, K et d'une matrice génératrice

Dans un contexte non bruité, la taille de la relation de parité de plus petit degré correspond à la taille de la première matrice de rang déficient. Nous avons vu que cette valeur, notée n_a, était

Algorithme 3 : Estimation d'une base du code dual

Entrées : la séquence d'entrée $\mathbf{y}$, $\hat{n}$, $\hat{P}_e$ et $\mathcal{I}$
Sorties : une base du code dual : D

pour $l \in \mathcal{I}$ **faire**
  Construire $\tilde{R}_l$ de taille $M \times l$ avec $\mathbf{y}$;
  Trianguliariser $\tilde{R}_l \Rightarrow T_l = A_l.\tilde{R}_l.B_l$;
  pour $i = 1$ *à* l **faire**
    $\hat{P} = 1 - \frac{1+(1-2.\hat{P}_e)^{w(\mathbf{b}_i^l)}}{2}$;
    Calculer γ_{opt} (2.101);
    si $N_l(i) \leq \frac{M-l}{2}.\gamma_{opt}$ **alors**
      $D \longleftarrow D \cup \{\mathbf{b}_i^l\}$;
    fin
  fin
fin

telle que :

$$n_a = n.\left\lfloor \frac{\mu^{\perp}}{n-k} + 1 \right\rfloor \tag{2.117}$$

et le rang de la matrice de taille n_a est défini par :

$$rg(R_{n_a}) = n_a - Q(n_a) = n_a.\frac{k}{n} + \mu^{\perp} \tag{2.118}$$

Nous savons également que le nombre d'entrées d'un codeur, k, est compris entre 1 et $n-1$ et que la déficience de rang de la matrice de taille n_a, $Q(n_a)$, est comprise entre 1 et $n-k$. Par conséquent, en connaissant les valeurs de $\hat{n}$ et $\hat{n_a}$, nous serons en mesure de faire différentes hypothèses sur les valeurs de $\hat{\mu^{\perp}}$ et $\hat{k}$ possibles, voir algorithme 4. Dans l'algorithme 4, $\hat{\mu}^{\perp}$, $\hat{k}$ et $\hat{\mu}$ sont des vecteurs

Algorithme 4 : Estimation de $\hat{k}$ et $\hat{\mu^{\perp}}$

Entrées : $\hat{n}$ et $\hat{n_a}$
Sorties : $\hat{k}$, $\hat{\mu}^{\perp}$ et $\hat{\mu}$

pour $k = 1$ *à* $\hat{n} - 1$ **faire**
  pour $Q = 1$ *à* $n - k$ **faire**
    $\hat{\mu}^{\perp} = \left[\hat{\mu}^{\perp} \quad \hat{n_a}(1 - \frac{k}{\hat{n}}) - Q\right]$;
    $\hat{k} = \left[\hat{k} \quad k\right]$;
    $\hat{\mu} = \left[\hat{\mu} \quad \left\lceil \frac{\hat{\mu}^{\perp}}{\hat{k}} \right\rceil\right]$;
  fin
fin

ou un élément est ajouté à chaque nouvelle valeur de k et Q.

D'après la section 2.2.5, afin d'estimer une matrice génératrice du code, nous avons besoin de connaître les $(n-k)$ relations de parité du code. Ainsi, pour chaque couple $(\hat{k}, \hat{\mu})$, nous chercherons $(\hat{n} - \hat{k})$ relations de parité qui permettent d'obtenir une matrice génératrice d'un code optimal. Si pour un couple donné, nous obtenons l'ensemble de ces paramètres alors nous pourrons sortir de l'algorithme. En revanche, si en sortie nous n'obtenons pas les paramètres d'un code optimal, alors nous testerons les couples suivants. Nous savons qu'en pratique, la plupart des codes convolutifs optimaux utilisés sont de rendement $1/n$ ou $(n-1)/n$. De ce fait, nous pourrons tester en priorité

les couples tels que $\hat{k} = 1$ et $\hat{k} = \hat{n} - 1$.

Code de rendement $(n-1)/n$

Lors d'une transmission sans bruit, nous avons montré que la relation de parité du code qui était utile afin d'identifier une matrice génératrice du code correspondait à la relation de parité du code de degré $n.(\mu^{\perp}+1)$. Cette relation de parité correspond à la relation $\hat{\mathbf{h}}$ estimée précédemment. Par conséquent, afin d'estimer une matrice génératrice du code, il suffit d'appliquer l'algorithme présenté en section 2.2.5.

Code de rendement k/n

Nous avons montré précédemment que pour identifier une matrice génératrice du code, nous devions connaître les $(n-k)$ relations de parité de degré $(k+1).(\mu^{\perp}+1)$. Afin d'identifier ces $(n-k)$ relations, nous avions séparé les sorties du codeur afin de construire $(n-k)$ systèmes. Nous appliquerons, dans ce contexte bruité, le même principe afin d'identifier les $(\hat{n}-\hat{k})$ relations de parité du code. Nous noterons $\mathbf{x}_s$ un vecteur définie par :

$$\mathbf{x}_s = \begin{pmatrix} y_1(t) & \cdots & y_{\hat{k}}(t) & y_{\hat{k}+s}(t) & y_1(t+1) & \cdots & y_{\hat{k}}(t+1) & y_{\hat{k}+s}(t+1) & \cdots \end{pmatrix}, \ \forall s = 1, \cdots, \hat{n}-\hat{k} \tag{2.119}$$

Pour ces $(\hat{n}-\hat{k})$ vecteurs, nous allons construire les matrices $\tilde{R}_l^s$, de la même façon que les matrices $\tilde{R}_l$ étaient construites avec le vecteur $\mathbf{y}$. Puis, pour chacune de ces matrices, nous chercherons une relation linéaire qui sera de degré $(\hat{k}+1).(\hat{\mu^{\perp}}+1)$. En sortie de cet algorithme 5, nous obtiendrons les $(\hat{n}-\hat{k})$ relations de parité du code que nous noterons $\hat{\mathbf{h}}_s$.

Algorithme 5 : Estimation des $(\hat{n}-\hat{k})$ relations de parité

Entrées : la séquence d'entrée $\mathbf{y}$, $\hat{n}$, $\hat{k}$, $\hat{\mu^{\perp}}$, $\hat{P}_e$
Sorties : les $(\hat{n}-\hat{k})$ relations de parité $\hat{\mathbf{h}}_s$

pour $s = 1$ *à* $(\hat{n}-\hat{k})$ **faire**
  $\mathbf{x}_s = \begin{pmatrix} y_1(t) & \cdots & y_{\hat{k}}(t) & y_{\hat{k}+s}(t) & y_1(t+1) & \cdots & y_{\hat{k}}(t+1) & y_{\hat{k}+s}(t+1) & \cdots \end{pmatrix}$;
  pour $l = (\hat{k}+1).(\hat{\mu^{\perp}}+1)$ *à* l_{max} **faire**
    Construire $\tilde{R}_l^s$ de taille $M \times l$ avec $\mathbf{x}_s$;
    Trianguliariser $\tilde{R}_l^s \Rightarrow T_l = A_l.\tilde{R}_l.B_l$;
    pour $i = 1$ *à* l **faire**
      $\hat{P} = 1 - \frac{1+(1-2.\hat{P}_e)^{w(\mathbf{b}_i^l)}}{2}$;
      Calculer le seuil optimal ;
      si $N_l(i) \leq \frac{M-l}{2}.\gamma_{opt}$ **alors**
        si $\deg \mathbf{b}_i^l = (\hat{k}+1).(\hat{\mu^{\perp}}+1)$ **alors**
          $\hat{\mathbf{h}}_s = \mathbf{b}_i^l$;
        fin
      fin
    fin
  fin
fin

Estimation d'une matrice génératrice

Lors d'une transmission sans bruit, nous avons mis en place un algorithme qui permettait à partir de la connaissance des paramètres d'un code et des $(n-k)$ relations de parité $\mathbf{h}_s$, d'identifier une matrice génératrice. Cet algorithme présenté en section 2.2.5 permet d'identifier une (ou un ensemble) de matrice(s) génératrice(s) en construisant les $(n-k)$ matrices $D^{(s)}$ composées des bits des relations de parité $\mathbf{h}_s$ et en résolvant les $(n-k)$ systèmes suivant :

$$D^{(s)}.\begin{bmatrix}\mathbf{g}_{i,1} & \cdots & \mathbf{g}_{i,k} & \mathbf{g}_{i,k+s}\end{bmatrix}^T, \quad \forall s = 1, \cdots, n-k \tag{2.120}$$

Par conséquent, l'identification des matrices génératrices ne dépend plus des mots de code. En effet, elle dépend uniquement des paramètres et des relations de parité identifiés. Par conséquent, dans notre contexte bruité, nous utiliserons cette même méthode afin d'estimer une matrice génératrice du code.

Au final, si il n'y a eu aucune erreur sur l'estimation des paramètres précédents, nous obtiendrons en sortie une matrice génératrice du code. En revanche, si une ou plusieurs erreurs ont été faites lors de l'estimation des paramètres, nous serons face à deux situation possibles. En effet, soit les paramètres estimés permettent d'identifier une matrice génératrice d'un code optimal mais qui ne correspond pas au code utilisé au départ, soit l'algorithme n'est pas capable, avec ces paramètres, d'identifier une matrice génératrice d'un code optimal.

Cet algorithme d'identification aveugle a pour objectif, à partir de la seule connaissance d'un train binaire codé et bruité, de reconnaître l'ensemble des paramètres et une matrice génératrice d'un code convolutif optimal. Il existe deux situations possibles en sortie de cet algorithme :

1. Un code optimal a été identifié
2. Aucun code optimal n'a été identifié

Dans cette deuxième situation, il est possible d'augmenter la probabilité de détecter un code optimal en réalisant une nouvelle itération de l'algorithme. Pour chaque itération, les mêmes données $\mathbf{y}$ seront utilisées mais nous réaliserons une permutation sur les lignes des matrices $\tilde{R}_l$. Cette permutation va permettre de créer une nouvelle réalisation virtuelle des données sans nécessiter de nouvelle données. Durant la triangularisation à l'aide du pivot de Gauss, si les pivots sont erronés, ces erreurs vont être propager dans la matrice T_l. De ce fait, en réalisant une permutation sur les lignes, nous augmentons la probabilité d'obtenir des pivots non-erronés.

2.4 Analyse de l'algorithme

Dans cette partie, nous proposerons une analyse des performances de l'algorithme de reconnaissance aveugle présenté précédemment. Notre algorithme de reconnaissance aveugle a pour objectif final d'identifier l'ensemble des paramètres et une matrice génératrice d'un code convolutif. Par conséquent, les performances de notre algorithme seront évaluées en terme de probabilité de détection, noté $\mathcal{P}_{det}$. Dans notre contexte, la détection inclut l'identification complète du code convolutif. Rappelons tout d'abord les différentes hypothèses qui ont été faites lors de la conception de cet algorithme :

- Les codes convolutifs identifiés sont des codes optimaux
- Le train binaire reçue est synchronisée
- Le canal de transmission est un BSC
- Le train binaire reçue est composée d'au minimum $l_{max}.M$ bits

L'un des paramètres important dans notre algorithme de reconnaissance est le nombre de bits nécessaire pour identifier un code. En effet, il est important de montrer que cet algorithme pourrait être adapté dans un contexte réel. Il est donc nécessaire de tenir compte des débits proposés par les standards actuels et futurs. Prenons l'exemple de l'UMTS (Universal Mobile Telecommunications

System) [3GP05b] qui offre actuellement un débit de l'ordre des 2Mbps. En fixant les paramètres $l_{max} = 100$ et $M = 200$, notre algorithme nécessitera un train binaire de 20000 bits. Avec un débit de 2Mbps, cette trame serait obtenue en à peine 10ms. De plus, les débits sont amenés à être augmentés, par conséquent notre algorithme pourrait effectivement être adapté dans un contexte réel. Pour chaque simulation, 1000 tirages de Monte-Carlo ont été réalisés, où pour chaque essai la séquence des mots d'information et les erreurs ont été tirées aléatoirement. Notre algorithme étant un algorithme itératif, nous étudierons dans un premier temps l'influence de ces itérations sur les performances globales de notre algorithme.

Nous proposons de réaliser ces différentes études sur trois codes convolutifs :

– Le $C(2, 1, 7)$ code de matrice génératrice :

$$G = \begin{bmatrix} 133 & 171 \end{bmatrix} \tag{2.121}$$

– Le $C(3, 1, 4)$ code de matrice génératrice :

$$G = \begin{bmatrix} 13 & 15 & 17 \end{bmatrix} \tag{2.122}$$

– Le $C(3, 2, 3)$ code de matrice génératrice :

$$G = \begin{bmatrix} 7 & 4 & 1 \\ 2 & 5 & 7 \end{bmatrix} \tag{2.123}$$

2.4.1 Le gain apporté par notre algorithme itératif

Le nombre d'itérations nous permettra de trouver un compromis entre les performances de détection et le temps de calcul introduit au niveau du récepteur par cet algorithme. Étant donné le choix que nous avons fait de ne prendre en compte qu'une quantité relativement faible de données reçues (20000 bits) pour des raisons d'implémentation opérationnelle, le nombre d'itérations sera théoriquement limité afin de garantir une indépendance statistique durant le processus de triangularisation par la méthode du pivot de Gauss. Afin de délimiter le nombre d'itérations nécessaires pour atteindre une probabilité de détection optimale, nous fixerons le nombre d'itérations maximum à 50. Dans le but d'évaluer ce nombre d'itérations, nous noterons $\lambda_{x \to y}$, le gain obtenu entre l'itération x et y, tel que :

$$\lambda_{x \to y} = \frac{\mathcal{P}_{det}(y) - \mathcal{P}_{det}(x)}{\mathcal{P}_{det}(x)} \tag{2.124}$$

où $\mathcal{P}_{det}(i)$ est la probabilité de détecter le bon code à la i-ème itération. Nous exprimerons ce gain en pourcentage.

– Le $C(2, 1, 7)$ code convolutif

Nous avons représenté sur la figure 2.19 la probabilité de détection, $\mathcal{P}_{det}$, obtenue en fonction de la probabilité d'erreur du canal, P_e, pour les itérations 1, 5 et 10. Pour ce code, nous pouvons remarquer que dès la 5-ième itération, les performances optimales de notre algorithme sont quasiment atteintes. En effet, le gain entre la 5-ième et la 50-ième itération est proche de 0. En revanche, le gain entre l'itération 1 et 5, représenté dans le tableau 2.3, est très important. En effet, pour $P_e = 0.03$, $\lambda_{1 \to 5}$ est proche de 112%. Pour cette probabilité d'erreur du canal, après 1 itération $\mathcal{P}_{det}$ est proche de 0.4 et nous passons à 0.8 après 5 itérations.

Tableau 2.3 — Gain de détection pour le $C(2, 1, 7)$ code

P_e	0.01	0.02	0.03
$C(2, 1, 7) : \lambda_{1 \to 5}$ (%)	0.1%	13%	112%

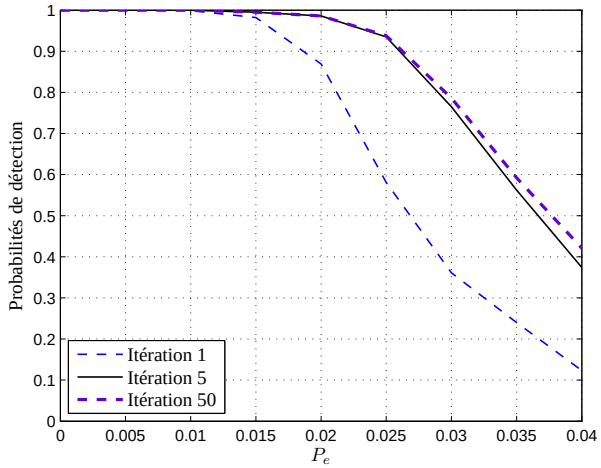


Figure 2.19 — $C(2,1,7)$: Probabilité de détection en fonction de P_e

– Le $C(3,2,3)$ code convolutif

Pour ce code, la probabilité de détection en fonction de la probabilité d'erreur du canal est représentée sur la figure 2.20 pour 1, 10 et 50 itérations. Nous constatons que pour ce code, il est nécessaire de réaliser 10 itérations afin d'atteindre les performances optimales. Le gain, représenté dans le tableau 2.4, entre l'itération 10 et 50 est proche de 0. Comme pour le $C(2,1,7)$ code, pour P_e supérieur à 0.02 les performances sont largement améliorées par ces itérations. En effet, le gain entre l'itération 1 et 10 pour $P_e = 0.02$ est proche des 150%.

Tableau 2.4 — Gain de détection pour le $C(3,2,3)$ code

P_e	0.01	0.02	0.03
$C(3,2,3)$: $\lambda_{1\to5}$ (%)	1%	150%	315%

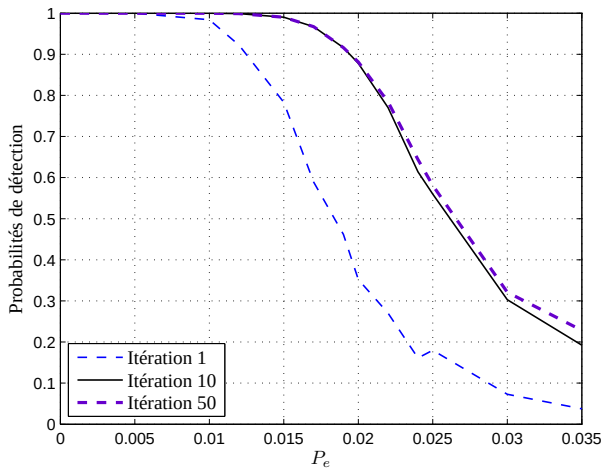


Figure 2.20 — $C(3,2,3)$: Probabilité de détection en fonction de P_e

– Le $C(3,1,4)$ code convolutif

Pour ce code, la probabilité de détection en fonction de la probabilité d'erreur du canal est représentée sur la figure 2.21 pour 1, 10 et 50 itérations. Pour ce code, nous remarquons que le gain apporté par les itérations est quasiment nul. En effet, dès la première itération les probabilités de détection maximales sont obtenues. Pour un code de rendement k/n, avec $k < n-1$, lors de la reconnaissance aveugle de la matrice génératrice (algorithme 5), nous réalisons implicitement de nouvelles itérations afin d'identifier les $(n-k)$ relations de parité. De ce fait, il ne sera pas nécessaire pour ces codes de réaliser beaucoup d'itérations afin d'atteindre les performances maximales de notre algorithme.

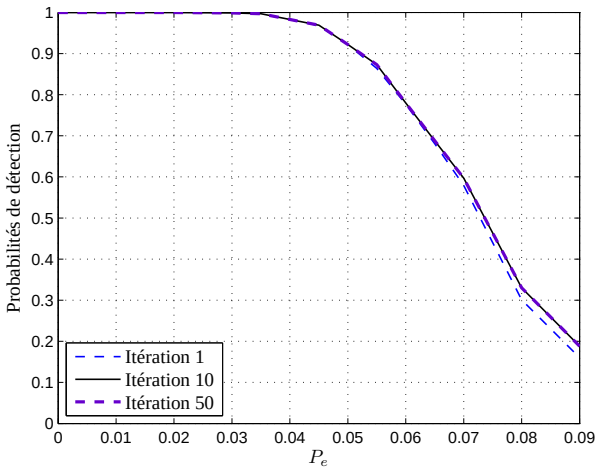


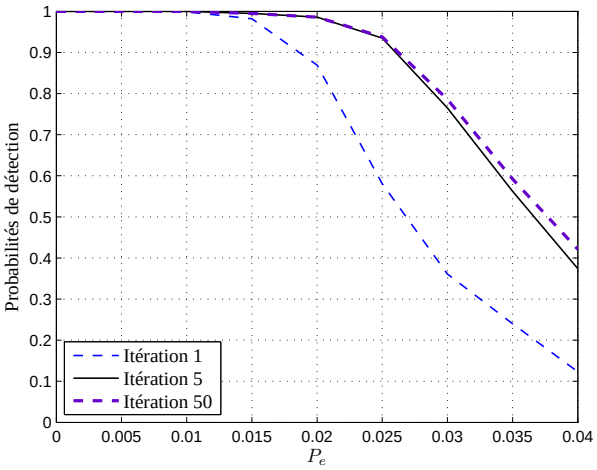
Figure 2.21 — $C(3,1,4)$: Probabilité de détection en fonction de P_e

Le nombre d'itérations réalisées est un paramètre important, en particuliers pour les codes de rendement $(n-1)/n$. En effet, les performances sont nettement améliorées par ce processus itératif. En revanche, la longueur de la trame reçue $\mathbf{y}$ étant faible, le nombre d'itérations permettant d'améliorer les performances de détection arrive à saturation au bout d'une dizaine d'itérations. Par conséquent, nous avons réalisé différents tests pour le $C(2,1,7)$ code en augmentant le nombre de données reçues.

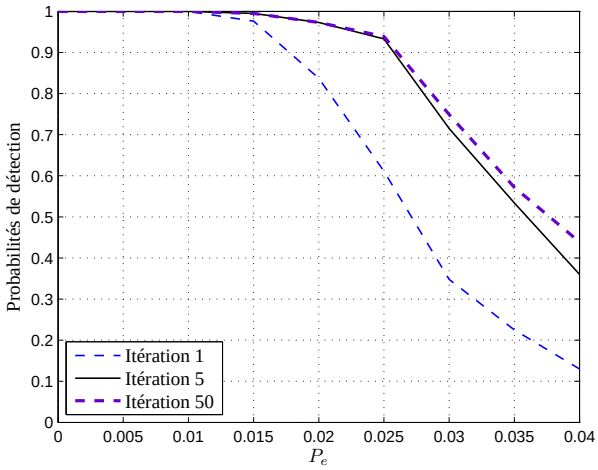
Nous avons remis sur la figure 2.22(a) les probabilités de détection en fonction de P_e que nous avons obtenu avec la méthode initiale. En revanche, sur la figure 2.22(b) les probabilités de détection représentées n'ont pas été obtenues avec la même méthode. Initialement, entre chaque itération l'ordre des lignes des matrices $\tilde{R}_l$ était permuté afin d'obtenir de nouveaux tirages virtuels. Or, pour cette seconde figure, les nouveaux tirages ont été obtenus avec des nouveaux mots de code. Ce qui implique qu'entre chaque itération, une nouvelle trame de 20000 bits est reçue. Nous remarquons qu'entre ces deux méthodes, les performances de détection sont similaires. Le fait de prendre un nouveau jeu de données entre chaque itération ne permet pas d'améliorer significativement les performances de détection de notre algorithme, par conséquent la méthode décrite initialement (permutation sur les lignes des matrices $\tilde{R}_l$) est un très bon compromis entre les performances de détection et le nombre de données nécessaires.

En reprenant la méthode initiale, nous avons représenté sur les figures 2.22(c) et 2.22(d), les probabilités de détection en fonction de P_e en augmentant la valeur de M. Pour la figure 2.22(c), nous avons fixé $l_{max} = 100$ et $M = 400$ alors que pour la figure 2.22(d), le paramètre M est fixé à 500. Ces différentes probabilités de détection ont été tracées pour 1, 10, 40 et 50 itérations. Pour $M = 400$ et $M = 500$, il est possible de réaliser un plus grand nombre d'itérations pour atteindre les performances optimales de notre algorithme. Dans ces deux cas, les performances optimales sont atteintes à la 40-ième itération (contre 5 lorsque $M = 200$). Nous pouvons également préciser

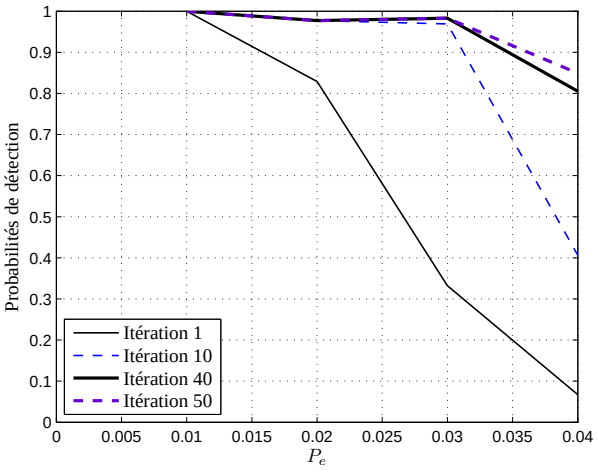
que les performances obtenues pour $M = 400$ et $M = 500$ sont similaires. Comparons maintenant les performances obtenues entre la figure 2.22(a) (pour $M = 200$) et 2.22(c) (pour $M = 400$). Nous remarquons que pour des probabilités d'erreur du canal supérieur à 0.03, les probabilités de détection sont améliorées lorsque la valeur de M croît. En effet, pour $P_e = 0.03$, nous passons de $\mathcal{P}_{det} = 0.76$ à $\mathcal{P}_{det} = 0.98$ et pour $P_e = 0.04$, la valeur de $\mathcal{P}_{det}$ passe de 0.37 à 0.8. Par conséquent, il est possible d'améliorer les performances de notre algorithme en augmentant la longueur de la trame reçue. Mais, comme nous l'avons précisé, il est également important de tenir compte des débits offerts par les standards. En fixant $M = 400$, il serait nécessaire de disposer d'une trame de 40000 bits et en reprenant le débit de l'UMTS, cette trame serait obtenue en 20ms. De plus, il est évident que le processus de triangularisation des matrices $\tilde{R}_l$ sera plus complexe lorsque M augmentera. Le fait de prendre $M = 200$ semble être un bon compromis entre le temps de calcul et les performances de détection de notre algorithme. En effet, nous montrerons par la suite que les performances de détection obtenues pour $M = 200$ sont très satisfaisantes. Un récapitulatif des différents gains obtenus par le processus itératif de notre algorithme est représenté dans le tableau 2.5.



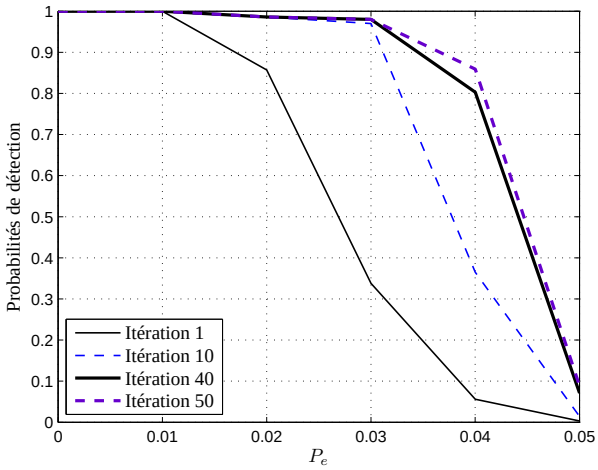
(a) Nombre d'itération pour $M = 200$



(b) Nombre d'itération pour $M = 200$ et des nouveaux tirages



(c) Nombre d'itération pour $M = 400$



(d) Nombre d'itération pour $M = 500$

Figure 2.22 — Nombre d'itérations pour le $C(2, 1, 7)$ code

D'après les différentes performances de détection que nous obtenons pour les trois codes étudiés, nous remarquons que les performances de détection sont différentes. En effet, l'identification aveugle

Tableau 2.5 — Gain de détection pour le $C(2,1,7)$ code avec plusieurs valeurs de M

P_e	0.01	0.02	0.03	0.04
$M = 200$: $\lambda_{1\to5}$ (%)	0.1%	13%	112%	204%
$M = 200$ et nouveaux tirages : $\lambda_{1\to5}$ (%)	0%	16%	105%	177%
$M = 400$: $\lambda_{1\to40}$ (%)	0%	37%	196%	1101%
$M = 500$: $\lambda_{1\to40}$ (%)	0%	15%	190%	1343%

du $C(3,2,3)$ est plus difficile que celle du $C(2,1,7)$ et du $C(3,1,7)$ code. Les performances de détection sont d'une part moins bonnes et il est nécessaire d'effectuer plus d'itérations pour atteindre ces performances. Mais ceci est simplement dû à la nature du code. En effet, le $C(3,2,3)$ introduit moins de redondance que les deux autres, il est donc évident que ce code sera plus difficile à identifier. Mais nous verrons également que ce code offre de moins bonnes performances en terme de correction des erreurs.

2.4.2 Les probabilités de détection

Afin d'analyser les performances de détection de notre algorithme, nous prendrons en considération trois probabilités :

1. Probabilité de détection $\mathcal{P}_{det}$: la probabilité d'identifier le bon code
2. Probabilité de fausse alarme $\mathcal{P}_{fa}$: la probabilité d'identifier un code optimal mais pas le bon
3. Probabilité de non détection $\mathcal{P}_{nd}$: la probabilité de n'identifier aucun code

Dans le but d'évaluer la pertinence de nos résultats, il est important de comparer les performances de détection au pouvoir de correction des codes. Pour cela, nous noterons TEB_r, le taux d'erreur binaire résiduel théorique obtenu après le décodage des mots bruités. Le canal utilisé pour simuler nos erreurs de transmission étant un canal binaire, les seules informations sont des données binaires. De ce fait, le décodeur de Viterbi utilisé sera un décodeur à décision dure [JZ99]. En regardant les standards actuels, comme l'UMTS par exemple, pour garantir une QoS en temps réel raisonnable, les taux d'erreur binaire résiduels doivent être compris entre 10^{-3} et 10^{-7} en zone urbaine et entre 10^{-3} et 10^{-4} en zone rurale. Par conséquent, nous considérerons ici que le TEB_r est acceptable si il est inférieur à 10^{-5}.

Nous avons représenté sur les figures 2.23, les TEB_r théoriques en fonction de la probabilité d'erreur du canal P_e pour les trois codes suivant : $C(2,1,7)$, $C(3,2,3)$ et $C(3,1,4)$. Le TEB_r théorique pour un code convolutif lors d'un décodage à décision dure est donné dans [JZ99] par l'expression ci-dessous :

$$TEB_r = \frac{1}{2}.\binom{d}{d/2} P_e^{d/2}.(1-P_e)^{d-d/2} + \sum_{i=d/2+1}^{d} \binom{d}{i} P_e^i.(1-P_e)^{d-i} \tag{2.125}$$

où d est la distance libre du code.

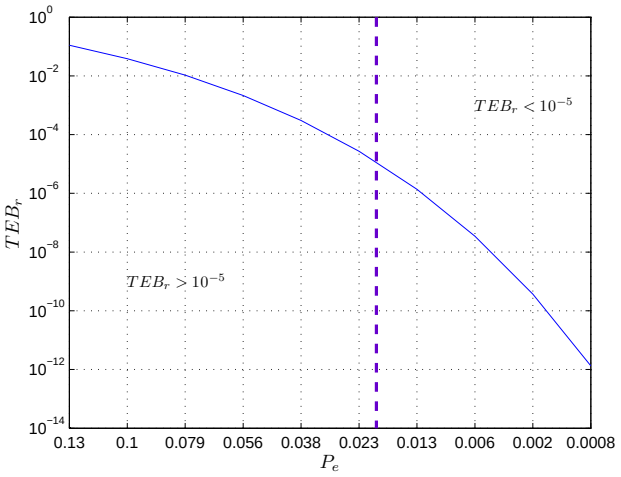
En règle générale, les courbes représentant les TEB sont tracées en fonction du rapport signal à bruit en dB, noté $\frac{E_b}{N_0}$. Avec E_b qui correspond à l'énergie reçue par bit et $N_0/2$ la densité spectrale du bruit Gaussien. Dans le but de pouvoir comparer les performances de détection de notre algorithme avec les TEB_r théoriques des codes, nous préférerons ici représenter ces courbes en fonction de la probabilité d'erreur du canal P_e. Le lien entre P_e et le rapport E_b/N_0 est tel que :

$$P_e = \mathrm{Q}\left(\sqrt{2.\frac{E_b}{N_0}.\frac{k}{n}}\right) \tag{2.126}$$

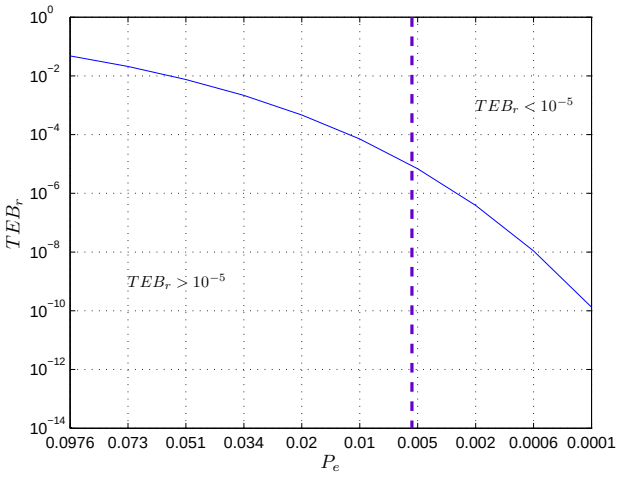
avec :

$$\mathrm{Q}(x) = \frac{1}{2}\mathrm{erfc}\left(\frac{x}{\sqrt{2}}\right), \tag{2.127}$$

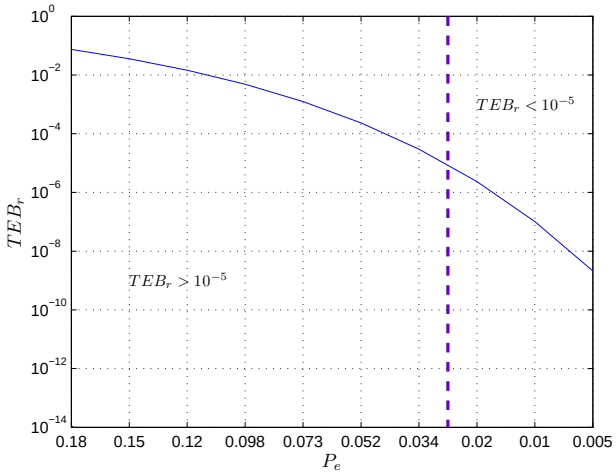
Les TEB_r ont été calculés pour une rapport E_b/N_0 variant de 1 à 10 dB.



(a) TEB_r théorique du $C(2, 1, 7)$ code



(b) TEB_r théorique du $C(3, 2, 3)$ code

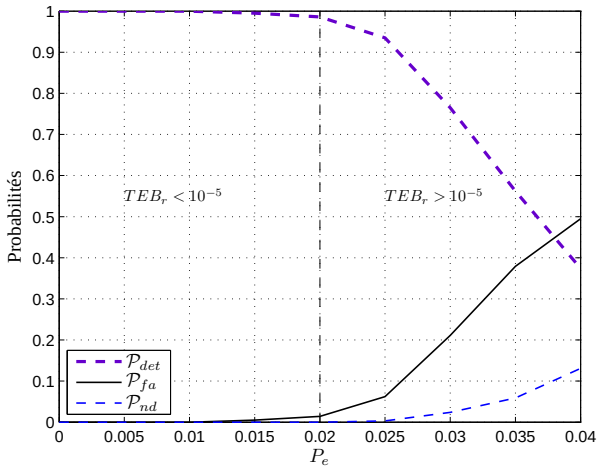


(c) TEB_r théorique du $C(3, 1, 4)$ code

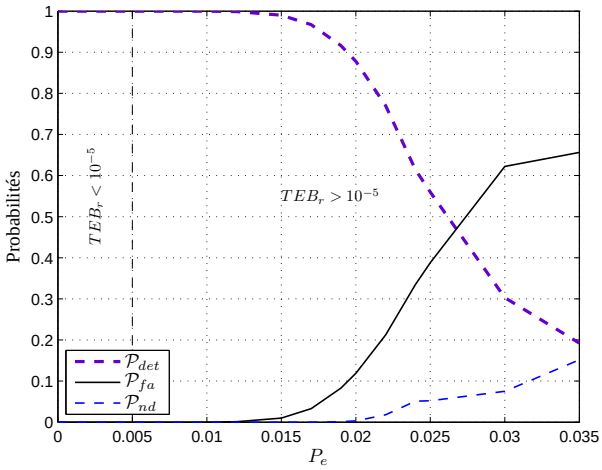
Figure 2.23 — Les TEB_r théoriques

Sur les figures 2.24, les trois probabilités ($\mathcal{P}_{det}$, $\mathcal{P}_{fa}$ et $\mathcal{P}_{nd}$) en fonction de la probabilité d'erreur du canal sont représentées pour les trois codes précédents. Nous avons également délimité, sur chaque figure, les zones où le TEB_r est supérieur à 10^{-5} de celles où il est inférieur. Pour les trois codes étudiés, nous notons que notre algorithme offre d'excellentes performances. En effet, la probabilité de détection est proche de 1 pour les zones correspondant à des TEB_r inférieur à 10^{-5}. Par conséquent, nous pouvons voir qu'il n'est pas nécessaire de disposer de plus de données afin d'obtenir d'excellentes performances de détection. En effet, comme nous l'avons remarqué avec le $C(2, 1, 7)$ en prenant $M = 400$ ou $M = 500$, pour des probabilités d'erreur du canal inférieures à 0.03, les probabilités de détection étaient proche de 1. Mais pour une telle probabilité d'erreur du canal, cet unique code convolutif ne serait pas utilisé en pratique puisque le taux d'erreur binaire résiduel après décodage serait proche de 10^{-4}.

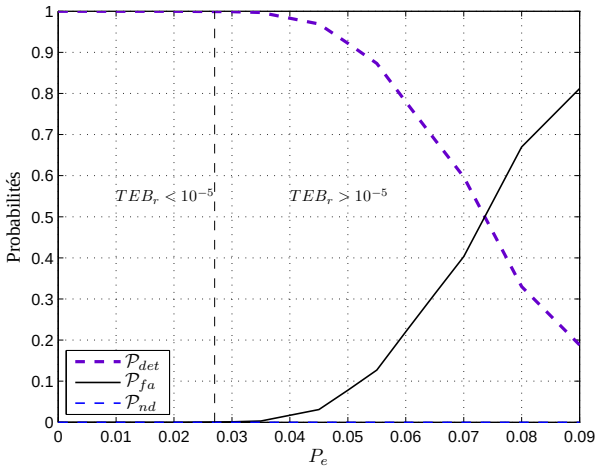
Pour ces trois codes, nous remarquons qu'à partir d'un certain seuil, les probabilités de fausse alarme (qui correspondent aux probabilités de détecter un code optimal mais pas le bon) deviennent supérieures aux probabilités de détection. En comparant le point de croisement de ces deux courbes aux TEB_r théoriques, nous remarquons que ces croisements se réalisent lorsque le TEB_r après décodage est proche des 10^{-3}. Par conséquent, il sera très rare de se retrouver dans une telle situation puisqu'un unique code ne sera pas utilisé dans ce contexte ($TEB_r > 10^{-3}$).



(a) Probabilités de détection du $C(2,1,7)$ code



(b) Probabilités de détection du $C(3,2,3)$ code



(c) Probabilités de détection du $C(3,1,4)$ code

Figure 2.24 — Les probabilités de détection

2.5 Conclusions

Dans ce chapitre nous avons présenté deux méthodes d'identification aveugle des codes convolutifs. La première méthode permet dans un contexte non-bruité d'identifier l'ensemble des paramètres et une matrice génératrice d'un code convolutif. Nous avons ensuite montré que cette méthode devenait très vite inefficace lorsque les mots de code étaient bruités. Par conséquent, nous avons développé une seconde méthode capable d'identifier en aveugle un code convolutif lorsque le train binaire reçu était bruité.

Nous avons ensuite analysé les performances de détection du second algorithme. Ce second

algorithme étant un algorithme itératif, nous avons tout d'abord mis en avant l'influence de ces itérations sur les performances de détection. Puis, nous avons réalisé une étude sur les performances globales de cet algorithme. Afin de montrer la pertinence de nos résultats, nous avons tenu compte des TEB résiduels après décodage. En considérant un TEB_r acceptable inférieur à 10^{-5}, nous avons montré qu'avec la seule connaissance d'une trame composée de 20000 bits, la probabilité d'identifier le bon code convolutif était proche de 1.

Récapitulatif des paramètres identifiés par notre algorithme :

- Les paramètres d'un code convolutif : n, k et K ;
- La matrice de parité d'un code ;
- Une matrice génératrice d'un code ;

Récapitulatif des points forts de notre algorithme :

- Une quantité relativement faible de données reçues est nécessaire (20000 bits) ;
- Une procédure itérative qui permet d'améliorer de manière significative les probabilités de détecter le bon code ;
- L'identification de l'ensemble des paramètres et d'une matrice génératrice d'un code convolutif optimal en présence d'erreurs de transmission ;
- Pour un TEB_r proche de 10^{-5} nous obtenons des probabilités de détection proche de 1 ;

Dans le prochain chapitre, nous étudierons tout d'abord une technique simple qui permettra d'augmenter le rendement d'un code convolutif. Puis, nous proposerons une méthode d'identification aveugle de ces nouveaux codes à rendement élevés.

CHAPITRE 3

Les codes convolutifs poinçonnés

3.1 Introduction

De nos jours, les chaînes de communication numérique nécessitent une robustesse et une rapidité de communication de plus en plus grande. En effet les téléphones mobiles, par exemple, ne sont plus simplement utilisés pour téléphoner mais ils permettent maintenant de naviguer sur internet, de regarder et/où de transmettre des vidéos, etc. Ces nouvelles applications nécessitent d'avoir des débits de plus en plus élevés. Dans une chaîne de communication classique, avant le bloc de codage canal, le codage de source permet de diminuer le débit nécessaire à la transmission en compressant les données mais le code correcteur d'erreur va ensuite augmenter ce débit en introduisant de la redondance. En effet, le débit utile après codage correspond au débit avant codage multiplié par le rendement du code. Les codes à rendement élevé (proche de 1) sont, en terme de débit, les meilleurs codes puisqu'ils introduisent moins de redondance mais du même coup perdent leurs intérêts d'un point de vue pouvoir de correction. Pour obtenir un code à rendement élevé et au pouvoir de correction convenable, il faudrait modifier la structure du bloc codage (comme dans le cas des codes concaténés) et ceci en augmentant alors la complexité du décodeur. De ce fait, dans la plupart des standards utilisant des codes convoltuifs, les codes utilisés sont de rendement peu élevé. Il existe une technique simple, appelée poinçonnage, qui permettra d'augmenter le rendement d'un code sans pour autant augmenter la complexité du décodeur. Dans ce chapitre, nous présenterons tout d'abord cette technique de poinçonnage, qui comme nous le verrons consiste simplement à ne pas transmettre tous les bits d'une trame. Nous développerons ensuite une méthode qui permettra d'identifier en aveugle les paramètres du code convolutif et du poinçonnage à partir de la seule connaissance de la trame poinçonnée. Enfin, nous proposerons d'analyser les performances que nous avons obtenu avec notre algorithme de reconnaissance aveugle dans le cadre d'une transmission bruitée.

3.2 Présentation des codes convolutifs poinçonnés

Les codes convolutifs poinçonnés ont été introduits par Cain et al. [CCG79]. Le poinçonnage permet par des techniques simples de construire de nouveaux codes à rendement élevé à partir d'anciens codes optimaux. Ces codes ont depuis connu un très grand succès, car ils permettent d'obtenir des codes à fort rendement tout en gardant une simplicité de décodage ainsi que des performances quasiment aussi bonnes qu'un code non poinçonné de même rendement. De nombreux résultats ont depuis été obtenus, notamment dans la recherche de codes à haut débit pouvant être décodés à l'aide de l'algorithme de Viterbi ([HB89], [BHP90], [BK97], [LS06]).

3.2.1 Principe du poinçonnage

Le principe d'un code convolutif est d'envoyer à chaque "top" d'horloge un mot d'information de k bits en entrée du codeur et il en ressort un mot de code de n bits. Il serait possible, d'envoyer $\mathcal{M}$ mots d'information de k bits en entrée et nous obtiendrions en sortie $\mathcal{M}$ mots de code de n bits. Nous passerions donc d'un codeur $C(n, k, K)$, que nous appellerons code parent, à un codeur $C^{[\mathcal{M}]}(\mathcal{M}.n, \mathcal{M}.k, K_p)$ qui est de rendement et de distance libre identiques à ceux du code parent et de longueur de contrainte, notée K_p, inférieure ou égale à celle du code parent, K. Cette étape consiste à effectuer un regroupement de profondeur $\mathcal{M}$. Le poinçonnage consiste à ne pas transmettre tous les bits constituant les mots de code. Ce poinçonnage est réalisé à partir d'une matrice de poinçonnage notée P qui est de taille $(n \times \mathcal{M})$. On note N' le nombre d'éléments à "1" dans la matrice P. Les éléments à "1" correspondent aux bits qui seront transmis et ceux à "0" aux bits qui ne seront pas transmis. Ce motif de poinçonnage est appliqué au $C^{[\mathcal{M}]}(\mathcal{M}.n, \mathcal{M}.k, K_p)$ code et nous obtenons un $C_p(N', \mathcal{M}.k, K_p)$ code, que nous appellerons code poinçonné, qui est un code de rendement plus élevé que celui du code parent. De plus, sa distance libre est très proche du codeur optimal de même paramètre (i.e sans poinçonnage) tout en ayant l'avantage d'avoir une faible complexité de décodage. En effet, le décodage se fera à partir du code parent en ne tenant pas compte des bits qui n'ont pas été transmis.

Exemple 3.33.

Prenons l'exemple d'un $C(2, 1, K)$ code parent. Le codage des mots d'information de 1 bit en leurs mots de code de 2 bits est représenté ci-dessous.

$$\begin{pmatrix} m_1(0) & m_1(1) & m_1(2) & m_1(3) & \cdots \end{pmatrix} \Rightarrow \begin{pmatrix} c_1(0) & c_1(1) & c_1(2) & c_1(3) & c_1(4) & \cdots \\ c_2(0) & c_2(1) & c_2(2) & c_2(3) & c_2(4) & \cdots \end{pmatrix}$$

En opérant un regroupement par bloc de profondeur $\mathcal{M} = 3$ sur les mots d'information :

$$\begin{pmatrix} m_1(0) & m_1(3) & \cdots \\ m_1(1) & m_1(4) & \cdots \\ m_1(2) & m_1(5) & \cdots \end{pmatrix}$$

puis sur les mots de code :

$$\begin{pmatrix} \begin{bmatrix} c_1(0) & c_1(1) & c_1(2) \\ c_2(0) & c_2(1) & c_2(2) \end{bmatrix} \begin{bmatrix} c_1(3) & c_1(4) & c_1(5) \\ c_2(3) & c_2(4) & c_2(5) \end{bmatrix} \begin{bmatrix} c_1(6) & c_1(7) & c_1(8) \\ c_2(6) & c_2(7) & c_2(8) \end{bmatrix} & \cdots \\ & \cdots \end{pmatrix}$$

on obtient le $C^{[3]}(3.2, 3.1, K_p) = C^{[3]}(6, 3, K_p)$ code qui est équivalent au code parent (même rendement et distance libre). La longueur de contrainte de ce code regroupé, K_p, dépend de la longueur de contrainte du code parent et de la profondeur du poinçonnage $\mathcal{M}$.

Prenons la matrice de poinçonnage P définie comme ci-dessous :

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

et appliquons ce motif de poinçonnage aux mots de code du $C^{[\mathcal{M}]}(6, 3, K_p)$ code.

$$\Rightarrow \begin{pmatrix} \begin{bmatrix} c_1(0) & & c_1(2) \\ c_2(0) & c_2(1) & \end{bmatrix} \begin{bmatrix} c_1(3) & & c_1(5) \\ c_2(3) & c_2(4) & \end{bmatrix} \begin{bmatrix} c_1(6) & & c_1(8) \\ c_2(6) & c_2(7) & \end{bmatrix} & \cdots \\ & \cdots \end{pmatrix}$$

En résumant le regroupement et le poinçonnage défini par la matrice P, nous obtenons la

transformation suivante :

$$\begin{pmatrix} m_1(0) & m_1(3) & m_1(6) & \cdots \\ m_1(1) & m_1(4) & m_1(7) & \cdots \\ m_1(2) & m_1(5) & m_1(8) & \cdots \end{pmatrix} \Rightarrow \begin{pmatrix} c_1(0) & c_1(3) & c_1(6) & \cdots \\ c_2(0) & c_2(3) & c_2(6) & \cdots \\ c_2(1) & c_2(4) & c_2(7) & \cdots \\ c_1(2) & c_1(5) & c_1(8) & \cdots \end{pmatrix},$$

soit le $C_p(4, 3, K_p)$ code avec $N' = 4$ (nombre de "1" dans P) et $\mathcal{M}.k = 3$.

Le codeur obtenu a un rendement plus élevé que celui du code parent, 3/4 contre 1/2, mais il garde la même simplicité de décodage que celui du code parent. En effet, le décodage se fera avec la matrice du code parent et à partir des mots de code suivant :

$$\begin{pmatrix} c_1(0) & X & c_1(2) & c_1(3) & X & c_1(5) & c_1(6) & X & c_1(8) & \cdots \\ c_2(0) & c_2(1) & X & c_2(3) & c_2(4) & X & c_2(6) & c_2(7) & X & \cdots \end{pmatrix}$$

où X correspond soit à un "1" soit à un "0", selon la définition du décodeur utilisé.

3.2.2 Construction du code poinçonné

Un code convolutif poinçonné peut être représenté par les paramètres de son code parent et le motif de poinçonnage, mais il peut également être représenté par les paramètres du code équivalent engendré par son code parent et le motif de poinçonnage en question. Nous noterons $C_p(n_p, k_p, K_p)$ le code convolutif équivalent après poinçonnage, qui est appelé code convolutif poinçonné, et qui sera représenté par une matrice génératrice, notée G_p. Cette matrice G_p dépendra de la matrice génératrice du code parent, G, et du motif de poinçonnage, P. La longueur de contrainte de ce code poinçonné, notée K_p, dépend de la longueur de contrainte du code parent K et de la profondeur du poinçonnage $\mathcal{M}$. Une première approche permettant d'obtenir cette matrice a été présentée dans [Hol91b] et [Hol91a], puis elle a été généralisée dans [McE98]. Afin d'obtenir la matrice génératrice du code poinçonné, il faut tout d'abord construire la matrice du code regroupé de profondeur $\mathcal{M}$, puis en opérant le poinçonnage sur cette matrice nous obtiendrons le code poinçonné.

- **Notation :**
  - $C(n, k, K)$: le code parent
  - P : la matrice de poinçonnage
  - $\mathcal{M}$: le nombre de bits permettant de générer un bloc de sortie, soit la profondeur du regroupement
  - N' : le nombre de bits non nuls dans la matrice P
  - $C_p(n_p, k_p, K_p)$: le code poinçonné (avec $n_p = N'$, $k_p = \mathcal{M}.k$ et $K_p \leq K$)

- **Code regroupé de profondeur $\mathcal{M}$:**

Nous avons vu qu'il était possible de passer d'un $C(n, k, K)$ code parent à un $C^{[\mathcal{M}]}(\mathcal{M}.n, \mathcal{M}.k, K_p)$ code que nous appellerons code regroupé. La matrice génératrice sera obtenue en réalisant une *décomposition polyphase* d'ordre $\mathcal{M}$ des polynômes générateurs du code parent. Cette décomposition consiste à associer à un polynôme un ensemble de $\mathcal{M}$ sous-polynômes.

Définition 3.16. *Soit $a(D)$ un polynôme d'indéterminée en D.*

$$a(D) = a_0 + a_1.D + a_2.D^2 + \cdots \tag{3.1}$$

Alors pour tout entier $\mathcal{M} \geq 1$, la décomposition polyphase d'ordre $\mathcal{M}$ de $a(D)$ est la liste des $\mathcal{M}$

sous-polynômes suivants :

$$(q_0(D), \cdots, q_{\mathcal{M}-1}(D)) = \left(\sum_{j\geq 0} a_{j.\mathcal{M}}.D^j, \cdots, \sum_{j\geq 0} a_{j.\mathcal{M}+(\mathcal{M}-1)}.D^j \right) \tag{3.2}$$

On appelle $q_j(D)$, la j-ème composante polyphase de $a(D)$. La j-ème composante polyphase de $a(D)$ est une série dont les coefficients sont obtenus en commençant au j-ème coefficient de $a(D)$ et tous les multiples de $\mathcal{M}$ de cette position.

Exemple 3.34.

Réalisons la décomposition polyphase d'ordre $\mathcal{M} = 3$ du polynôme $a(D) = 1+D^2+D^3+D^4+D^6$:

$$\begin{aligned} q_0(D) &= \sum_{j\geq 0} a_{3.j}.D^j = a_0 + a_3.D + a_6.D^2 = 1 + D + D^2 \\ q_1(D) &= \sum_{j\geq 0} .a_{3.j+1}.D^j = a_1 + a_4.D = D \\ q_2(D) &= \sum_{j\geq 0} a_{3.j+2}.D^j = a_2 + a_5.D = 1 \end{aligned}$$

Hole a montré dans [Hol91b] une autre façon de représenter la *décomposition polyphase* d'un polynôme. Pour cela, nous noterons $a(D^r)$, un polynôme en D^r. Notons $a(D)$ un polynôme défini par :

$$a(D) = a_0 + a_1.D + a_2.D^2 + \cdots + a_\mu.D^\mu \tag{3.3}$$

alors, le polynôme $a(D^r)$ est tel que :

$$a(D^r) = a_0.D^{r.0} + a_1.D^{r.1} + a_2.D^{r.2} + \cdots + a_\mu.D^{r.\mu} \tag{3.4}$$

Nous définirons $[j]_\mathcal{M}$ comme étant la classe des entiers j modulo $\mathcal{M}$, qui correspondent aux entiers de la forme $j + l.\mathcal{M}$ (pour $l = 0, 1, \cdots, \lfloor \frac{\mu - j}{\mathcal{M}} \rfloor$). Enfin, nous noterons $a_{[j]_\mathcal{M}}(D^r)$ un polynôme en D^r composé des coefficients $[j]_\mathcal{M}$ du polynôme $a(D)$. Les composantes polyphases d'ordre $\mathcal{M}$ de $a(D)$ sont définies par :

$$q_j(D) = D^{-j/\mathcal{M}}.a_{[j]_\mathcal{M}}(D^{1/\mathcal{M}}),\ \forall j = 0, \cdots, \mathcal{M}-1 \tag{3.5}$$

Les degrés des composantes polyphases d'ordre $\mathcal{M}$ d'un polynôme de degré μ sont tels que :

$$\begin{cases} \deg q_0(D) & \leq & \left\lfloor \frac{\mu}{\mathcal{M}} \right\rfloor \\ \deg q_1(D) & \leq & \left\lfloor \frac{\mu-1}{\mathcal{M}} \right\rfloor \\ \quad \vdots & & \\ \deg q_{\mathcal{M}-1}(D) & \leq & \left\lfloor \frac{\mu-(M-1)}{\mathcal{M}} \right\rfloor \end{cases} \tag{3.6}$$

Exemple 3.35.

Reprenons le polynôme de l'exemple 3.34, soit :

$$a(D) = 1 + D^2 + D^3 + D^4 + D^6$$

et réalisons une décomposition polyphase d'ordre $\mathcal{M} = 3$. Calculons tout d'abord la classe des entiers j modulo $\mathcal{M}$ pour $j = 0, \cdots, \mathcal{M} - 1$.

j	l	$[j]_{\mathcal{M}}$
0	$0,1,2$	$0,3,6$
1	$0,1$	$1,4$
2	$0,1$	$2,5$

Afin de calculer les composantes polyphases de $a(D)$, nous devons obtenir le polynôme $a(D^{1/\mathcal{M}})$:

$$\begin{aligned} a(D^{1/\mathcal{M}}) &= a_0 + a_1(D^{1/\mathcal{M}}) + a_2(D^{2/\mathcal{M}}) + a_3(D^{3/\mathcal{M}}) + a_4(D^{4/\mathcal{M}}) + a_5(D^{5/\mathcal{M}}) + a_6(D^{6/\mathcal{M}}) \\ &= a_0 + a_2(D^{2/3}) + a_3(D^{3/3}) + a_4(D^{4/3}) + a_6(D^{6/3}) \end{aligned}$$

D'après l'équation (3.5), les composantes polyphases de $a(D)$ sont telles que :

$$\begin{aligned} q_0(D) &= D^{-0/3}.a_{[0]_3}(D^{1/3}) = a_0 + a_3(D^{3/3}) + a_6(D^{6/3}) = 1 + D + D^2 \\ q_1(D) &= D^{-1/3.}a_{[1]_3}(D^{1/3}) = D^{-1/3}.\left(a_1(D^{1/3}) + a_4(D^{4/3})\right) = D \\ q_2(D) &= D^{-2/3}a_{[2]_3}(D^{1/3}) = D^{-2/3}\left(a_2(D^{2/3}) + a_5(D^{5/3})\right) = 1 \end{aligned}$$

Nous pouvons vérifier que nous obtenons les mêmes composantes polyphases que dans l'exemple 3.33.

Définition 3.17. *Soit $a(D) = a_0 + a_1.D + a_2.D^2 + \cdots$, une série causale d'indéterminée en D et $(q_0(D), q_1(D), \cdots, q_{\mathcal{M}-1}(D))$ la décomposition polyphase d'ordre $\mathcal{M}$ de $a(D)$. On appelle $\mathcal{Q}^{[\mathcal{M}]}(D)$, la $\mathcal{M}$-ième matrice polycyclique pseudocirculante (notée PCPC pour PolyCyclic PseudoCirculante) associée à $a(D)$, qui est de taille $\mathcal{M} \times \mathcal{M}$ et définie par :*

$$\mathcal{Q}^{[\mathcal{M}]}(D) = \begin{bmatrix} q_0(D) & q_1(D) & \cdots & q_{\mathcal{M}-2}(D) & q_{\mathcal{M}-1}(D) \\ D.q_{\mathcal{M}-1}(D) & q_0(D) & \cdots & q_{\mathcal{M}-3}(D) & q_{\mathcal{M}-2}(D) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ D.q_2(D) & D.q_3(D) & \cdots & q_0(D) & q_1(D) \\ D.q_1(D) & D.q_2(D) & \cdots & D.q_{\mathcal{M}-1}(D) & q_0(D) \end{bmatrix} \tag{3.7}$$

où les éléments des colonnes permutées en dessous de la diagonale sont tous multipliés par D.

Nous définirons le degré de la matrice $\mathcal{Q}^{[\mathcal{M}]}(D)$ comme étant le degré maximal de ces polynômes. Alors, d'après l'équation (3.6), le degré de cette matrice est tel que :

$$\deg \mathcal{Q}^{[\mathcal{M}]}(D) \leq \left\lfloor \frac{\mu - 1}{\mathcal{M}} \right\rfloor + 1 = \left\lceil \frac{\mu}{\mathcal{M}} \right\rceil \tag{3.8}$$

Exemple 3.36.

Suite de l'exemple 3.34.

La 3-ième matrice PCPC associée au polynôme $a(D)$ défini précédemment est telle que :

$$\mathcal{Q}^{[3]}(D) = \begin{bmatrix} 1 + D + D^2 & D & 1 \\ D & 1 + D + D^2 & D \\ D^2 & D & 1 + D + D^2 \end{bmatrix}$$

A partir du $C(n,k,K)$ code parent, on construit un code convolutif par regroupement de profondeur $\mathcal{M}$, le $C^{[\mathcal{M}]}(\mathcal{M}.n, \mathcal{M}.k, K_p)$ code. Le théorème présenté dans [McE98] permet de construire une matrice génératrice $G^{[\mathcal{M}]}(D)$ pour le $C^{[\mathcal{M}]}(\mathcal{M}.n, \mathcal{M}.k, K_p)$ code à partir de la matrice génératrice $G(D)$ du code parent.

Théorème 3.6. *On note $g_{i,j}(D)$ un polynôme générateur de la matrice génératrice $G(D)$, alors la matrice génératrice du $C^{[\mathcal{M}]}$ code, notée $G^{[\mathcal{M}]}(D)$, peut être obtenue en remplaçant les polynômes $g_{i,j}(D)$ par leurs $\mathcal{M}$-ième matrices polycycliques pseudocirculantes et en entrelaçant les lignes et les colonnes à la profondeur $\mathcal{M}$.*

D'après le théorème 3.6 et l'équation 3.8 la longueur de contrainte du code regroupé, notée K_p, est telle que :

$$K_p \leq \left\lceil \frac{\mu}{\mathcal{M}} \right\rceil + 1 \leq K \tag{3.9}$$

Exemple 3.37.

Prenons l'exemple d'un $C(2,1,3)$ code convolutif qui a pour matrice génératrice $G(D)$:

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

D'après le théorème 3.6 et les définitions précédentes, nous allons construire le $C^{[\mathcal{M}]}$ code pour $\mathcal{M} = 2$.

La 2-ième *décomposition polyphase* du premier polynôme $g_1(D) = 1 + D^2$ est :

$$(q_0(D), q_1(D)) = (1+D, 0)$$

et celle du second polynôme $g_2(D) = 1 + D + D^2$:

$$(q_0(D), q_1(D)) = (1+D, 1)$$

Notons $\mathcal{Q}_1^{[2]}(D)$ la 2-ième matrice PCPC du polynôme $g_1(D)$ et $\mathcal{Q}_2^{[2]}(D)$ celle du polynôme $g_2(D)$. D'après la décomposition polyphase des deux polynômes générateurs, les $\mathcal{M}$-ième PCPC sont :

$$\mathcal{Q}_1^{[2]}(D) = \begin{bmatrix} 1+D & 0 \\ 0 & 1+D \end{bmatrix} \text{ et } \mathcal{Q}_2^{[2]}(D) = \begin{bmatrix} 1+D & 1 \\ D & 1+D \end{bmatrix}$$

Notons $G'(D)$, la matrice composée de ces $\mathcal{M}$-ième PCPC :

$$G'(D) = \left[\begin{pmatrix} 1+D & 0 \\ 0 & 1+D \end{pmatrix} \begin{pmatrix} 1+D & 1 \\ D & 1+D \end{pmatrix} \right]$$

Alors, d'après le théorème 3.6, en entrelaçant les lignes à la profondeur de 2 (soit $(1,2) => (1,2)$) et les colonnes à la profondeur de 2 (soit $(1,2,3,4) => (1,3,2,4)$) de la matrice $G'(D)$, nous obtenons la matrice génératrice du code regroupé à la profondeur de 2 :

$$G^{[2]}(D) = \begin{bmatrix} 1+D & 1+D & 0 & 1 \\ 0 & D & 1+D & 1+D \end{bmatrix}$$

Nous obtenons au final un code regroupé de paramètres $C^{[2]}(\mathcal{M}.n, \mathcal{M}.k, K_p) = C^{[2]}(4,2,2)$ et de matrice génératrice $G^{[2]}(D)$. Nous pouvons vérifier que la longueur de contrainte du code regroupé, $K_p = 2$, est inférieure à la longueur de contrainte du code parent, $K = 3$.

- **Le poinçonnage :**

A partir de la matrice du code parent $G(D)$, nous avons obtenu la matrice génératrice du code regroupé de profondeur $\mathcal{M}$, $G^{[\mathcal{M}]}(D)$. Cette matrice du code regroupé a les mêmes propriétés que celle du code parent (canonique, même distance libre ...). Avec la matrice de poinçonnage P, nous allons obtenir la matrice génératrice du code poinçonné. La $G^{[\mathcal{M}]}(D)$ est une matrice de taille $(\mathcal{M}.k \times \mathcal{M}.n)$ et la matrice de poinçonnage P de taille $(n \times \mathcal{M})$. Le nombre de colonnes de la matrice $G^{[\mathcal{M}]}(D)$ correspond au nombre de coefficients dans la matrice P. Il existe donc une correspondance entre les colonnes de $G^{[\mathcal{M}]}(D)$ et les coefficients de P. Soit ϕ la bijection définie par :

$$(i, j) \rightarrow \phi(i, j) = i + n.(j - 1) \tag{3.10}$$

En associant chaque coefficient $P_{i,j}$ à la colonne $\phi(i, j)$ de $G^{[\mathcal{M}]}(D)$ nous allons obtenir la matrice génératrice du code poinçonné. Pour les coefficients $P_{i,j}$ nuls, les colonnes $\phi(i, j)$ de $G^{[\mathcal{M}]}(D)$ seront supprimées. Nous avons noté N' le nombre de coefficients non nuls dans P, donc en réalisant le poinçonnage sur la matrice $G^{[\mathcal{M}]}(D)$, nous obtenons une matrice de taille $\mathcal{M}.k \times N'$. Nous noterons $G_p(D)$ la matrice génératrice du code poinçonné et $C_p(N', \mathcal{M}.k, K_p)$ le code poinçonné, avec K_p définie à l'équation (3.9)

Exemple 3.38.

Suite de l'exemple 3.37.

Prenons comme motif de poinçonnage la matrice P définie ci-dessous.

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

D'après la bijection ϕ et la matrice P, il faut supprimer la colonne 3 de la matrice $G^{[2]}(D)$ ($P_{1,2} = 0$ et $\phi(1, 2) = 3$). La matrice génératrice $G_p(D)$ du code poinçonné associé au motif P est telle que :

$$G_p(D) = \begin{bmatrix} 1 + D & 1 + D & 1 \\ 0 & D & 1 + D \end{bmatrix}$$

Comparons maintenant les performances en terme de TEB_r du code parent ($C(2, 1, 3)$), du code poinçonné ($C_p(3, 2, 2)$) et du code optimal le plus proche du code poinçonné ($C(3, 2, 3)$). Le $C(3, 2, 3)$ optimal que nous avons utilisé a pour matrice génératrice :

$$G = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 7 \end{pmatrix} \tag{3.11}$$

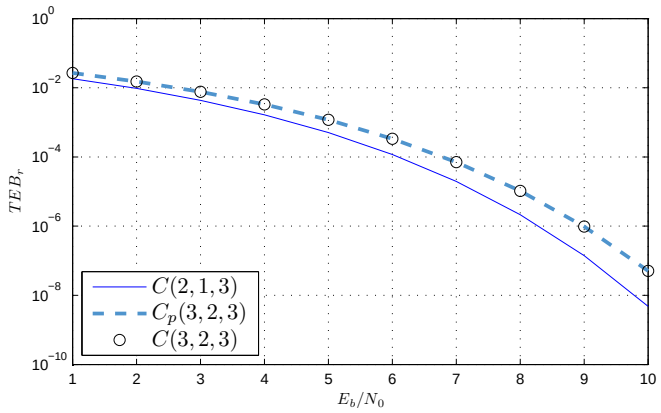


Figure 3.1 — Comparaison des TEB_r entre les codes poinçonnés et non poinçonnés

En comparant le TEB_r du code poinçonné et celui du code parent sur la figure 3.1, on remarque

que le poinçonnage dégrade légèrement les performances. En revanche, on remarque que les TEB_r du code poinçonné et celui du code optimal de même rendement et de longueur de contrainte la plus proche sont identiques. Le poinçonnage a permis par une technique simple d'augmenter le rendement du code sans pour autant augmenter la complexité du décodeur et dégrader de manière significative ses performances.

Réalisons maintenant un exemple de poinçonnage sur un code parent de rendement k/n avec $k > 1$.

Exemple 3.39.

Prenons l'exemple d'un $C(3, 2, 3)$ code de matrice génératrice :

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

Nous allons réaliser un poinçonnage de profondeur 2 avec le motif de poinçonnage suivant :

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

Le tableau, présenté ci-dessous, résume les *décompositions polyphases* d'ordre 2 ainsi que les matrices PCPC de chaque polynôme générateur du code parent.

$g_{i,j}(D)$	*décomposition polyphase* d'ordre 2	PCPC
$g_{1,1}(D) = 1+D+D^2$	$\begin{bmatrix} 1+D & 1 \end{bmatrix}$	$\begin{bmatrix} 1+D & 1 \\ D & 1+D \end{bmatrix}$
$g_{1,2}(D) = 1$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$g_{1,3}(D) = D^2$	$\begin{bmatrix} D & 0 \end{bmatrix}$	$\begin{bmatrix} D & 0 \\ 0 & D \end{bmatrix}$
$g_{2,1}(D) = D$	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ D & 0 \end{bmatrix}$
$g_{2,2}(D) = 1+D^2$	$\begin{bmatrix} 1+D & 0 \end{bmatrix}$	$\begin{bmatrix} 1+D & 0 \\ 0 & 1+D \end{bmatrix}$
$g_{2,3}(D) = 1+D+D^2$	$\begin{bmatrix} 1+D & 1 \end{bmatrix}$	$\begin{bmatrix} 1+D & 1 \\ D & 1+D \end{bmatrix}$

En regroupant les matrices PCPC des polynômes générateurs, on obtient la matrice $G'(D)$ telle que :

$$G'(D) = \begin{bmatrix} \begin{pmatrix} 1+D & 1 \\ D & 1+D \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ D & 0 \end{pmatrix} & \begin{pmatrix} 1+D & 0 \\ 0 & 1+D \end{pmatrix} & \begin{pmatrix} 1+D & 1 \\ D & 1+D \end{pmatrix} \end{bmatrix}$$

Réalisons l'entrelacement de profondeur 2 sur les lignes de cette matrice : $(1, 2, 3, 4) => (1, 3, 2, 4)$

$$=> \begin{bmatrix} 1+D & 1 & 1 & 0 & D & 0 \\ 0 & 1 & 1+D & 0 & 1+D & 1 \\ D & 1+D & 0 & 1 & 0 & D \\ D & 0 & 0 & 1+D & D & 1+D \end{bmatrix}$$

puis sur les colonnes : $(1, 2, 3, 4, 5, 6) => (1, 3, 5, 2, 4, 6)$ afin d'obtenir la matrice génératrice du

$C^{[2]}$ code :

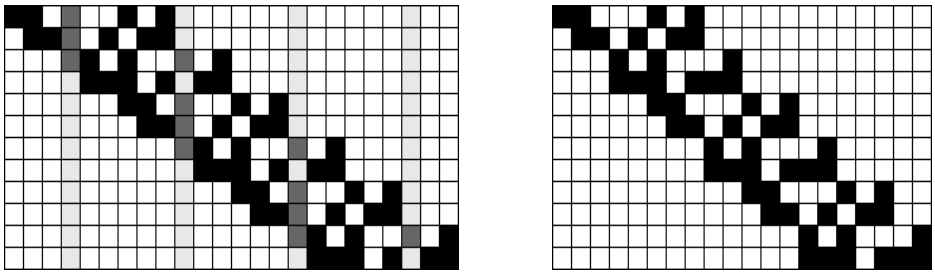
$$G^{[2]}(D) = \begin{bmatrix} 1+D & 1 & D & 1 & 0 & 0 \\ 0 & 1+D & 1+D & 1 & 0 & 1 \\ D & 0 & 0 & 1+D & 1 & D \\ D & 0 & D & 0 & 1+D & 1+D \end{bmatrix}$$

D'après le motif de poinçonnage et la bijection ϕ, il faut supprimer la colonne 4 de $G^{[2]}(D)$ ($P(2,1) = 0$ et $\phi(2,1) = 4$) :

$$G_p(D) = \begin{bmatrix} 1+D & 1 & D & 0 & 0 \\ 0 & 1+D & 1+D & 0 & 1 \\ D & 0 & 0 & 1 & D \\ D & 0 & D & 1+D & 1+D \end{bmatrix}$$

Nous obtenons donc la matrice génératrice du code poinçonné, soit le $Cp(5,4,2)$ code.

La figure 3.2 représente les matrices de codage du code parent et du code poinçonné. Sur ces figures, les bits à "1" sont représentés par les cases noires et les bits à "0" par les cases blanches.



(a) Code parent (b) Code poinçonné

Figure 3.2 — Matrice de codage du code parent et du code poinçonné

Sur la figure représentant la matrice de codage du code parent, les colonnes grises correspondent aux colonnes qui ont été supprimées pour obtenir le code poinçonné. Visuellement, nous remarquons effectivement qu'en suppriment ces colonnes grises nous obtenons la matrice de codage du code poinçonnage.

3.3 Reconnaissance aveugle d'un code convolutif poinçonné

Un code convolutif poinçonné pouvant être représenté par un code qui est équivalent au code parent et au poinçonnage, la reconnaissance aveugle de ce code poinçonné ne diffère pas de celle d'un code classique. Ainsi, les algorithmes présentés dans le chapitre 2 vont nous permettre d'identifier les paramètres du code poinçonné, soit le $C_p(n_p, k_p, K_p)$ code, et une matrice génératrice de ce code, $G_p(D)$. Dans la littérature, la plupart des algorithmes permettant d'identifier un code convolutif poinçonné s'arrêtent à cette étape ([Fil00] et [Bar07b]). En effet, l'unique connaissance de ce code peut permettre de décoder les mots de code afin d'obtenir les mots d'information. En revanche, nous avons vu que l'avantage d'utiliser le poinçonnage était d'augmenter le débit de la transmission sans pour autant augmenter la complexité du décodeur. Or, en décodant avec le code poinçonné, le décodage sera plus complexe puisque le rendement du code poinçonné est strictement supérieur au rendement du code parent. De ce fait, l'utilisation du poinçonnage perdrait son avantage au niveau du récepteur. L'intérêt est donc de réussir à mettre en oeuvre un algorithme, qui avec la seule connaissance du code poinçonné, serait capable d'identifier les paramètres du code parent et du motif de poinçonnage.

Une première approche permettant d'identifier le code parent et le motif de poinçonnage a été présenté dans [SLLZ04] et [LLZL05]. Dans ces articles, plusieurs hypothèses étaient faites concernant le code parent et le motif de poinçonnage. En effet, l'algorithme présenté fonctionne uniquement pour les codes parents de rendement $1/2$ et lorsque le motif de poinçonnage P est tel que :

$$P = \begin{pmatrix} 1 & \cdots & 1 & 1 \\ 0 & \cdots & 0 & 1 \end{pmatrix} \tag{3.12}$$

Dernièrement, un algorithme de reconnaissance aveugle de code convolutif poinçonné a également été proposé dans [CF09] dans le cas d'un code parent de rendement $1/n$.

Nous proposerons dans cette partie une méthode permettant à partir de la seule connaissance du code poinçonné (qui aura au préalable été identifiée avec les méthodes du chapitre 2) de retrouver le code parent et le motif de poinçonnage dans le cas général d'un code parent de rendement k/n. Avant de présenter cette méthode, nous proposons d'expliquer le principe de cet algorithme de reconnaissance.

3.3.1 Principe de la méthode de reconnaissance

Il a été montré, dans [Hol91b], le lien entre les polynômes générateurs du code parent et leurs matrices PCPC. Nous noterons $\mathcal{Q}_{i,j}^{[\mathcal{M}]}$, la $\mathcal{M}$-ième matrice *polycyclique pseudocirculante* du (i,j)-ème polynôme générateur définie par :

$$\mathcal{Q}_{i,j}^{[\mathcal{M}]} = \begin{bmatrix} q_{i,j_{(1,1)}}(D) & q_{i,j_{(1,2)}}(D) & \cdots & q_{i,j_{(1,\mathcal{M})}}(D) \\ \vdots & \vdots & \cdots & \vdots \\ q_{i,j_{(\mathcal{M},1)}}(D) & q_{i,j_{(\mathcal{M},2)}}(D) & \cdots & q_{i,j_{(\mathcal{M},\mathcal{M})}}(D) \end{bmatrix} \tag{3.13}$$

Nous définirons la matrice β de taille $(\mathcal{M} \times \mathcal{M})$ telle que :

$$\beta = \begin{pmatrix} \mathcal{M} & \mathcal{M}+1 & \cdots & \mathcal{M}+(\mathcal{M}-1) \\ \mathcal{M}-1 & \mathcal{M} & \cdots & \mathcal{M}+(\mathcal{M}-2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \cdots & \mathcal{M} \end{pmatrix} \tag{3.14}$$

Alors le lien entre le (i,j)-ème polynôme générateur $g_{i,j}(D)$ et sa $\mathcal{M}$-ième matrice PCPC est :

$$g_{i,j}(D) = \sum_{m=1}^{\mathcal{M}} D^{\beta(m,l)-\mathcal{M}} \,.\, q_{i,j_{(m,l)}}(D^{\mathcal{M}}) \quad \forall l = 1, \cdots, \mathcal{M} \tag{3.15}$$

où $\beta(m,l)$ désigne l'élément sur la m-ième ligne et la l-ième colonne de la matrice β. D'après cette équation 3.15, il est possible avec une seule colonne de la $\mathcal{M}$-ième PCPC d'obtenir le polynôme générateur correspondant à cette PCPC.

Exemple 3.40.

Suite de l'exemple 3.37 :

Nous avons défini les 2-ièmes matrices PCPC de ce codeur comme étant :

$$\mathcal{Q}_{1,1}^{[2]}(D) = \begin{bmatrix} 1+D & 0 \\ 0 & 1+D \end{bmatrix} \text{ et } \mathcal{Q}_{1,2}^{[2]}(D) = \begin{bmatrix} 1+D & 1 \\ D & 1+D \end{bmatrix}$$

La matrice β pour $\mathcal{M} = 2$ est telle que :

$$\beta = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}$$

D'après l'équation (3.15), pour $j = 1$ et 2, nous obtenons pour $l = 1$:

$$\begin{aligned} g_{1,j}(D) &= \sum_{m=1}^{2} D^{\beta(m,1)-2} \,.\, q_{1,j_{(m,1)}}(D^2) \\ &= D^0.q_{1,j_{(1,1)}}(D^2) + D^{-1}.q_{1,j_{(2,1)}}(D^2) \end{aligned}$$

Soit, les polynômes $g_{1,1}(D)$ et $g_{1,2}(D)$ qui sont tels que :

$$\begin{aligned} g_{1,1}(D) &= D^0.(1 + D^2) + D^{-1}.(0) = 1 + D^2 \\ g_{1,2}(D) &= D^0.(1 + D^2) + D^{-1}.(D^2) = 1 + D + D^2 \end{aligned}$$

Ces polynômes peuvent également être obtenus en posant $l = 2$:

$$\begin{aligned} g_{1,j}(D) &= \sum_{m=1}^{2} D^{\beta(m,2)-2} \,.\, q_{1,j_{(m,2)}}(D^2) \\ &= D^1.q_{1,j_{(1,1)}}(D^2) + D^0.q_{1,j_{(2,1)}}(D^2) \end{aligned}$$

$$\begin{aligned} g_{1,1}(D) &= D^1.(0) + D^0.(1 + D^2) = 1 + D^2 \\ g_{1,2}(D) &= D^1.(1) + D^0.(1 + D^2) = 1 + D + D^2 \end{aligned}$$

Nous pouvons vérifier qu'avec une colonne des PCPC il est possible d'obtenir le polynôme générateur correspondant.

Exemple 3.41.

Suite de l'exemple 3.39 :
Reprenons le $C(3, 2, 3)$ code parent et leurs 2-ième PCPC. La matrice β pour $\mathcal{M} = 2$ est telle que :

$$\beta = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}$$

Pour $l = 1$:

$$\begin{aligned} g_{i,j}(D) &= \sum_{m=1}^{2} D^{\beta(m,1)-2} \,.\, q_{i,j_{(m,1)}}(D^2) \\ &= D^0.q_{i,j_{(1,1)}}(D^2) + D^{-1}.q_{i,j_{(2,1)}}(D^2) \end{aligned}$$

nous obtenons les $k \times n$ polynômes générateurs suivants :

$$\begin{aligned} g_{1,1}(D) &= (1 + D^2) + D^{-1}.(D^2) = 1 + D + D^2 \\ g_{1,2}(D) &= (1) + D^{-1}.(0) = 1 \\ g_{1,3}(D) &= (D^2) + D^{-1}.(0) = D^2 \\ g_{2,1}(D) &= (0) + D^{-1}.(D^2) = D \\ g_{2,2}(D) &= (1 + D^2) + D^{-1}.(0) = 1 + D^2 \\ g_{2,3}(D) &= (1 + D^2) + D^{-1}.(D^2) = 1 + D + D^2 \end{aligned}$$

Pour $l = 2$:

$$\begin{aligned} g_{i,j}(D) &= \sum_{m=1}^{2} D^{\beta(m,2)-2} \,.\, q_{i,j_{(m,2)}}(D^2) \\ &= D^1.q_{i,j_{(1,2)}}(D^2) + D^0.q_{i,j_{(2,2)}}(D^2) \end{aligned}$$

les polynômes générateurs obtenus sont tels que :

$$\begin{aligned} g_{1,1}(D) &= D.(1) + (1 + D^2) = 1 + D + D^2 \\ g_{1,2}(D) &= D.(0) + (1) = 1 \\ g_{1,3}(D) &= D.(0) + (D^2) = D^2 \\ g_{2,1}(D) &= D.(1) + (0) = D \\ g_{2,2}(D) &= D.(0) + (1 + D^2) = 1 + D^2 \\ g_{2,3}(D) &= D.(1) + (1 + D^2) = 1 + D + D^2 \end{aligned}$$

Nous vérifions que les polynômes générateurs obtenus pour $l = 1$ et $l = 2$ sont identiques et qu'ils correspondent aux polynômes du code parent.

Nous venons de montrer qu'avec une seule colonne de la $\mathcal{M}$-ième PCPC, il était possible d'obtenir le polynôme générateur correspondant à cette PCPC. Or, la matrice du code poinçonné, $G_p(D)$, est composée de certaines colonnes des $\mathcal{M}$-ième PCPC. Par conséquent, il sera possible à partir de la matrice du code poinçonné, de retrouver la matrice génératrice du code parent, ainsi que la matrice de poinçonnage. Les codes poinçonnés étant déterminés par recherche exhaustive, une liste de quelques "bons" codes poinçonnés a été regroupée en annexe E.

D'après les différentes définitions que nous venons de voir, nous présenterons dans les prochaines parties notre algorithme de reconnaissance. Nous exposerons tout d'abord cette méthode dans le cas d'un codeur de rendement $1/n$, puis nous la généraliserons au code de rendement k/n.

3.3.2 Reconnaissance aveugle pour un code parent de rendement $1/n$

Faisons l'hypothèse que le code parent est de rendement $1/n$, soit $k = 1$. Dans ce cas, le rendement du code poinçonné, k_p/n_p, est tel que $k_p = \mathcal{M}.k = \mathcal{M}$. De ce fait, après l'identification du code poinçonné, nous connaîtrons la valeur de k_p et par conséquent la profondeur du poinçonnage $\mathcal{M} = k_p$. A partir de la matrice génératrice du code poinçonné, nous allons développer un algorithme itératif qui permettra d'identifier le code parent et le motif de poinçonnage. Rappelons tout d'abord le principe de construction du code poinçonné.

Notons $G'(D)$ la matrice composée des $\mathcal{M}$-ième PCPC des n polynômes générateurs du code parent :

$$G'(D) = \begin{bmatrix} \mathcal{Q}_{1,1}^{[\mathcal{M}]}(D) & \mathcal{Q}_{1,2}^{[\mathcal{M}]}(D) & \cdots & \mathcal{Q}_{1,n}^{[\mathcal{M}]}(D) \\ \vdots & \vdots & \cdots & \vdots \\ \mathcal{Q}_{k,1}^{[\mathcal{M}]}(D) & \mathcal{Q}_{k,2}^{[\mathcal{M}]}(D) & \cdots & \mathcal{Q}_{k,n}^{[\mathcal{M}]}(D) \end{bmatrix} \tag{3.16}$$

où les matrices $\mathcal{Q}_{1,j}^{[\mathcal{M}]}(D)$ sont de taille $\mathcal{M} \times \mathcal{M}$. La matrice du code regroupé $(G^{[\mathcal{M}]}(D))$ de profondeur $\mathcal{M}$ est obtenue en entrelaçant les colonnes de $G'(D)$ à la profondeur de $\mathcal{M}$. Enfin, la matrice du code poinçonné est obtenue en supprimant certaines colonnes de $G'(D)$. Cette matrice, $G_p(D)$, est telle que :

$$G_p(D) = \begin{bmatrix} g_{p_{(1,1)}}(D) & \cdots & g_{p_{(1,n_p)}}(D) \\ \vdots & \cdots & \vdots \\ g_{p_{(k_p,1)}}(D) & \cdots & g_{p_{(k_p,n_p)}}(D) \end{bmatrix} \tag{3.17}$$

où $g_{p_{(i,j)}}(D)$ est le (i,j)-ème polynôme de $G_p(D)$, $\forall i = 1, \cdots, k_p$ et $\forall j = 1, \cdots, n_p$.

Nous pouvons vérifier que chaque colonne de cette matrice correspond à une colonne d'une des $\mathcal{M}$-ième PCPC des polynômes générateurs du code parent. Or, en connaissant une colonne de la $\mathcal{M}$-ième PCPC d'un polynôme, nous sommes capable d'identifier le polynôme générateur associé à cette $\mathcal{M}$-ième PCPC. De ce fait, il nous sera possible à partir de la matrice $G_p(D)$ d'identifier les polynômes générateurs du code parent.

Le principe de notre algorithme est de prendre la première colonne de $G_p(D)$ et de calculer le polynôme qui lui est associé, d'après l'équation (3.15). Notons $q'(D)$ ce polynôme qui est tel que :

$$q'(D) = \sum_{m=1}^{\mathcal{M}} D^{\beta(m,l)-\mathcal{M}} \, . \, g_{p_{(m,i)}}(D^{\mathcal{M}}), \quad \forall i = 1, \cdots, n_p \tag{3.18}$$

La valeur de l peut varier entre 1 et $\mathcal{M}$. Par conséquent, nous essayerons tout d'abord de calculer le polynôme $q'(D)$ avec $l = 1$. Si le polynôme obtenu est un polynôme contenant des puissances négatives, alors nous le recalculerons mais en incrémentant la valeur de l (qui peut varier de 1 à $\mathcal{M}$). Une fois un polynôme $q'(D)$ identifié, nous allons construire sa $\mathcal{M}$-ième PCPC (3.7) que nous noterons $\mathcal{Q}'^{[\mathcal{M}]}(D)$. Puis pour i variant de 1 à $\mathcal{M}$, nous comparerons la i-ème colonne de $\mathcal{Q}'^{[M]}(D)$ aux colonnes de $G_p(D)$. Nous avons montré lors de la construction du code poinçonné, que chaque ligne du motif de poinçonnage (P) correspondait aux colonnes de la $\mathcal{M}$-ième PCPC qui se trouvait dans le matrice $G_p(D)$. Par conséquent, si la i-ème colonne de $\mathcal{Q}'^{[\mathcal{M}]}(D)$ se trouve également dans $G_p(D)$, alors nous pourrons supprimer cette colonne de $G_p(D)$ et nous construirons le motif de poinçonnage en lui associant un bit à "1". En revanche, si cette colonne ne se trouve pas dans $G_p(D)$, nous associerons au motif de poinçonnage un bit à "0". Cet algorithme qui est présenté en algorithme 6 est itéré jusqu'à ce qu'il n'y ait plus de colonne dans la matrice $G_p(D)$.

Nous noterons $g'_j(D)$ les polynômes générateurs du code parent et P' le motif de poinçonnage identifié. En sortie de cet algorithme, nous obtiendrons les n polynômes générateurs du code parent et la matrice de poinçonnage P' qui sera de taille $n \times \mathcal{M}$ et composée de n_p éléments non-nuls.

Algorithme 6 : Algorithme d'identification du code parent et du motif de poinçonnage

Entrées : La matrice du code poinçonné ($G_p(D)$) et la profondeur du regroupement ($\mathcal{M}$)
Sorties : La matrice génératrice du code parent ($g'(D)$) et le motif de poinçonnage (P')

$j = 0$;
$P' = []$;
tant que $G_p(D)$ *n'est pas vide* **faire**
  $\hat{P} = []$;
  pour $l = 1$ *à* $\mathcal{M}$ **faire**
    tant que $l < \mathcal{M} + 1$ **faire**
      $q'(D) = \sum_{m=1}^{\mathcal{M}} D^{\beta(m,l)-\mathcal{M}}.g_{p_{(m,1)}}(D^{\mathcal{M}})$;
      si *Toute les puissances de* $q'(D)$ *sont positives* **alors**
        On construit la $\mathcal{M}$-ième PCPC de $q'(D)$ => $\mathcal{Q}'^{[\mathcal{M}]}(D)$;
        pour $i = 1$ *à* $\mathcal{M}$ **faire**
          si *La i-ème colonne de* $\mathcal{Q}'^{[\mathcal{M}]}(D)$ *est dans* $G_p(D)$ **alors**
            $\hat{P} = [\hat{P} \quad 1]$;
            On supprime la colonne de $G_p(D)$;
          sinon
            $\hat{P} = [\hat{P} \quad 0]$;
          fin
        fin
        $l = \mathcal{M} + 1$;
      fin
    fin
  fin
  $j = j + 1$;
  $g'_j(D) = q'(D)$;
  $P' = [P'; \hat{P}]$;
fin
n=j;

Exemple 3.42.

Suite de l'exemple 3.38.
La matrice génératrice du code poinçonné est telle que :

$$G_p(D) = \begin{bmatrix} 1+D & 1+D & 1 \\ 0 & D & 1+D \end{bmatrix}$$

D'après l'équation (3.18), le polynôme $q'(D)$ associé à la première colonne de $G_p(D)$ est défini par :

$$q'(D) = \sum_{m=1}^{2} D^{\beta(m,l)-2} \cdot g_{p_{(m,1)}}(D^2)$$

Calculons tout d'abord ce polynôme pour $l = 1$:

$$q'(D) = D^{\beta(1,1)-2} \cdot g_{p_{(1,1)}}(D^2) + D^{\beta(2,1)-2} \cdot g_{p_{(2,1)}}(D^2) = 1 + D^2$$

Ce polynôme contient uniquement des puissances positives, donc nous allons calculer sa 2-ième PCPC :

$$\mathcal{Q}'^{[2]}(D) = \begin{bmatrix} 1+D & 0 \\ 0 & 1+D \end{bmatrix}$$

Afin de construire le motif de poinçonnage, nous allons rechercher les colonnes de $\mathcal{Q}'^{[2]}(D)$ qui se trouvent également dans $G_p(D)$. Nous noterons respectivement $c_q(i)$ et $c_g(i)$ les i-ème colonnes de $\mathcal{Q}'^{[2]}(D)$ et $G_p(D)$. Nous obtenons :

$$c_q(1) = c_g(1); \quad c_q(2) = \varnothing$$

soit le motif P' associé au polynôme $q'(D)$:

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

et le polynôme $q'(D)$ qui correspond au polynôme $g'_1(D)$:

$$g'_1(D) = 1 + D^2$$

Afin d'identifier le second polynôme générateur du code parent, nous allons tout d'abord supprimer la colonne 1 de la matrice $G_p(D)$:

$$G_p(D) = \begin{bmatrix} 1+D & 1 \\ D & 1+D \end{bmatrix}$$

Pour $l = 1$, le polynôme associé à la première colonne de $G_p(D)$ est :

$$q'(D) = D^{\beta(1,1)-2} \cdot g_{p_{(1,1)}}(D^2) + D^{\beta(2,1)-2} \cdot g_{p_{(2,1)}}(D^2) = 1 + D + D^2$$

Ce polynôme contient uniquement des puissances positives donc nous allons calculer sa 2-ième PCPC :

$$\mathcal{Q}'^{[2]}(D) = \begin{bmatrix} 1+D & 1 \\ D & 1+D \end{bmatrix}$$

Nous allons rechercher les colonnes de $\mathcal{Q}'^{[2]}(D)$ qui se trouvent également dans $G_p(D)$ pour construire la motif de poinçonnage associé à ce polynôme :

$$c_q(1) = c_g(1); \quad c_q(2) = c_g(2)$$

soit le motif P' associé au polynôme $q'(D)$:

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

et le polynôme $q'(D)$ qui correspond au polynôme générateur $g'_2(D)$:

$$g'_2(D) = 1 + D + D^2$$

La matrice $G_p(D)$ étant vide (après la suppression des colonnes 1 et 2), nous obtenons au final les paramètres suivants :

$$g'_1(D) = 1 + D^2 \quad g'_2(D) = 1 + D + D^2$$

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

Nous pouvons vérifier que ces paramètres correspondent bien aux paramètres du code parent et au motif de poinçonnage utilisés.

Exemple 3.43.

Prenons l'exemple de reconnaissance aveugle d'un $C(3, 1, 6)$ code parent.

La matrice génératrice du code est :

$$G(D) = \begin{bmatrix} 1+D+D^2+D^3+D^5 & 1+D^2+D^4+D^5 & 1+D^3+D^4+D^5 \end{bmatrix}$$

Nous allons réaliser un poinçonnage de profondeur 4 ($\mathcal{M} = 4$) avec le motif P suivant :

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

Nous allons donc obtenir un code poinçonné de rendement $\frac{4}{7}$.
Voici les 4-ièmes *décompositions polyphases* des 3 polynômes générateurs, ainsi que leurs PCPC :

$g_{1,i}(D)$	4-ième *décomposition polyphase*	PCPC
$g_{1,1}(D) = 1+D+D^2+D^3+D^5$	$\begin{bmatrix} 1 & 1+D & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1+D & 1 & 1 \\ D & 1 & 1+D & 1 \\ D & D & 1 & 1+D \\ D+D^2 & D & D & 1 \end{bmatrix}$
$g_{1,2}(D) = 1+D^2+D^4+D^5$	$\begin{bmatrix} 1+D & D & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1+D & D & 1 & 0 \\ 0 & 1+D & D & 1 \\ D & 0 & 1+D & D \\ D^2 & D & 0 & 1+D \end{bmatrix}$
$g_{1,3}(D) = 1+D^3+D^4+D^5$	$\begin{bmatrix} 1+D & D & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1+D & D & 0 & 1 \\ D & 1+D & D & 0 \\ 0 & D & 1+D & D \\ D^2 & 0 & D & 1+D \end{bmatrix}$

Afin d'obtenir la matrice du code regroupé, nous réaliserons un entrelacement de profondeur 4 sur les lignes :

$$(1,2,3,4) => (1,2,3,4)$$

et sur les colonnes :

$$(1,2,3,4,5,6,7,8,9,10,11,12) => (1,5,9,2,6,10,3,7,11,4,8,12)$$

Nous obtenons la matrice du code regroupé à la profondeur 4 :

$$G^{[4]}(D) = \begin{bmatrix} 1 & 1+D & 1+D & 1+D & D & D & 1 & 1 & 0 & 1 & 0 & 1 \\ D & 0 & D & 1 & 1+D & 1+D & 1+D & D & D & 1 & 1 & 0 \\ D & D & 0 & D & 0 & D & 1 & 1+D & 1+D & 1+D & D & D \\ D+D^2 & D^2 & D^2 & D & D & 0 & D & 0 & D & 1 & 1+D & 1+D \end{bmatrix}$$

D'après la bijection ϕ et la matrice de poinçonnage P, nous devons supprimer les colonnes $(3,5,8,9,12)$. Après la suppression de ces colonnes, nous obtenons la matrice du code poinçonné :

$$G_p(D) = \begin{bmatrix} 1 & 1+D & 1+D & D & 1 & 1 & 0 \\ D & 0 & 1 & 1+D & 1+D & 1 & 1 \\ D & D & D & D & 1 & 1+D & D \\ D+D^2 & D^2 & D & 0 & D & 1 & 1+D \end{bmatrix}$$

Nous allons identifier le code parent et le motif de poinçonnage à partir de la seule connaissance du $C_p(7,4,3)$ code poinçonné. Afin d'identifier le code parent et le motif de poinçonnage, nous appliquerons l'algorithme 6. Nous définirons la matrice β telle que :

$$\beta = \begin{pmatrix} 4 & 5 & 6 & 7 \\ 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

D'après (3.18), le polynôme $q'(D)$ associé à la première colonne de $G_p(D)$, pour $l = 1$, est

$$\begin{aligned} q'(D) &= D^0.(1) + D^{-1}.(D^4) + D^{-2}.(D^4) + D^{-3}.(D^4 + D^8) \\ &= 1 + D + D^2 + D^3 + D^5 \end{aligned}$$

On construit la 4-ième PCPC de ce polynôme :

$$\mathcal{Q}'^{[4](D)} = \begin{bmatrix} 1 & 1+D & 1 & 1 \\ D & 1 & 1+D & 1 \\ D & D & 1 & 1+D \\ D+D^2 & D & D & 1 \end{bmatrix}$$

et on compare les colonnes de cette matrice à celles de $G_p(D)$:

$$c_q(1) = c_g(1); \quad c_q(2) = c_g(3); \quad c_q(3) = c_g(5); \quad c_q(4) = c_g(6);$$

Par conséquent, le motif de poinçonnage associé à ce polynôme est :

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

et le polynôme $q'(D)$ correspond au premier polynôme générateur du code parent :

$$g'_1(D) = 1 + D + D^2 + D^3 + D^5$$

Après la suppression des colonnes de $G_p(D)$ qui sont associées à la 4-ième PCPC de ce polynôme, nous obtenons :

$$G'_p(D) = \begin{bmatrix} 1+D & D & 0 \\ 0 & 1+D & 1 \\ D & D & D \\ D^2 & 0 & 1+D \end{bmatrix}$$

Prenons la première colonne de $G_p(D)$ et calculons le polynôme qui lui est associé pour $l = 1$:

$$\begin{aligned} q'(D) &= D^0.(1 + D^4) + D^{-1}.(0) + D^{-2}.(D^4) + D^{-3}.(D^8) \\ &= 1 + D^2 + D^4 + D^5 \end{aligned}$$

On construit la 4-ième PCPC de ce polynôme :

$$\mathcal{Q}'^{[4](D)} = \begin{bmatrix} 1+D & D & 1 & 0 \\ 0 & 1+D & D & 1 \\ D & 0 & 1+D & D \\ D^2 & D & 0 & 1+D \end{bmatrix}$$

et nous comparons les colonnes de cette matrice à celles de $G_p(D)$:

$$c_q(1) = c_g(1); \quad c_q(2) = \varnothing; \quad c_q(3) = \varnothing; \quad c_q(4) = c_g(3);$$

Alors, le motif de poinçonnage associé à ce polynôme est :

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

et le deuxième polynôme générateur du code parent est tel que :

$$g'_2(D) = 1 + D^2 + D^4 + D^5$$

Après la suppression des colonnes 1 et 4, la matrice $G_p(D)$ est telle que :

$$G_p(D) = \begin{bmatrix} D \\ 1+D \\ D \\ 0 \end{bmatrix}$$

Calculons le polynôme associé à cette dernière colonne de $G_p(D)$ pour $l = 1$:

$$\begin{aligned} q'(D) &= D^0.(D^4) + D^{-1}.(1+D^4) + D^{-2}.(D^4) + D^{-3}.(0) \\ &= D^{-1} + D^2 + D^3 + D^4 \end{aligned}$$

Ce polynôme étant composé de puissances négatives, nous allons incrémenter la valeur de l afin de calculer le polynôme associé à cette colonne :

$$\begin{aligned} q'(D) &= D^1.(D^4) + D^0.(1+D^4) + D^{-1}.(D^4) + D^{-2}.(0) \\ &= 1 + D^3 + D^4 + D^5 \end{aligned}$$

On construit la 4-ième PCPC de ce polynôme :

$$\mathcal{Q}'^{[4](D)} = \begin{bmatrix} 1+D & D & 0 & 1 \\ D & 1+D & D & 0 \\ 0 & D & 1+D & D \\ D^2 & 0 & D & 1+D \end{bmatrix}$$

et on compare les colonnes de cette matrice à celles de $G_p(D)$:

$$c_q(1) = \varnothing; \quad c_q(2) = c_g(1); \quad c_q(3) = \varnothing; \quad c_q(4) = \varnothing;$$

Alors, le motif de poinçonnage associé à ce polynôme est :

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

et le troisième polynôme générateur du code parent est tel que :

$$g'_3(D) = 1 + D^3 + D^4 + D^5$$

La matrice $G_p(D)$ étant nulle après la suppression de la colonne $c_g(1)$, l'ensemble des paramètres du code parent et du motif de poinçonnage ont été identifiés.

Récapitulatif des résultats obtenus :

$$G'(D) = \begin{bmatrix} 1+D+D^2+D^3+D^5 & 1+D^2+D^4+D^5 & 1+D^3+D^4+D^5 \end{bmatrix}$$

et

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

Nous pouvons vérifier que ces résultats correspondent à la matrice génératrice du code parent ainsi qu'à la matrice de poinçonnage.

3.3.3 Reconnaissance aveugle pour un code parent de rendement k/n

Dans le cas d'un code de rendement k/n, avec $k > 1$, l'algorithme d'identification du code parent et du motif de poinçonnage à partir de la seule connaissance du code poinçonné est plus

complexe que dans le cas d'un code parent de rendement $1/n$. En effet, il nous faudra tout d'abord identifier la profondeur du regroupement $\mathcal{M}$ et le nombre d'entrées du code parent k. Nous savons que :

$$k_p = \mathcal{M}.k \tag{3.19}$$

En conséquence, connaissant la valeur de k_p, nous devrons tester l'ensemble des couples $(k, \mathcal{M})$ possibles. Comme nous savons que les valeurs de k et $\mathcal{M}$ sont des entiers, nous verrons qu'il existe très peu de couples possibles. Puis, pour chaque couple, nous appliquerons l'algorithme que nous allons présenter afin d'identifier un code parent optimal et un motif de poinçonnage.

Afin d'expliquer cette méthode d'identification, nous ferons tout d'abord un bref rappel sur le principe de construction du code poinçonné. Notons $G'(D)$ la matrice composée des $\mathcal{M}$-ième PCPC des $(k \times n)$ polynômes générateurs du code parent :

$$G'(D) = \begin{bmatrix} \mathcal{Q}_{1,1}^{[\mathcal{M}]}(D) & \mathcal{Q}_{1,2}^{[\mathcal{M}]}(D) & \cdots & \mathcal{Q}_{1,n}^{[\mathcal{M}]}(D) \\ \vdots & \vdots & \cdots & \vdots \\ \mathcal{Q}_{k,1}^{[\mathcal{M}]}(D) & \mathcal{Q}_{k,2}^{[\mathcal{M}]}(D) & \cdots & \mathcal{Q}_{k,n}^{[\mathcal{M}]}(D) \end{bmatrix} \tag{3.20}$$

où les matrices $\mathcal{Q}_{i,j}^{[\mathcal{M}]}(D)$ sont des matrices de taille $\mathcal{M} \times \mathcal{M}$. La matrice du code regroupé sur la profondeur de $\mathcal{M}$, notée $G^{[\mathcal{M}]}(D)$, est obtenue en entrelaçant les lignes et les colonnes de la matrice $G'(D)$ de profondeur $\mathcal{M}$. Enfin, on obtient la matrice du code poinçonné en supprimant certaines colonnes (en suivant le motif de poinçonnage P) de la matrice du code regroupé. Cette matrice, notée $G_p(D)$, est telle que :

$$G_p(D) = \begin{bmatrix} g_{p_{(1,1)}}(D) & \cdots & g_{p_{(1,n_p)}}(D) \\ \vdots & \cdots & \vdots \\ g_{p_{(k_p,1)}}(D) & \cdots & g_{p_{(k_p,n_p)}}(D) \end{bmatrix} \tag{3.21}$$

Dans le cas d'un code de rendement $1/n$, chaque colonne de la matrice $G_p(D)$ était associée à un unique polynôme générateur. Or, dans le cas d'un code de rendement k/n, chaque colonne de cette matrice est associée à k polynômes. De par l'entrelacement sur les lignes qui a été réalisé pour obtenir la matrice du code regroupé, nous devrons dans un premier temps désentrelacer les lignes de la matrice $G_p(D)$. Nous noterons, $G'_p(D)$, la matrice qui correspond à la matrice $G_p(D)$ où un entrelacement de profondeur $\mathcal{M}$ a été réalisé sur ses lignes :

$$G'_p(D) = \begin{bmatrix} g'_{p_{(1,1)}}(D) & \cdots & g'_{p_{(1,n_p)}}(D) \\ \vdots & \cdots & \vdots \\ g'_{p_{(k_p,1)}}(D) & \cdots & g'_{p_{(k_p,n_p)}}(D) \end{bmatrix} \tag{3.22}$$

Nous définirons k sous-matrices $G_p^j(D)$ de taille $\mathcal{M} \times n_p$ $(\forall j = 1, \cdots, k)$ telles que :

$$G_p^j(D) = \begin{bmatrix} g'_{p_{((j-1).\mathcal{M}+1,1)}}(D) & \cdots & g'_{p_{((j-1).\mathcal{M}+1,n_p)}}(D) \\ \vdots & \cdots & \vdots \\ g'_{p_{(j.\mathcal{M},1)}}(D) & \cdots & g'_{p_{(j.\mathcal{M},n_p)}}(D) \end{bmatrix} \tag{3.23}$$

Alors, en appliquant directement l'algorithme 6 sur les k sous-matrices $G_p^j(D)$, nous obtiendrons pour ces k sous-matrices n polynômes générateurs et une matrice de poinçonnage de taille $\mathcal{M} \times n$. En regroupant les n polynômes obtenus pour les k sous-matrices, nous obtiendrons au final $k \times n$ polynômes générateurs. En sortie de cet algorithme, nous devrons tout d'abord vérifier que les motifs de poinçonnage identifiés avec les k sous-matrices $G_p^j(D)$ sont identiques. Puis, il faudra

également vérifier que la matrice génératrice du code parent est une matrice génératrice d'un code optimal.

Exemple 3.44.

Suite de l'exemple 3.39.

Les paramètres du code poinçonné ainsi que sa matrice génératrice sont tels que :

$$\begin{cases} n_p = 5 \\ k_p = 4 \\ K_p = 2 \end{cases}$$

$$G_p(D) = \begin{bmatrix} 1+D & 1 & D & 0 & 0 \\ 0 & 1+D & 1+D & 0 & 1 \\ D & 0 & 0 & 1 & D \\ D & 0 & D & 1+D & 1+D \end{bmatrix}$$

La méthode de reconnaissance aveugle expliquée dans le chapitre 2 permet d'obtenir ces paramètres.

Connaissant le $C_p(5,4,2)$ code poinçonné, nous devons identifier le code parent et le motif de poinçonnage. D'après les paramètres du code poinçonné, nous savons que :

$$k_p = \mathcal{M}.k = 4$$

Par conséquent, il existe uniquement deux couples $(k, \mathcal{M})$ possibles :

$$\begin{cases} k = 1 \text{ et } \mathcal{M} = 4 \\ k = 2 \text{ et } \mathcal{M} = 2 \end{cases}$$

Nous devrons tester ces deux couples afin d'identifier un code parent optimal (au sens du chapitre 1) et un motif de poinçonnage P de taille $n \times \mathcal{M}$ qui sera composé de n_p bits à "1".

– Test pour $k = 1$ et $\mathcal{M} = 4$:

En appliquant l'algorithme 6 sur la matrice génératrice du code poinçonné, nous obtenons un $C(3,1,6)$ code de matrice génératrice $G'(D)$:

$$G'(D) = \begin{bmatrix} 1+D+D^2+D^4 & 1+D+D^4 & 1+D^2+D^4+D^5 \end{bmatrix}$$

et le motif de poinçonnage suivant :

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

Or, d'après les propriétés énoncées dans le chapitre 1, la matrice génératrice $G'(D)$ ne correspond pas à la matrice génératrice d'un code optimal. Par conséquent, nous savons que le code identifié n'est pas un code optimal donc nous allons tester le second couple $(k, \mathcal{M})$.

– Test pour $k = 2$ et $\mathcal{M} = 2$:

D'après ces paramètres, la première étape est de réaliser un entrelacement de profondeur $\mathcal{M} = 2$

sur les lignes de la matrice $G_p(D)$. Nous obtenons la matrice $G'p(D)$:

$$G'_p(D) = \begin{bmatrix} 1+D & 1 & D & 0 & 0 \\ D & 0 & 0 & 1 & D \\ 0 & 1+D & 1+D & 0 & 1 \\ D & 0 & D & 1+D & 1+D \end{bmatrix}$$

Nous noterons $G^1_p(D)$ la matrice composée des 2 premières lignes de $G'_p(D)$ et $G^2_p(D)$ celle composée des 2 dernières lignes :

$$G^1_p(D) = \begin{bmatrix} 1+D & 1 & D & 0 & 0 \\ D & 0 & 0 & 1 & D \end{bmatrix} \quad G^2_p(D) = \begin{bmatrix} 0 & 1+D & 1+D & 0 & 1 \\ D & 0 & D & 1+D & 1+D \end{bmatrix}$$

Pour chacune de ces matrices, nous allons appliquer l'algorithme 6. Nous allons tout d'abord construire la matrice β :

$$\beta = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}$$

Avec la matrice $G^1_p(D)$, nous obtenons 3 polynômes générateurs que nous noterons $g'_{1,j}(D)$ ($\forall j = 1, \cdots, 3$) :

$$\begin{cases} g'_{1,1}(D) = 1 + D + D^2 \\ g'_{1,2}(D) = 1 \\ g'_{1,3}(D) = D^2 \end{cases}$$

et le motif de poinçonnage P' suivant :

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

Puis avec la matrice $G^2_p(D)$, nous obtenons 3 polynômes générateurs que nous noterons $g'_{2,j}(D)$ ($\forall j = 1, \cdots, 3$) :

$$\begin{cases} g'_{2,1}(D) = D \\ g'_{2,2}(D) = 1 + D^2 \\ g'_{2,3}(D) = 1 + D + D^2 \end{cases}$$

et le motif de poinçonnage P' :

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

Nous vérifions tout d'abord que les motifs de poinçonnage identifiés sont identiques. Puis on construit la matrice génératrice du code parent avec les polynômes générateurs $g'_{i,j}(D)$ identifiés :

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

D'après les propriétés énoncées dans le chapitre 1, nous pouvons vérifier que cette matrice génératrice est une matrice génératrice d'un code optimal.

Dans cette section, nous avons présenté une méthode qui permet à partir de la seule connaissance du code poinçonné de retrouver le code parent et le motif de poinçonnage. Lors de l'identification du code parent et du poinçonnage, les seules données utiles correspondent aux paramètres du code poinçonné, par conséquent l'identification aveugle du code parent et du poinçonnage reste identique lors de l'interception d'une communication bruitée et non-bruitée. En effet, les mots bruités sont

uniquement utilisés pour identifier le code poinçonné et ce code est identifié avec les méthodes de reconnaissances aveugles présentées dans le chapitre 2.

3.4 Analyse de l'algorithme

Nous proposerons dans cette partie une analyse des performances de notre algorithme de reconnaissance aveugle des codes poinçonnés. Pour cette analyse, nous reprendrons les mêmes hypothèses et notations qui ont été introduites lors de l'analyse réalisée dans le chapitre 2 en section 2.4.

- Les codes convolutifs identifiés sont des codes optimaux
- Le train binaire reçu est synchronisée
- Le canal de transmission est un BSC
- Le train binaire reçu est composé d'au minimum $l_{max}.M$ bits

Nous fixerons les paramètres $l_{max} = 100$ et $M = 200$. Il sera donc nécessaire de disposer d'un train binaire de 20000 bits. Pour chaque simulation, 1000 tirages de Monte-Carlo ont été réalisés, où pour chaque essai la séquence des mots d'information et les erreurs ont été tirés aléatoirement. Nous étudierons dans un premier temps l'influence des itérations sur les performances de notre algorithme.

Nous proposons de réaliser ces différentes études sur trois codes convolutifs poinçonnés :

- Le $C(2,1,7)$ code parent poinçonné à la profondeur de $\mathcal{M} = 2$:

$$G = \begin{pmatrix} 133 & 171 \end{pmatrix} \quad \text{et} \quad P = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{3.24}$$

  Soit le $C_p(3,2,4)$ code poinçonné de matrice génératrice :

$$G_p = \begin{pmatrix} 15 & 15 & 6 \\ 3 & 6 & 15 \end{pmatrix} \tag{3.25}$$

- Le $C(3,1,7)$ code parent poinçonné à la profondeur de $\mathcal{M} = 3$:

$$G = \begin{pmatrix} 171 & 165 & 133 \end{pmatrix} \quad \text{et} \quad P = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.26}$$

  Soit le $C_p(4,3,3)$ code poinçonné de matrice génératrice :

$$G_p = \begin{pmatrix} 7 & 6 & 0 & 4 \\ 2 & 5 & 7 & 4 \\ 2 & 2 & 3 & 7 \end{pmatrix} \tag{3.27}$$

- Le $C(3,2,3)$ code parent poinçonné à la profondeur de $\mathcal{M} = 2$:

$$G = \begin{pmatrix} 7 & 4 & 1 \\ 2 & 5 & 7 \end{pmatrix} \quad \text{et} \quad P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{3.28}$$

Soit le $C_p(5, 4, 2)$ code poinçonné de matrice génératrice :

$$G_p = \begin{pmatrix} 3 & 2 & 1 & 0 & 0 \\ 0 & 3 & 3 & 0 & 2 \\ 1 & 0 & 0 & 2 & 1 \\ 1 & 0 & 1 & 3 & 3 \end{pmatrix} \tag{3.29}$$

3.4.1 Le gain apporté par notre algorithme itératif

Afin de limiter le temps de calcul permettant d'atteindre une bonne probabilité de détection, nous fixerons le nombre d'itérations maximum à 50. Comme au chapitre 2, dans le but d'évaluer ce nombre d'itérations, nous noterons $\lambda_{x\to y}$, le gain obtenu entre l'itération x et y, tel que :

$$\lambda_{x\to y} = \frac{\mathcal{P}_{det}(y) - \mathcal{P}_{det}(x)}{\mathcal{P}_{det}(x)} \tag{3.30}$$

où $\mathcal{P}_{det}(i)$ est la probabilité de détecter le bon code à la i-ème itération. Nous exprimerons ce gain en pourcentage. Dans ce chapitre, la probabilité de détecter le bon code correspond à la probabilité d'identifier le bon code poinçonné, le bon code parent et le bon motif de poinçonnage.

– Le $C(2, 1, 7)$ code parent poinçonné à la profondeur de $\mathcal{M} = 2$

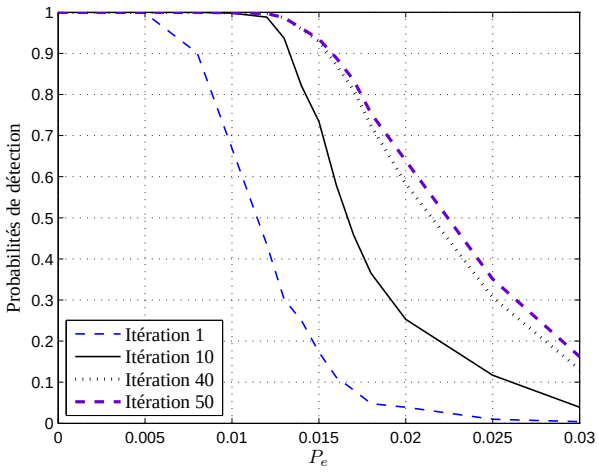


Figure 3.3 — $C(2, 1, 7)$ et $\mathcal{M} = 2$: Probabilité de détection en fonction de P_e

Nous avons représenté sur la figure 3.3 la probabilité de détection, $\mathcal{P}_{det}$, obtenue en fonction de la probabilité d'erreur du canal, P_e, pour les itérations 1, 10, 40 et 50. Nous pouvons remarquer que pour ce code, les performances optimales de notre algorithme sont atteintes à la 40-ième itération. En effet, le gain entre l'itération 40 et 50 est proche de 0. En revanche, le gain (représenté dans le tableau 3.1) entre l'itération 1 et 40 est très important. Effectivement, pour $P_e = 0.015$, $\lambda_{1\to 40}$ est proche de 436%. Pour cette P_e, après 1 itération $\mathcal{P}_{det}$ est proche de 0.17 et nous passons à 0.93 après 40 itérations.

Tableau 3.1 — Gain de détection pour le $C(2, 1, 7)$ code poinçonné à la profondeur de $\mathcal{M} = 2$

P_e	0.01	0.015	0.02	0.03
$C(2, 1, 7)$ et $\mathcal{M} = 2$: $\lambda_{1\to 10}$ (%)	49%	324%	548%	875%
$C(2, 1, 7)$ et $\mathcal{M} = 2$: $\lambda_{1\to 40}$ (%)	49%	436%	1394%	3175%

– Le $C(3,1,7)$ code parent poinçonné à la profondeur de $\mathcal{M}=3$

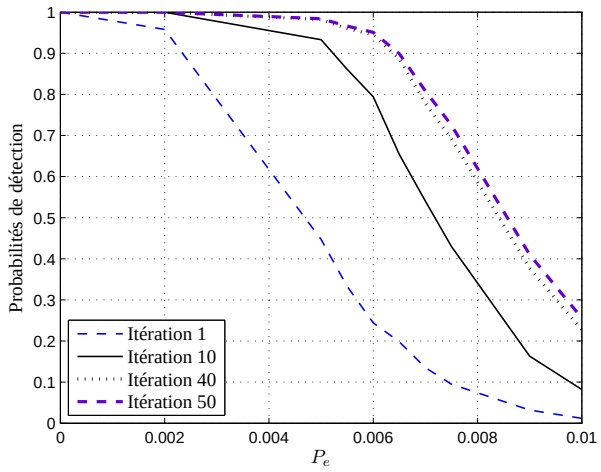


Figure 3.4 — $C(3,1,7)$ et $\mathcal{M}=3$: Probabilité de détection en fonction de P_e

Sur la figure 3.4, la probabilité de détection en fonction de la probabilité d'erreur du canal est représentée pour 1, 10, 40 et 50 itérations. Comme pour le code précédent, il est nécessaire de réaliser 40 itérations afin d'atteindre les performances optimales (le gain entre l'itération 40 et 50 est proche de 0). Nous remarquons que pour $P_e > 0.002$, le gain apporté par notre algorithme itératif est très important. En effet, pour $P_e = 0.006$, $\lambda_{1\to 40}$ est proche de 287%. Un récapitulatif des différents gains est représenté dans le tableau 3.2

Tableau 3.2 — Gain de détection pour le $C(3,1,7)$ code poinçonné à la profondeur de $\mathcal{M}=3$

P_e	0.002	0.005	0.006	0.01
$C(3,1,7)$ et $\mathcal{M}=3$: $\lambda_{1\to 10}$ (%)	4%	108%	225%	583%
$C(3,1,7)$ et $\mathcal{M}=3$: $\lambda_{1\to 40}$ (%)	4%	119%	287%	1791%

– Le $C(3,2,3)$ code parent poinçonné à la profondeur de $\mathcal{M}=2$

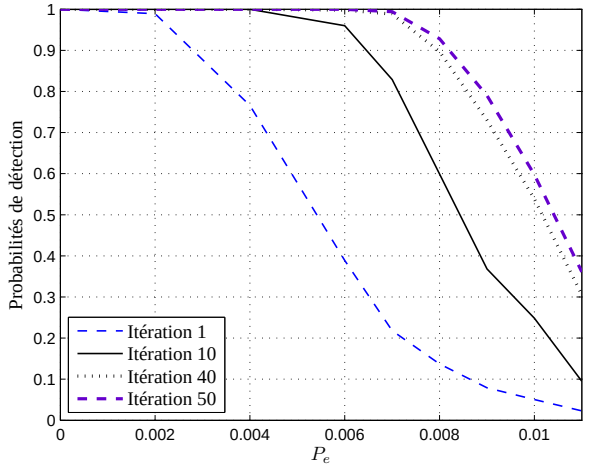


Figure 3.5 — $C(3,2,3)$ et $\mathcal{M}=2$: Probabilité de détection en fonction de P_e

Pour 1, 10, 40 et 50 itérations, nous avons représenté la probabilité de détection en fonction de la probabilité d'erreur du canal sur la figure 3.5. Nous constatons exactement le même comportement que pour les deux codes précédents. En effet, les performances optimales sont atteintes dès la 40-ième itération et le gain (représenté dans le tableau 3.3) obtenu entre l'itération 1 et 40 est très important.

Tableau 3.3 — Gain de détection pour le $C(3,2,3)$ code poinçonné à la profondeur de $\mathcal{M}=2$

P_e	0.004	0.006	0.008	0.011
$C(3,2,3)$ et $\mathcal{M}=2$: $\lambda_{1\to10}$ (%)	30%	146%	337%	313%
$C(3,2,3)$ et $\mathcal{M}=2$: $\lambda_{1\to40}$ (%)	30%	156%	554%	1226%

Nous venons de montrer que comme dans le cas d'un code convolutif non poinçonné, les performances globales de notre algorithme sont largement améliorées par le processus itératif. Nous pouvons également noter qu'il existe une différence entre les performances de détection que nous obtenons pour ces trois codes poinçonnés. En effet, le $C(2,1,7)$ code poinçonné avec $\mathcal{M}=2$ est plus facile à identifier que les deux autres. Mais comme nous l'avons vu dans le chapitre 2, ceci est simplement dû au fait que les deux autres codes poinçonnés introduisent moins de redondance. Effectivement le $C(2,1,7)$ code poinçonné avec $\mathcal{M}=2$ est équivalent à un $C_p(3,2,4)$ code, alors que les deux autres sont, respectivement, équivalent à un $C_p(4,3,3)$ code et à un $C_p(5,4,2)$ code.

Si nous comparons ces résultats à ceux obtenus lors de la reconnaissance aveugle d'un code non-poinçonné (voir chapitre 2), nous remarquons qu'il est nécessaire de réaliser plus d'itérations pour identifier un code poinçonné. Lors de la reconnaissance aveugle d'un code poinçonné, l'algorithme peut-être découpé en deux étapes. La première étape consiste à identifier le code poinçonné à partir des mots poinçonnés et bruités, avec la méthode du chapitre 2. Puis, la seconde étape a pour objectif d'identifier un code parent et un motif de poinçonnage à partir du code poinçonné. Par conséquent, si le premier algorithme identifie un code, mais que celui-ci ne correspond pas au code poinçonné, il sera peu probable que le second algorithme identifie un code parent optimal et un motif de poinçonnage associé au code identifié. De ce fait, il sera effectivement nécessaire de réaliser plus d'itérations que dans le cas de l'identification d'un simple code afin d'atteindre les performances optimales de notre algorithme.

3.4.2 Les probabilités de détection

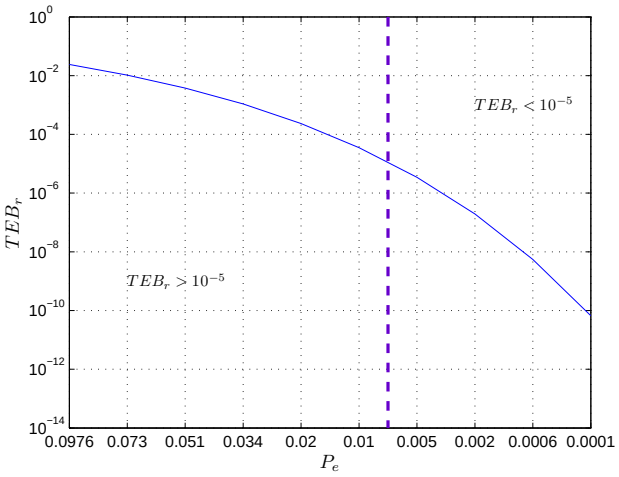
Comme dans le chapitre précédent, afin d'analyser les performances de détection de notre algorithme, nous utiliserons une fois de plus les trois probabilités suivantes :

1. Probabilité de détection $\mathcal{P}_{det}$: la probabilité d'identifier le bon code poinçonné, le bon code parent et le bon motif de poinçonnage
2. Probabilité de fausse alarme $\mathcal{P}_{fa}$: la probabilité d'identifier un code parent optimal mais pas le bon
3. Probabilité de non détection $\mathcal{P}_{nd}$: la probabilité de n'identifier aucun code

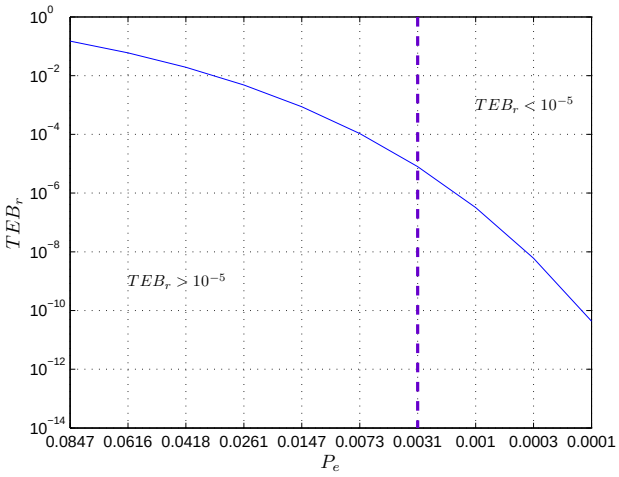
Rappelons que l'identification aveugle à partir des mots bruités permet d'identifier le code équivalent poinçonné. En effet, l'identification du code parent et du motif de poinçonnage est réalisée à partir de la connaissance du code poinçonné. Par conséquent, afin d'évaluer la pertinence de nos résultats, nous les comparerons au pouvoir de correction du code poinçonné et non du code parent.

Sur les figures 3.6, les taux d'erreur binaire résiduels théoriques, noté TEB_r, en fonction de la probabilité d'erreur du canal P_e sont représentés pour les trois codes poinçonnés suivant : $C_p(3,2,4)$, $C_p(4,3,3)$ et $C_p(5,4,2)$. Nous avons défini dans le chapitre 2 l'expression théorique de ce TEB_r dans l'équation 2.125. Nous avons également expliqué dans ce chapitre 2 que nous considérerons

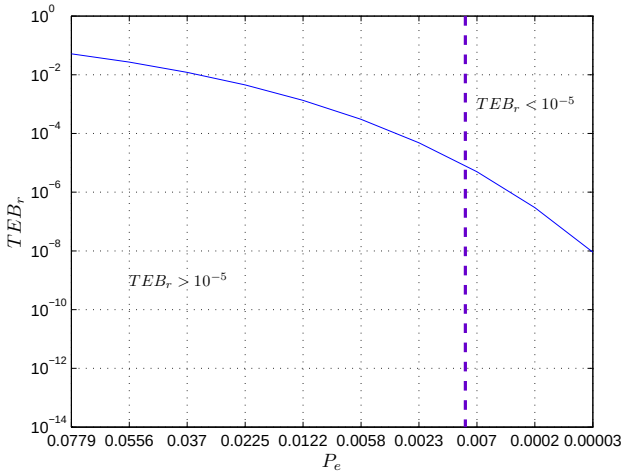
que le TEB_r est acceptable s'il est inférieur à 10^{-5}. Par conséquent, nous avons délimité sur ces figures les zones correspondant à un TEB_r inférieure à 10^{-5}, et celles où il est supérieur à 10^{-5}.



(a) TEB_r théorique du $C_p(3,2,4)$ code



(b) TEB_r théorique du $C_p(4,3,3)$ code



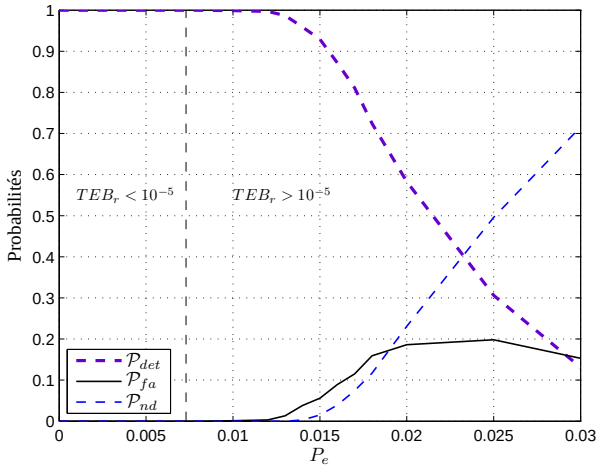
(c) TEB_r théorique du $C_p(5,4,2)$ code

Figure 3.6 — Les TEB_r théoriques des codes poinçonnés

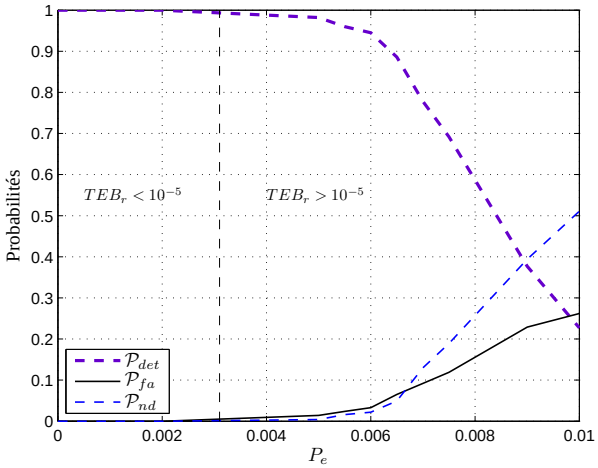
Sur les figures 3.7 sont représentées les trois probabilités ($\mathcal{P}_{det}$, $\mathcal{P}_{fa}$ et $\mathcal{P}_{nd}$) en fonction de la probabilité d'erreur du canal pour les trois codes précédents. Nous avons également délimité, sur chaque figure, les zones où le TEB_r est supérieur à 10^{-5} et celles où il est inférieur. Nous remarquons que pour ces trois codes poinçonnés, notre algorithme offre d'excellentes performances. En effet, pour les zones correspondant à un TEB_r inférieur à 10^{-5}, la probabilité de détection est proche de 1.

Pour ces trois codes, nous remarquons qu'à partir d'un certain seuil, les probabilités de non détection deviennent supérieures aux probabilités de détection. En comparant le point de croisement de ces deux courbes aux TEB_r théoriques, nous notons que ces croisements se réalisent lorsque le TEB_r est strictement supérieur à 10^{-4}. Par conséquent, de tels codes ne seraient pas utilisés dans ce contexte, puisque même en contexte coopératif le décodeur serait incapable de décoder la trame reçue avec un TEB résiduel après décodage acceptable.

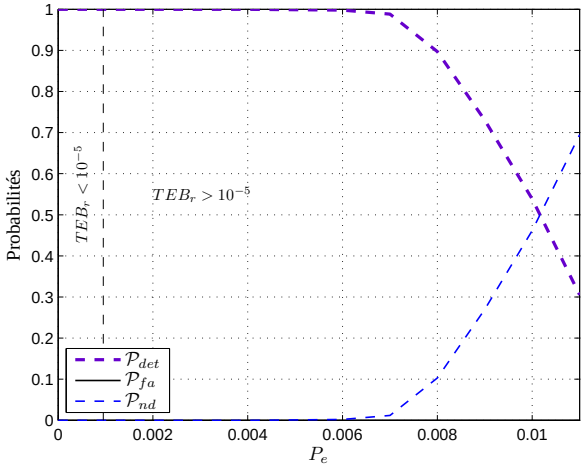
Dans le cas de la reconnaissance d'un code non-poinçonné (voir chapitre 2), nous avions remarqué que les probabilités de fausse alarme dépassaient, à compter d'un certain seuil, les probabilités



(a) Probabilités de détection du $C(2,1,7)$ code et $\mathcal{M}=2$



(b) Probabilités de détection du $C(3,1,7)$ code et $\mathcal{M}=3$



(c) Probabilités de détection du $C(3,2,3)$ code et $\mathcal{M}=2$

Figure 3.7 — Les probabilités de détection

de détection. Or dans le cas d'un code poinçonné, se sont les probabilités de non détection. Dans le cas de l'identification aveugle d'un code poinçonné, nous utilisons l'algorithme du chapitre 2 pour identifier le code convolutif poinçonné. Puis, à partir de ce code poinçonné nous appliquons l'algorithme décrit dans ce chapitre afin d'identifier un code parent et un motif de poinçonnage qui est équivalent au code poinçonné identifié. De ce fait, il est peu probable à partir d'un mauvais code poinçonné identifié avec la méthode du chapitre 2 d'identifier par la suite un code parent et un motif de poinçonnage. Par conséquent, la probabilité de fausse alarme sera faible comparée à la probabilité de non détection.

3.5 Conclusions

Dans ce chapitre, nous avons tout d'abord présenté une technique simple permettant d'augmenter le rendement du code. Cette technique, nommée poinçonnage, consiste simplement à ne pas transmettre tous les bits des mots de code. Nous avons montré qu'un code convolutif dont les mots de code avaient été poinçonnés pouvait être décrit par un autre code convolutif construit à partir du code parent et d'un motif de poinçonnage. Nous avons ensuite vu que l'identification aveugle

du code poinçonné était identique à celle de l'identification d'un code convolutif non-poinçonné. Puis, dans la deuxième partie de ce chapitre, nous avons développé une méthode permettant, à partir de la connaissance du code poinçonné, d'identifier le code parent et le motif de poinçonnage. Et enfin, nous avons analysé les performances de détection de notre algorithme d'identification aveugle d'un code convolutif poinçonné. Nous avons montré que pour un TEB résiduel théorique après décodage inférieur à 10^{-5}, notre algorithme nous permettait d'identifier le code poinçonné, le code parent et le motif de poinçonnage avec une probabilité de détection proche de 1.

Récapitulatif des paramètres identifiés par notre algorithme :

- Les paramètres du code poinçonné : n_p, k_p et K_p
- La matrice génératrice du code poinçonné $G_p(D)$
- Les paramètres du code parent : n, k et K
- La matrice génératrice du code parent $G(D)$
- Le motif de poinçonnage P

Récapitulatif des points forts de notre algorithme :

- Une quantité relativement faible de données reçues est nécessaire (20000 bits)
- Une procédure itérative qui permet d'améliorer de manière significative les probabilités de détecter le bon code
- L'identification du code poinçonné, du code parent et du motif de poinçonnage
- Pour un TEB_r proche de 10^{-5} nous obtenons des probabilités de détection proche de 1

Dans le prochain chapitre, nous nous intéresserons à des schémas de codage qui permettront d'améliorer, de manière significative, la robustesse d'une transmission. Nous proposerons tout d'abord une étude théorique sur ces schémas de codage puis nous développerons des algorithmes de reconnaissance aveugle de ces schémas de codage.

CHAPITRE 4

Reconnaissance des codes concaténés et des turbocodes

4.1 Introduction

Comme nous l'avons expliqué dans le chapitre 3, les applications actuelles nécessitent des transmissions toujours plus fiables et plus rapides. Dans le chapitre précédent, nous avons présenté une technique simple qui permet d'augmenter le rendement d'un code, le poinçonnage des mots de code. Si cette technique permet d'augmenter le débit utile de la transmission, elle ne permet pas d'améliorer sa robustesse. De ce fait, nous présenterons dans ce chapitre des schémas de codage qui permettent d'améliorer de manière significative la capacité de correction d'un code.

La capacité de correction d'un code convolutif est évaluée par sa distance libre : plus cette distance sera grande et meilleure sera sa capacité de correction. Une liste non-exhaustive de codes convolutifs optimaux est présentée dans l'annexe D. En regardant cette liste, on remarque que pour un rendement donné, la distance libre augmente en même temps que la longueur de contrainte des codes. On en conclut aisément que les meilleurs codes, en terme de capacité de correction, sont les codes ayant une longueur de contrainte élevée. Or, en pratique, les codes convolutifs utilisés ont une longueur de contrainte relativement faible (≤ 10). En effet, les codes ayant des longueurs de contraintes élevées ont certes une capacité de correction élevée mais la complexité du décodeur de Viterbi croît de manière exponentielle avec cette longueur de contrainte. Par conséquent, l'utilisation d'un code à longueur de contrainte élevée rendrait la complexité du décodeur rédhibitoire pour les transmissions actuelles. Un moyen simple, pour d'un côté garantir une grande distance libre et d'un autre côté une complexité de décodage raisonnable, est de combiner plusieurs codes.

4.2 Concaténation de codes

Le premier schéma de codage de ce type fut proposé dans [For66]. La figure 4.1(a) représente ce schéma qui était, à l'origine, composé d'un premier codeur, appelé codeur externe, et d'un second codeur, appelé codeur interne. Afin d'augmenter la robustesse de ce schéma de codage, une fonction de permutation ou d'entrelacement peut-être placée entre le code externe et le code interne, voir figure 4.1(b).

Cette concaténation de code est aujourd'hui appelée concaténation série et l'information est codée deux fois, une première fois par le code externe et une seconde par le code interne. Dans cette configuration, les deux codes utilisés sont en règle générale complémentaires. En effet, le code interne est généralement un code convolutif et le code externe un code RS (Reed-Solomon). La combinaison de ces deux codes permet d'obtenir de bonnes performances en terme de correction des erreurs puisque le code convoltuif permet de lutter efficacement contre les erreurs isolées alors

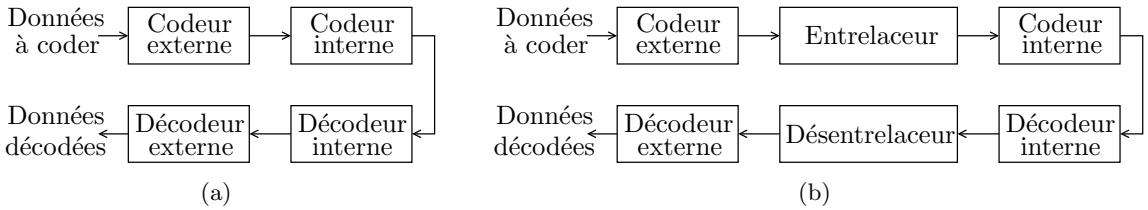


Figure 4.1 — Schéma d'un code concaténé sans et avec entrelacement

que le code RS est adéquat pour lutter contre les erreurs de type *burst*. Le décodeur convolutif s'occupera des erreurs aléatoires pour les faibles rapports signal sur bruit, tandis que le décodeur RS s'occupera des erreurs par paquets pour des rapports signal sur bruit plus élevés. On retrouve ce type de concaténation dans les standards de diffusion par satellite (norme DVB-S [DVB09]) ou par diffusion hertzienne terrestre (norme DVB-T).

Dans cette version, l'enchaînement simple des deux décodeurs élémentaires n'est pas optimal, en effet le décodeur interne ne tire pas profit de la redondance introduite par le code externe. Cette observation a conduit à l'invention d'un nouveau schéma de codage/décodage s'approchant à $0.5dB$ de la limite de Shannon sur canal gaussien, les turbocodes. Ce schéma de codage/décodage fut inventé par C. Berrou et al. [BGT93]. La figure 4.2 représente un turbocode qui est une concaténation en parallèle de deux codes séparés par un entrelaceur. Dans sa version d'origine, les deux codeurs étaient des codes convolutifs RSC et les sorties systématiques du second codeur étaient poinçonnées, c-à-d qu'elles n'étaient pas transmises. Mais depuis cette invention, le principe des turbocodes s'est généralisé et les deux codeurs ne sont plus nécessairement des codes convolutifs RSC. De par le fait qu'ils allient une complexité de décodage raisonnable à d'excellentes performances de correction, les turbocodes ont connu, depuis leur invention, un très grand succès. En effet, de nombreux standards, en particuliers en radio-mobile (tel que l'UMTS [3GP05b] et le CDMA-2000 [3GP09]), utilisent ces schémas de codage afin de protéger leurs données du bruit.

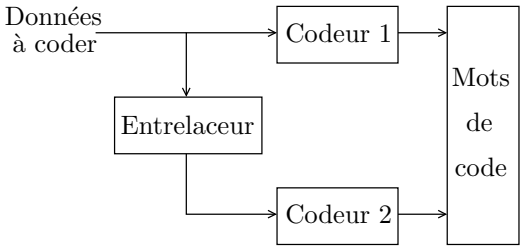


Figure 4.2 — Schéma d'un turbocode

4.2.1 Les entrelaceurs

La fonction d'entrelacement et de désentrelacement joue un rôle prédominant dans les codes concaténés. En effet, l'entrelacement permet de disperser temporellement les erreurs. Dans l'hypothèse que les erreurs interviennent par paquets, cette dispersion les transformera en erreurs isolées. De plus, un entrelaceur choisi judicieusement permettra d'augmenter la distance minimale globale du code concaténé. Effectivement, la distance minimale du code concaténé est déterminée conjointement par les deux codeurs et l'entrelaceur. Comme pour les codeurs, il existe deux grandes classes d'entrelaceur, les entrelaceurs par blocs et les entrelaceurs convolutifs (encore appelé convolutionnels ou multiplexés).

Les entrelaceurs par blocs prennent des blocs de symboles en entrée d'une certaine longueur que nous noterons l_e. Ils effectuent une permutation sur ces symboles au sein du même bloc et restituent en sortie le bloc de symboles permutés. Au contraire, les entrelaceurs convolutifs ne prennent pas les

symboles d'entrée par blocs mais prennent en entrée un flux de symboles et effectuent la permutation en flux tendu à l'aide de registres à décalage superposés et de tailles différentes. Dans la suite de ce document, nous nous intéresserons uniquement aux entrelaceurs par blocs.

Un entrelaceur par bloc est représenté, soit par un vecteur de permutation de taille l_e, soit par une matrice d'entrelacement de taille $l_e \times l_e$. Précisons que la taille de l'entrelaceur l_e n'est pas choisie au hasard mais dépend des paramètres des codeurs. Nous noterons p le vecteur de permutation et E la matrice d'entrelacement. Cette matrice est composée d'un "1" unique sur chaque colonne et ligne et de "0" ailleurs. En notant $\mathbf{m}$ un vecteur de taille l_e, alors la version entrelacée de ce vecteur, notée $\mathbf{m}'$, est définie par :

$$\begin{cases} \mathbf{m}' = \mathbf{m}.E \\ \mathbf{m}' = \mathbf{m}(p) \end{cases} \tag{4.1}$$

Lorsque l'on considère plusieurs blocs, il est possible de représenter l'entrelaceur global par une matrice bloc-diagonale, notée E_g, qui sera composée sur sa diagonale de la matrice d'entrelacement E. Notons L la taille du vecteur $\mathbf{m}$ (avec $L = \alpha.l_e$), alors la version entrelacée de ce vecteur, notée $\mathbf{m}'$, est telle que :

$$\mathbf{m}' = \mathbf{m}.E_g \tag{4.2}$$

avec E_g une matrice de taille $L \times L$ composée de α matrices E :

$$E_g = \begin{pmatrix} E & & \\ & \ddots & \\ & & E \end{pmatrix} \tag{4.3}$$

Exemple 4.45.

Prenons l'exemple d'un entrelaceur de taille 8 qui est défini par le vecteur de permutation suivant :

$$p = \begin{bmatrix} 4 & 5 & 8 & 1 & 6 & 7 & 3 & 2 \end{bmatrix}$$

D'après le vecteur de permutation p, la matrice d'entrelacement est telle que :

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

Notons $\mathbf{m}$ le vecteur initial de taille l_e :

$$\mathbf{m} = \begin{pmatrix} m(0) & m(1) & m(2) & m(3) & m(4) & m(5) & m(6) & m(7) \end{pmatrix}$$

alors la version entrelacée de ce vecteur, $\mathbf{m}' = \mathbf{m}(p) = \mathbf{m}.E$, est telle que :

$$\mathbf{m}' = \begin{pmatrix} m(3) & m(4) & m(7) & m(0) & m(5) & m(6) & m(2) & m(1) \end{pmatrix}$$

4.2.2 Les turbocodes série

Nous avons vu que les codes concaténés en série étaient en règle générale composés de deux codes différents (convolutif et RS), mais ils peuvent également être composés de deux codes convolutifs. La

figure 4.2.2 représente une concaténation série de deux codes convolutifs qui est également appelée turbocode série.

Figure 4.3 — Schéma d'une concaténation série

La multiplication du rendement des deux codes convolutifs r_1 et r_2 permet d'obtenir le rendement global du turbocode série, noté r_s :

$$r_s = r_1.r_2 = \frac{k_1.k_2}{n_1.n_2} = \frac{k_s}{n_s} \tag{4.4}$$

4.2.3 Les turbocodes parallèle

Comme nous l'avons précisé, dans sa version d'origine, un turbocode était composé de deux codes convolutifs séparés par un entrelaceur. Le second codeur était un code RSC dont les voies systématiques étaient poinçonnées. Si cette version reste celle qui est la plus utilisée dans les standards, un turbocode n'est pas forcément tel que décrit ci-dessus. En effet, les voies systématiques ne sont pas nécessairement poinçonnées et les deux codes ne sont pas forcément des codes convolutifs. Dans ces travaux, nous nous intéresserons uniquement au turbocode convolutif, soit un turbocode composé de deux codes convolutifs. Afin de distinguer les deux types de turbocodes, nous nommerons *turbocode*, un turbocode dont le second codeur est de forme RSC avec ses voies systématiques poinçonnées et *code concaténé en parallèle*, un turbocode qui est composé de deux codeurs de forme RSC ou NRNSC et où il n'y a pas forcément de poinçonnage des voies systématiques. Dans la littérature, ces derniers schémas de codage sont couramment appelés PCCC pour "Parallel Concatenated Convolutional Code", mais dans ce mémoire nous les nommerons simplement CCP pour "Code Concaténé en parallèle" puisque nous traiterons uniquement des schémas de codage à base de codes convolutifs.

Dans le cas des turbocodes parallèles, la taille de l'entrelaceur est étroitement liée aux paramètres des deux codeurs :

- $l_e = \alpha_1.k_1$
- $l_e = \alpha_2.k_2$
- $l_e >> max(K_1, K_2)$

Dans la suite de ce manuscrit, nous ferons l'hypothèse que ces contraintes sur la taille de l'entrelaceur sont respectées.

4.2.3.1 Les codes concaténés en parallèle

La figure 4.4 représente un code concaténé en parallèle qui est composé de deux codes convolutifs (RSC ou NRNSC) et d'un entrelaceur. Nous noterons r_1 le rendement du premier codeur et r_2 celui du second, alors le rendement global du CCP (noté r_{par}) est tel que :

$$r_{par} = \frac{r_1.r_2}{r_1 + r_2} = \frac{k_1.k_2}{k_1.n_2 + k_2.n_1} \tag{4.5}$$

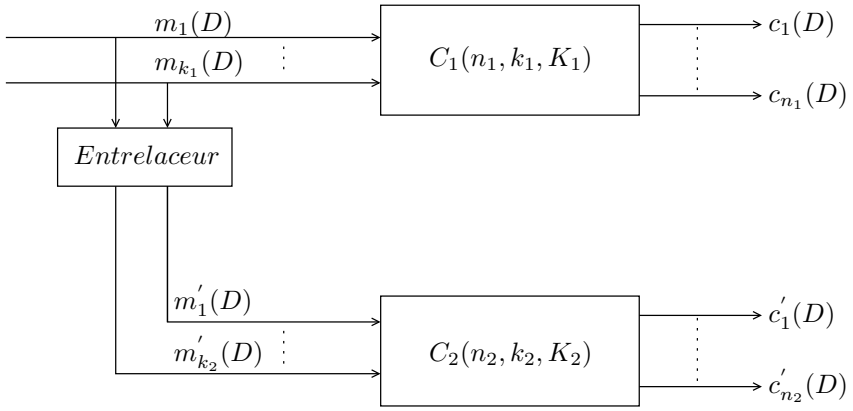


Figure 4.4 — Représentation d'un code concaténé en parallèle

Le vecteur d'entrée du code concaténé est représenté par un vecteur noté $\mathbf{m}$ et la version entrelacée de ce vecteur est notée $\mathbf{m}'$. Le vecteur contenant les sorties du premier codeur est noté $\mathbf{c}$ et celui du second codeur $\mathbf{c}'$:

$$\begin{cases} \mathbf{c} = \begin{pmatrix} \mathbf{c}(0) & \mathbf{c}(1) & \cdots \end{pmatrix} \\ \mathbf{c}' = \begin{pmatrix} \mathbf{c}'(0) & \mathbf{c}'(1) & \cdots \end{pmatrix} \end{cases} \tag{4.6}$$

où $\mathbf{c}(t) = \begin{pmatrix} c_1(t) & \cdots & c_{n_1}(t) \end{pmatrix}$ et $\mathbf{c}'(t) = \begin{pmatrix} c'_1(t) & \cdots & c'_{n_2}(t) \end{pmatrix}$.

Nous noterons $\mathbf{x}$ le vecteur de sortie du CCP qui est tel que :

$$\mathbf{x} = \begin{pmatrix} \mathbf{c}(0) & \mathbf{c}'(0) & \mathbf{c}(1) & \mathbf{c}'(1) & \cdots \end{pmatrix} \tag{4.7}$$

4.2.3.2 Les turbocodes

Dans le cas d'un turbocode, le deuxième codeur $C_2(n_2, k_2, K_2)$ est un codeur RSC et les voies systématiques de ce codeur $(c'_1(D), \cdots, c'_{k_2}(D))$ ne sont pas transmises. Le premier codeur $C_1(n_1, k_1, K_1)$ peut être de forme RSC ou NRNSC. La figure 4.5 représente un turbocode tel que présenté ci-dessus. Les voies systématiques du second codeur qui ne sont pas transmises sont représentées par des traits en pointillé sur cette figure.

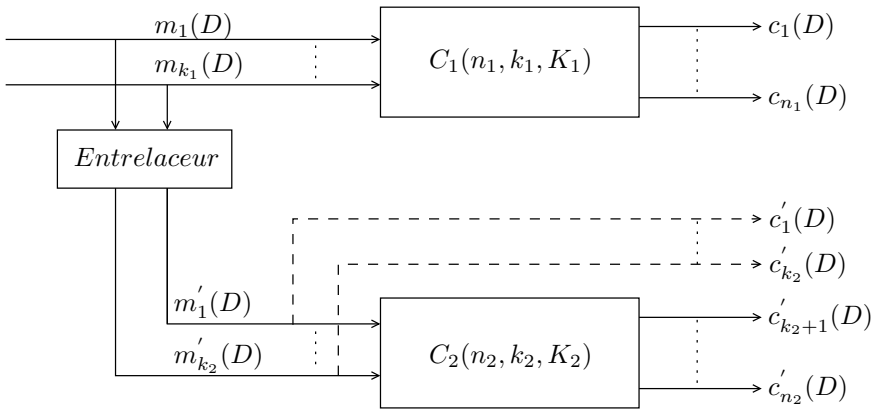


Figure 4.5 — Représentation d'un turbocode

Dans cette configuration, le rendement du turbocode est défini par :

$$r_t = \frac{r_1.r_2}{r_1 + r_2 - r_1.r_2} = \frac{k_1.k_2}{k_1.n_2 + k_2.n_1 - k_1.k_2} \tag{4.8}$$

Le vecteur d'entrée du turbocode est représenté par un vecteur noté $\mathbf{m}$ et la version entrelacée de ce vecteur est notée $\mathbf{m}'$. Le vecteur contenant les sorties du premier codeur est noté $\mathbf{c}$ et celui du second codeur $\mathbf{c}'$:

$$\begin{cases} \mathbf{c} = \begin{pmatrix} \mathbf{c}(0) & \mathbf{c}(1) & \cdots \end{pmatrix} \\ \mathbf{c}' = \begin{pmatrix} \mathbf{c}'(0) & \mathbf{c}'(1) & \cdots \end{pmatrix} \end{cases} \tag{4.9}$$

où $\mathbf{c}(t) = \begin{pmatrix} c_1(t) & \cdots & c_{n_1}(t) \end{pmatrix}$ et $\mathbf{c}'(t) = \begin{pmatrix} c'_{k_2+1}(t) & \cdots & c'_{n_2}(t) \end{pmatrix}$.

Nous noterons $\mathbf{x}$ le vecteur de sortie du turbocode qui est tel que :

$$\mathbf{x} = \begin{pmatrix} \mathbf{c}(0) & \mathbf{c}'(0) & \mathbf{c}(1) & \mathbf{c}'(1) & \cdots \end{pmatrix} \tag{4.10}$$

Les turbocodes parallèles, soit les CCP et les turbocodes, étant les schémas de codage les plus utilisés en pratique (comparé au turbocode série), nous nous intéresserons, dans la suite de ce document, uniquement à la reconnaissance aveugle de ces deux schémas de codage. Le lecteur intéressé par plus d'information sur la théorie des turbocodes et la concaténation de codes pourra se référer à [Ber07].

La littérature concernant cette thématique, soit la reconnaissance aveugle des turbocodes, est peu riche. De plus dans les travaux existants [Bar05, Bar07a], de nombreuses hypothèses restrictives ont été faites. En effet, ils faisaient l'hypothèse que les deux codes C_1 et C_2 étaient identiques et qu'ils avaient accès aux mots de code $\mathbf{c}$ et $\mathbf{c}'$. Par conséquent, l'identification des deux codes du turbocode ne différaient pas de la reconnaissance classique d'un code convolutif. Dans la suite de ce chapitre, nous proposerons deux méthodes d'identification, l'une portant sur les CCP et l'autre sur les turbocodes. Nous ferons l'hypothèse que la seule information connue est le vecteur $\mathbf{x}$. Rappelons que ce vecteur est composé des mots de code $\mathbf{c}$ et $\mathbf{c}'$ multiplexés. De ce fait, avant d'obtenir les mots de code $\mathbf{c}$ et $\mathbf{c}'$, nous devrons tout d'abord identifier le nombre de sorties de chaque code, n_1 et n_2, afin de pouvoir démultiplexer le vecteur $\mathbf{x}$. Dans ces travaux, le vecteur $\mathbf{x}$ sera tel que :

$$\mathbf{x} = \begin{pmatrix} \mathbf{c}(0) & \mathbf{c}'(0) & \mathbf{c}(1) & \mathbf{c}'(1) & \cdots \end{pmatrix} \tag{4.11}$$

et nous ne prendrons aucune autre forme de multiplexage en compte.

Afin de développer nos méthodes de reconnaissance aveugle, nous utiliserons plusieurs schémas de CCP et de turbocode que nous avons récapitulé dans le tableau ci-dessous.

Nom du code	Code C_1	Code C_2	Taille de l'entrelaceur
CCP[1]	Code NRNSC : $C_1(2,1,3)$	Code NRNSC : $C_2(2,1,3)$	$l_e = 10$
CCP[2]	Code NRNSC : $C_1(2,1,3)$	Code NRNSC : $C_2(3,1,4)$	$l_e = 18$
CCP[3]	Code NRNSC : $C_1(3,2,3)$	Code RSC : $C_2(3,2,5)$	$l_e = 24$
Turbocode[1]	Code NRNSC : $C_1(3,2,3)$	Code RSC : $C_2(3,2,5)$	$l_e = 12$

Tableau 4.1 — Tableaux récapitulatif des schémas de codage étudiés

4.3 Reconnaissance aveugle des codes concaténés en parallèle

Dans cette partie, nous présenterons un algorithme qui nous permettra d'identifier en aveugle l'ensemble des paramètres du CCP, soit les deux codes convolutifs et la fonction d'entrelacement. Nous ferons l'hypothèse que le train binaire est synchronisé et non-entaché d'erreur. L'algorithme que nous présenterons sera découpé en trois étapes. La première étape consistera à identifier les paramètres du CCP, soit son rendement et la taille de l'entrelaceur. La seconde étape permettra d'identifier le nombre de sorties de chaque code. Une fois ces paramètres n_1 et n_2 connus, nous serons en mesure de séparer les pistes de codés qui appartiennent à chacun des codeurs. La dernière étape consistera à identifier les deux codes et le vecteur de permutation.

Dans la suite de ce chapitre, nous prendrons différents exemples de CCP et de turbocode. De ce fait, un tableau récapitulatif des différents CCP et turbocode utilisés est présenté dans le tableau 4.1 page 118 de ce chapitre.

4.3.1 Identification des paramètres des codes concaténés en parallèle

Dans la première partie du chapitre 2, nous avons présenté une méthode qui permettait d'identifier l'ensemble des paramètres d'un code convolutif. Cette méthode était basée sur le calcul du rang des matrices construites avec les mots de code. Afin d'identifier les paramètres du CCP, nous utiliserons cette même méthode. Dans le cas d'un simple code convolutif, nous avons constaté que les matrices de taille $\alpha.n \geq n_a$ présentaient des déficiences de rang. Dans le cas d'un CCP, de par la présence de deux codes convolutifs et de l'entrelaceur, nous verrons apparaître deux sortes de chutes de rang. En effet, les deux codes ainsi que l'entrelaceur vont engendrer des chutes de rang maximales et il apparaîtra des chutes de rang moyennes qui seront uniquement liées à la présence des deux codeurs.

Nous noterons R_l les matrices construites avec les données codées par le CCP, soit le vecteur $\mathbf{x}$. Ces matrices seront de taille $M \times l$ avec $M > l$. Afin d'identifier les paramètres du CCP, nous calculerons le rang dans $GF(2)$ de chacune de ces matrices. Avant de présenter notre méthode d'identification, nous introduirons quelques notations relatives aux deux codes convolutifs du CCP.

– **Notations des codes convolutifs :**

Nous noterons $\mu_1^\perp$ la mémoire du code dual $C_1^\perp$ et $\mu_2^\perp$ la mémoire de $C_2^\perp$. Nous avons démontré dans l'annexe B, que la matrice de parité d'un code convolutif était composée des polynômes générateurs du codeur sous sa forme RSC équivalente. De ce fait, la mémoire du code dual est identique à la mémoire du codeur sous sa forme RSC équivalente. En notant μ_j^i la mémoire de la i-ème entrée du j-ème code, les mémoires des codes RSC équivalents sont définies par :

$$\mu_1^\perp = \sum_{i=1}^{k_1} \mu_1^i \quad \text{et} \quad \mu_2^\perp = \sum_{i=1}^{k_2} \mu_2^i \tag{4.12}$$

Nous avons montré dans le second chapitre que lors de la reconnaissance d'un unique code convolutif, la première matrice de rang déficient était de taille $n_a = \alpha.n > n$. Nous noterons n_{a1} et n_{a2} les tailles des premières matrices de rang déficient pour les codes C_1 et C_2, respectivement :

$$n_{a_1} = n_1.\left\lfloor \frac{\mu_1^\perp}{n_1 - k_1} + 1 \right\rfloor \tag{4.13}$$

et

$$n_{a_2} = n_2.\left\lfloor \frac{\mu_2^\perp}{n_2 - k_2} + 1 \right\rfloor \tag{4.14}$$

4.3.1.1 Les chutes de rang maximales

La combinaison de l'entrelaceur et des deux codeurs va engendrer, lors du calcul du rang des matrices R_l, une chute de rang maximale tous les multiples de la taille de l'entrelaceur divisé par le rendement du CCP, soit $\alpha.\frac{l_e}{r_{par}}$. La première chute de rang maximale apparaîtra pour $l = \frac{l_e}{r_{par}}$ et la différence entre les rangs de deux matrices correspondant à des chutes de rang maximales successives correspondra à la taille de l'entrelaceur. L'équation de la droite passant par ces chutes de rang maximales est telle que :

$$rg(R_l) = l.r_{par} + \mu^\perp \qquad \forall l = \alpha.\frac{l_e}{r_{par}} \tag{4.15}$$

où $\mu^{\perp}$ est défini par :

$$\mu^{\perp} = \mu_1^{\perp} + \mu_2^{\perp} \tag{4.16}$$

Connaissant deux matrices correspondant à des chutes de rang maximales, nous serons en mesure d'identifier le rendement du CCP et la taille de l'entrelaceur. Nous noterons n_m la taille de la première matrice présentant une chute de rang maximale telle que :

$$n_m = \frac{l_e}{r_{par}} \tag{4.17}$$

1. Identification de l_e :
   La différence entre les rangs de deux matrices correspondant à des chutes de rang maximales successives correspond à la taille de l'entrelaceur :

$$l_e = rg\left(R_{(\alpha+1).n_m}\right) - rg\left(R_{\alpha.n_m}\right) \quad \forall \alpha \in \mathbb{N} \tag{4.18}$$

2. Identification de r_{par} :
   D'après l'équation (4.17), le rendement du CCP est :

$$r_{par} = \frac{l_e}{n_m} \tag{4.19}$$

3. Identification de $\mu^{\perp}$:
   La somme des mémoires des codeurs RSC équivalents est :

$$\mu^{\perp} = rg\left(R_{n_m}\right) - n_m.r_{par} \tag{4.20}$$

Exemple 4.46.

Prenons l'exemple d'un code concaténé en parallèle composé de deux codes convolutifs identiques, de rendement 1/2 et de longueur de contrainte égale à 3, de mémoire égale à 2 et d'un entrelaceur de taille 10. Le premier codeur sera noté $C_1(2,1,3)$ et le second $C_2(2,1,3)$. Afin de différencier les différents CCP que nous étudierons par la suite, nous noterons ce CCP le CCP^1. Le rendement du CCP^1, r_{par}, est tel que :

$$r_{par} = \frac{1}{4}$$

L'entrelaceur est défini par le vecteur de permutation suivant :

$$p = \begin{pmatrix} 6 & 8 & 1 & 7 & 5 & 4 & 10 & 9 & 3 & 2 \end{pmatrix}$$

Notons $\mathbf{m}$ le vecteur d'entrée et $\mathbf{m}'$ sa version entrelacée telle que :

$$\mathbf{m} = \begin{pmatrix} m(0) & m(1) & m(2) & m(3) & m(4) & m(5) & m(6) & m(7) & m(8) & m(9) & \cdots \end{pmatrix}$$

$$\begin{aligned} \mathbf{m}' &= (m'(0) \quad m'(1) \quad m'(2) \quad m'(3) \quad m'(4) \quad m'(5) \quad m'(6) \quad m'(7) \quad m'(8) \quad m'(9) \quad \cdots) \\ &= (m(5) \quad m(7) \quad m(0) \quad m(6) \quad m(4) \quad m(3) \quad m(9) \quad m(8) \quad m(2) \quad m(1) \quad \cdots) \end{aligned}$$

Les sorties du premier codeur, notées $c_1(t)$ et $c_2(t)$, et les sorties du second codeur, $c'_1(t)$ et $c'_2(t)$, sont définies par :

$$\begin{cases} c_1(t) = m(t) + m(t-2) \\ c_2(t) = m(t) + m(t-1) + m(t-2) \\ c'_1(t) = m'(t) + m'(t-2) \\ c'_2(t) = m'(t) + m'(t-1) + m'(t-2) \end{cases}$$

Alors, le vecteur $\mathbf{x}$ obtenu en sortie du CCP[1] est tel que :

$$\begin{aligned}\mathbf{x} &= \begin{pmatrix} x(0) & x(1) & x(2) & x(3) & x(4) & x(5) & x(6) & x(7) & \cdots \end{pmatrix} \\ &= \begin{pmatrix} c_1(0) & c_2(0) & c'_1(0) & c'_2(0) & c_1(1) & c_2(1) & c'_1(1) & c'_2(1) & \cdots \end{pmatrix}\end{aligned}$$

Nous avons montré dans le chapitre 1, que dans le cas des codes de rendement $1/n$, la relation liant la matrice génératrice du code NRNSC et de son RSC équivalent est :

$$G_{RSC}(D) = \frac{G_{NRNSC}(D)}{g_{1,1}(D)}$$

De ce fait, la mémoire du codeur NRNSC et de son RSC équivalent est identique. La mémoire du $C(2,1,3)$ code est : $\mu = K - 1 = 2$ et celle de son RSC équivalent est $\mu^{\perp} = \mu = 2$. Le CCP [1] étant composé de deux codes $C(2,1,3)$, les mémoires $\mu_1^{\perp}$ et $\mu_2^{\perp}$ sont telles que :

$$\mu_1^{\perp} = \mu_2^{\perp} = K - 1 = 2$$

Nous avons représenté sur la figure 4.6 le rang des matrices R_l construites avec les données $\mathbf{x}$, pour l variant de 1 à 80.

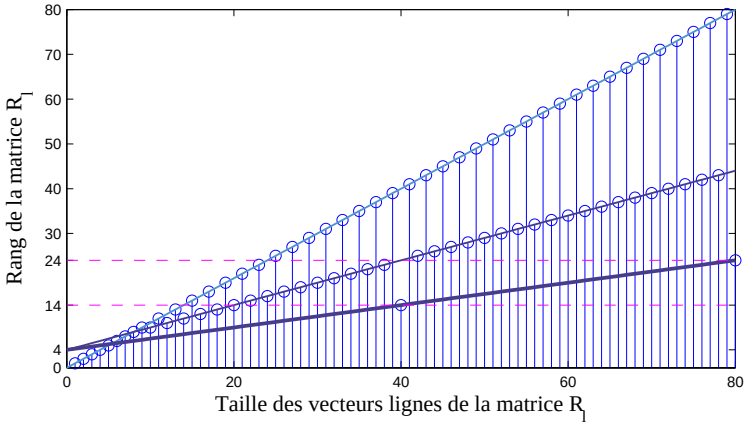


Figure 4.6 — Rang des matrices R_l du CCP[1] (cf tableau 4.1)

Dans un premier temps, nous nous intéresserons uniquement aux chutes de rang maximales. Nous remarquons sur cette figure, une première chute de rang maximale lorsque la matrice est de taille 40 et une seconde pour la matrice de taille 80. Connaissant le rang de ces deux matrices, nous serons en mesure d'identifier la taille de l'entrelaceur (4.18), le rendement du CCP[1] (4.19) et la somme des mémoires des deux codes (4.20).

$$\begin{cases} l_e = rg(80) - rg(40) = 24 - 14 = 10 \\ r_{par} = \frac{10}{40} = \frac{1}{4} \\ \mu^{\perp} = 14 - 40.\frac{1}{4} = 4 \end{cases}$$

Nous pouvons vérifier que les paramètres identifiés avec notre méthode de reconnaissance correspondent aux paramètres du CCP[1].

4.3.1.2 Les chutes de rang moyennes

Outres ces chutes de rang maximales, nous verrons apparaître des chutes de rang moyennes qui seront uniquement liées aux deux codes. L'équation de la droite passant par ces chutes de rang

moyennes est telle que :

$$rg(R_l) = \frac{k}{n}.l + \mu^{\perp}, \quad \forall l = \alpha.n \geq n_a \text{ et } l \neq \alpha.\frac{l_e}{.r_{par}} \tag{4.21}$$

avec n_a qui correspond à la taille de la première matrice de rang déficient :

$$n_a = n.\left\lfloor \frac{\mu^{\perp}}{n-k} + 1 \right\rfloor \tag{4.22}$$

Les valeurs de n et k peuvent avoir deux valeurs différentes en fonction des codes C_1 et C_2.

– Cas 1 :

Dans ce premier cas, les valeurs de k et n seront telles que :

$$\begin{cases} k = k_1 = k_2 \\ n = n_1 = n_2 \end{cases} \tag{4.23}$$

– Cas 2 :

En revanche, dans ce second cas, les valeurs de n et k seront telles que :

$$\begin{cases} k = k_1 + k_2 \\ n = n_1 + n_2 \end{cases} \tag{4.24}$$

Connaissant deux matrices de rang déficient, nous serons en mesure d'identifier les paramètres listés ci-dessous.

1. Identification de n :
   La différence entre la taille de deux matrices de rang déficients consécutives correspond à la valeur de n :
   $$n = (n_a + (\alpha+1).n) - (n_a + \alpha.n) \quad \forall \alpha \in \mathbb{N} \tag{4.25}$$
2. Identification de k :
   La différence entre le rang de deux matrices de rang déficient correspond à la valeur de k :
   $$k = rg\left(R_{n_a+(\alpha+1).n}\right) - rg\left(R_{n_a+\alpha.n}\right) \quad \forall \alpha \in \mathbb{N} \tag{4.26}$$
3. Identification de $\mu^{\perp}$:
   En connaissant les valeurs de k et n, la valeur de $\mu^{\perp}$ est :
   $$\mu^{\perp} = rg\left(R_{n_a+\alpha.n}\right) - \left(n_a + \alpha.n\right).\frac{k}{n} \quad \forall \alpha \in \mathbb{N} \tag{4.27}$$

Exemple 4.47.

Suite de l'exemple 4.46.

Sur la figure 4.6 de l'exemple précédent, nous remarquons que, outre les chutes de rang maximales, nous pouvons observer des chutes de rang moyennes. En effet, nous remarquons une première chute de rang moyenne lorsque la matrice est de taille 10 (soit $n_a = 10$), puis des chutes de rang moyennes tous les multiples de 2. D'après les équations (4.25), (4.26) et (4.27), connaissant le rang de deux matrices de rang déficient consécutives, nous sommes en mesure d'identifier les paramètres suivants :

$$\begin{cases} n = 12 - 10 = 2 \\ k = rg(12) - rg(10) = 10 - 9 = 1 \\ \mu^{\perp} = rg(10) - \frac{k}{n}.10 = 4 \end{cases}$$

4.3.2 Identification du nombre de sorties des deux codes

Une fois les paramètres du CCP identifiés, nous allons nous intéresser à l'identification du nombre de sorties de chaque codeur afin de pouvoir séparer les données. En effet, si nous identifions ces paramètres, nous serons en mesure de différencier les bits qui appartiennent aux mots de code du codeur C_1 de ceux qui appartiennent au codeur C_2, ce qui nous permettra ensuite d'identifier les deux codes.

Nous noterons L la longueur de la trame reçue et nous réorganiserons les données sous la forme d'une matrice, notée Y, qui sera de taille $L/n \times n$, telle que :

$$Y = \begin{pmatrix} x(0) & \cdots & x(n-1) \\ x(n) & \cdots & x(2.n-1) \\ \vdots & \vdots & \vdots \end{pmatrix} \tag{4.28}$$

où $x(i)$ correspond au i-ème bit de $\mathbf{x}$.

Connaissant n, il suffit de faire une recherche sur le nombre de sorties d'un des codeurs dans l'intervalle $[1, n-1]$. Ainsi, si nous voulons identifier le nombre de sorties du premier codeur, il suffit de supprimer pour une valeur de $\hat{n_1}$ (variant de 1 à $n-1$) donnée, les $(n - \hat{n_1})$ dernières colonnes de la matrice Y qui sont supposées correspondre aux données codées par le codeur C_2. Le principe de cette méthode sera de réorganiser ces données sous forme de matrices et de calculer leurs rangs normalisés dans $GF(2)$. Nous noterons $\tilde{R}_l$, les matrices construites avec ces données et $rg(\tilde{R}_l)$ le rang de ces matrices, alors le rang normalisé, noté $\rho(l)$, est tel que :

$$\rho(l) = \frac{rg(\tilde{R}_l)}{l} \tag{4.29}$$

Lorsque le bon nombre de colonnes de Y aura été supprimé, c-à-d lorsque $\hat{n_1}$ sera égale à n_1, la matrice $\tilde{R}_l$ présentera une chute de rang maximale. Par conséquent, en calculant le rang des matrices $\tilde{R}_l$, nous pourrons aisément identifier la tailles des mots de code de C_1 et de ce fait en déduire la taille des mots de code de C_2 :

$$n_2 = n - n_1 \tag{4.30}$$

En prenant uniquement les données liées au codeur C_1, nous savons que la première matrice présentant une chute de rang est de taille n_{a1} :

$$n_{a1} = n_1 . \left\lfloor \frac{\mu_1^{\perp}}{n_1 - k_1} + 1 \right\rfloor \tag{4.31}$$

Par conséquent, afin d'observer une chute de rang maximale lorsque $\hat{n_1} = n_1$, il est nécessaire que la matrice $\tilde{R}_l$ soit composée d'au minimum n_{a1} colonnes. De plus, nous savons que si la matrice est de taille $\alpha.\frac{l_e}{r_{par}}$, nous verrons apparaître des chutes de rang qui seront liées à la présence de l'entrelaceur. Par conséquent, afin d'analyser les chutes de rang qui sont uniquement liées à la présence des codeurs, nous devrons réorganiser les données sous la forme de matrices, $\tilde{R}_l$, de taille $L/l \times l$ avec $l = \hat{n_1}.(l_e - 1)$. La taille d'un entrelaceur étant grande devant les paramètres d'un code convolutif, en choisissant cette valeur de l pour construire la matrice $\tilde{R}_l$, nous pouvons vérifier que $l > n_{a_1}$ et que $l \neq \alpha.\frac{l_e}{r_{par}}$.

Nous avons vu qu'en fonction des deux codes, C_1 et C_2, les valeurs de n et k avaient deux comportements différents :

- Cas 1 : $n = n_1 = n_2$
- Cas 2 : $n = n_1 + n_2$

Dans le cas 2 ($n = n_1 + n_2$), en appliquant l'algorithme précédent, nous arriverons à identifier

les valeurs de n_1 et de n_2. En revanche, dans le cas 1 ($n = n_1 = n_2$), en appliquant le même algorithme, nous serons dans l'incapacité d'identifier les valeurs de n_1 et de n_2. En effet, dans cette configuration, toutes les matrices $\tilde{R}_l$ seront de rang plein. Si une telle situation se produit, nous doublerons la valeur de n et nous ré-appliquerons l'algorithme précédent avec cette nouvelle valeur de n pour identifier n_1 et n_2.

Exemple 4.48.

Suite de l'exemple 4.47.

Dans l'exemple précédent, nous avons identifié le paramètre $n = 2$. Nous réorganiserons les données sous la forme d'une matrice notée Y qui sera de taille $L/2 \times 2$:

$$Y = \begin{pmatrix} x(0) & x(1) \\ x(2) & x(3) \\ x(4) & x(5) \\ x(6) & x(6) \\ \vdots & \vdots \end{pmatrix} = \begin{pmatrix} c_1(0) & c_2(0) \\ c'_1(0) & c'_2(0) \\ c_1(1) & c_2(1) \\ c'_1(1) & c'_2(1) \\ \vdots & \vdots \end{pmatrix}$$

La valeur de n étant égale à 2, le principe de la méthode serait de supprimer la dernière colonne de Y et de réorganiser ces données sous la forme d'une matrice de taille $L/(\hat{n_1}.(l_e - 1)) \times \hat{n_1}.(l_e - 1)$ (avec $\hat{n_1} = 1$). Or, nous pouvons remarquer qu'en supprimant la dernière colonne de Y, nous aurons uniquement les bits correspondant aux premières sorties des deux codes, soit $c_1(t)$ et $c'_1(t)$. Par conséquent la matrice construite avec ces données sera de rang plein. Dans cette configuration, nous poserons $n = n + n = 4$ et nous ré-appliquerons la même méthode.

Dans ce cas, la matrice Y de taille $L/4 \times 4$ est telle que :

$$Y = \begin{pmatrix} x(0) & x(1) & x(2) & x(3) \\ x(4) & x(5) & x(6) & x(6) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} c_1(0) & c_2(0) & c'_1(0) & c'_2(0) \\ c_1(1) & c_2(1) & c'_1(1) & c'_2(1) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Nous ferons varier la valeur de $\hat{n_1}$ de 1 à $n - 1$ et nous supprimerons, à chaque fois, les $n - \hat{n_1}$ dernières colonnes de la matrice Y. Puis, avec ces données nous construirons les matrices $\tilde{R}_l$ avec l qui sera telle que :

$$l = \hat{n_1}.(l_e - 1) = \hat{n_1}.9$$

Pour $\hat{n_1} = 1$, nous devons supprimer les $n - \hat{n_1} = 3$ dernières colonnes de la matrice Y et réorganiser ces données sous la forme d'une matrice de taille 9. Après suppression des 3 colonnes de Y, la matrice $\tilde{R}_9$ est telle que

$$\tilde{R}_9 = \begin{pmatrix} c_1(0) & c_1(1) & c_1(2) & c_1(3) & c_1(4) & c_1(5) & c_1(6) & c_1(7) & c_1(8) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Cette matrice sera de rang plein puisqu'elle est uniquement composée des bits correspondant à la première sortie du codeur C_1.

Pour $\hat{n_1} = 2$, après la suppression des 2 dernières colonnes de Y, nous devons réorganiser les données sous la forme d'une matrice de taille $l = 18$:

$$\tilde{R}_{18} = \begin{pmatrix} c_1(0) & c_2(0) & c_1(1) & c_2(1) & \cdots & c_1(7) & c_2(7) & c_1(8) & c_2(8) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Dans ce cas, nous avons uniquement des bits correspondant au premier codeur. Par conséquent,

le rang de cette matrice sera :

$$rg(\tilde{R}_{18}) = 18.\frac{k_1}{n_1} + \mu_1^{\perp} = 18.\frac{1}{2} + 2 = 11$$

Puis, pour $\hat{n_1} = 3$, il faut supprimer la dernière colonne de Y et réorganiser les données sous la forme d'une matrice de taille $l = 27$. Cette matrice sera composée des bits correspondant au premier codeur et à la première sortie du codeur C_2. Le rang de cette matrice sera : $rg(\tilde{R}_{27}) = 20$.

Nous avons représenté sur la figure 4.7 le rang normalisé des matrices $\tilde{R}_l$ en fonction de $\hat{n_1}$.

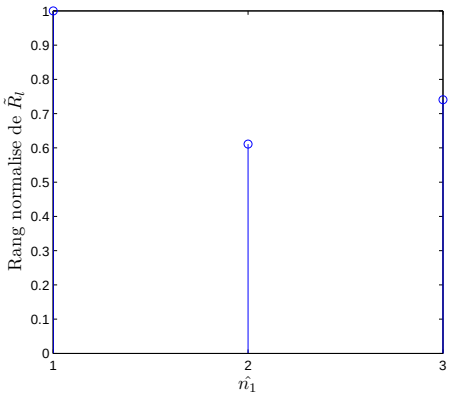


Figure 4.7 — Identification de n_1 du CCP[1] (cf tableau 4.1)

Nous remarquons sur cette figure que nous avons obtenu une chute de rang maximale pour $\hat{n_1} = 2$. Nous pouvons en conclure que $n_1 = 2$ et $n_2 = n - n_1 = 2$. Nous pouvons vérifier que ces deux valeurs correspondent au nombre de sorties des codeurs C_1 et C_2.

Prenons maintenant l'exemple d'un CCP composé de deux codes différents.

Exemple 4.49.

Prenons l'exemple d'un CCP, que nous noterons CCP[2], composé d'un premier codeur NRNSC : $C_1(2, 1, 3)$, d'un second codeur NRNSC : $C_2(3, 1, 4)$ et d'un entrelaceur de taille 18. Le rendement global du CCP[2] est tel que :

$$r_{par} = \frac{1}{5} = 0.2$$

Nous avons représenté sur la figure 4.8, le rang des matrices R_l construites avec les mots de code du CCP[2] en fonction de l, avec l variant de 1 à 180. Nous remarquons sur cette figure, la présence de chutes de rang maximales et de chutes de rang moyennes. Afin d'identifier le rendement du CCP[2] et la taille de l'entrelaceur, nous allons nous intéresser dans un premier temps aux chutes de rang maximales. La première chute de rang maximale apparaît pour $l = 90$ et la seconde pour $l = 180$. Connaissant le rang de ces deux matrices, nous pourrons d'après les équations (4.18) et (4.19) identifier la taille de l'entrelaceur et le rendement du CCP[2] :

$$l_e = rg(R_{180}) - rg(R_{90}) = 41 - 23 = 18$$

$$r_{par} = l_e/90 = 0.2$$

Puis, en nous servant des chutes de rang moyennes, nous pourrons identifier les paramètres n et k. Nous remarquons sur la figure 4.8 que la première chute de rang moyenne apparaît lorsque la matrice est de taille 10, puis pour toutes les matrices de taille $\alpha.5 \geq 10$. D'après les

équations (4.25) et (4.26), les paramètres n et k sont tels que :

$$\begin{cases} n = 5 \\ k = rg(R_{15}) - rg(R_{10}) = 11 - 9 = 2 \end{cases}$$

Afin d'identifier le nombre de sorties des deux codeurs, nous ferons varier la valeur de $\hat{n_1}$ de 1 à $n - 1 = 4$ et nous calculerons le rang normalisé des matrices $\tilde{R}_l$. Sur la figure 4.9 nous avons représenté le rang normalisé de ces matrices en fonction $\hat{n_1}$.

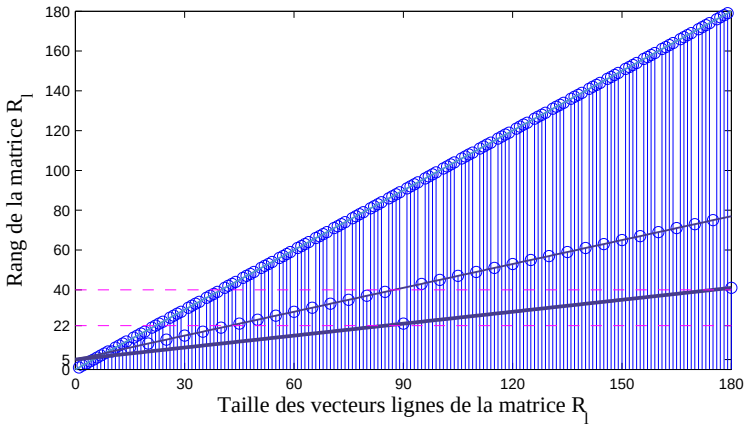


Figure 4.8 — Rang des matrices R_l du CCP² (cf tableau 4.1)

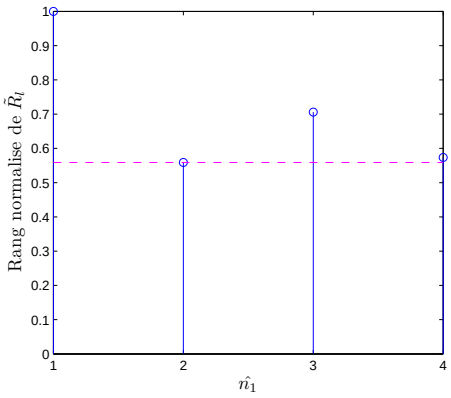


Figure 4.9 — Identification de n_1 du CCP² (cf tableau 4.1)

Nous observons une chute de rang maximale lorsque la valeur de $\hat{n_1}$ est égale à 2. De ce fait, nous pouvons conclure que :

$$\begin{cases} n_1 = 2 \\ n_2 = n - n_1 = 3 \end{cases}$$

Nous pouvons vérifier que l'ensemble des paramètres identifiés correspondent aux paramètres du CCP².

4.3.3 Identification des deux codes et de l'entrelaceur

Le nombre de sorties de chaque codeur, n_1 et n_2, ayant été identifié dans l'étape précédente, nous sommes en mesure de séparer les pistes de codés. Ainsi, nous connaîtrons le vecteur $\mathbf{c}$, qui correspond aux sorties de C_1 ainsi que le vecteur $\mathbf{c}'$, qui correspond aux sorties de C_2. Par conséquent, en appliquant les algorithmes de reconnaissance aveugle des codes convolutifs présentés dans

le chapitre 2, nous serons en mesure d'identifier les deux codes, C_1 avec la connaissance des mots de code $\mathbf{c}$ et C_2 avec les mots de code $\mathbf{c}'$.

Dans le chapitre 2, nous avons montré que les deux types de code, RSC et NRNSC, étaient traités de manière identique par nos algorithmes de reconnaissance. En revanche, nous avons vu que nos méthodes nous permettaient uniquement d'obtenir le codeur sous sa forme NRNSC et que sans connaissance a priori sur la nature du code, nous étions incapables de distinguer s'il s'agissait au départ d'un code de forme RSC ou NRNSC. Par contre, dans le cas d'un CCP, nous montrerons que l'identification du vecteur de permutation nous permettra également d'identifier la forme des deux codes.

Nous noterons $G_{1_{(NRNSC)}}$ et $G_{2_{(NRNSC)}}$ les matrices génératrices identifiées avec la méthode du chapitre 2 pour les codeurs C_1 et C_2. Ces matrices correspondent à des matrices génératrices d'un code NRNSC, elles sont donc composées de simples polynômes générateurs. D'après les propriétés énoncées dans le chapitre 1, nous sommes capables, à partir des matrices $G_{1_{(NRNSC)}}$ et $G_{2_{(NRNSC)}}$, d'écrire les matrices génératrices des codes RSC équivalents, que nous noterons respectivement $G_{1_{(RSC)}}$ et $G_{2_{(RSC)}}$. Si les matrices identifiées sont correctes, alors nous savons que l'un des quatre couples de codeurs listés ci-dessous correspond aux deux codes du CCP.

- $(G_{1_{(NRNSC)}}, G_{2_{(NRNSC)}})$
- $(G_{1_{(NRNSC)}}, G_{2_{(RSC)}})$
- $(G_{1_{(RSC)}}, G_{2_{(NRNSC)}})$
- $(G_{1_{(RSC)}}, G_{2_{(RSC)}})$

De ce fait, afin d'identifier le vecteur de permutation et les deux codeurs, nous devrons tester les quatre couples précédemment cités.

Dans le but d'identifier le couple de codeurs et le vecteur de permutation, nous décoderons les sorties de chaque codeur, $\mathbf{c}$ et $\mathbf{c}'$ à l'aide de l'algorithme de Viterbi. Pour chaque code, nous devrons décoder les sorties avec le codeur sous sa forme NRNSC et son codeur RSC équivalent. Nous noterons $\mathbf{m}_{(NRNSC)}$ et $\mathbf{m}_{(RSC)}$ les mots d'information obtenus en décodant les mots de code $\mathbf{c}$ avec le codeur C_1 sous sa forme NRNSC et RSC, respectivement. Les données $\mathbf{c}'$ décodées avec le codeur C_2 sous sa forme NRNSC et RSC équivalent seront notées $\mathbf{m}'_{(NRNSC)}$ et $\mathbf{m}'_{(RSC)}$.

Faisons l'hypothèse que le bon couple de codeurs a été utilisé pour le décodage des mots de code. Alors, les vecteurs d'information obtenus lors du décodage des sorties $\mathbf{c}$ et $\mathbf{c}'$ correspondent aux vecteurs $\mathbf{m}$ et $\mathbf{m}'$. La taille de l'entrelaceur, l_e, ayant été identifiée précédemment, afin d'identifier un vecteur de permutation de taille l_e, il suffit de comparer des blocs de taille l_e entre $\mathbf{m}$ et $\mathbf{m}'$. Le principe sera de réorganiser les données d'entrée $\mathbf{m}$ et $\mathbf{m}'$ sous forme de matrices, notées respectivement M et M'. Il ne sera pas nécessaire d'utiliser toutes les données pour identifier le vecteur de permutation, par conséquent nous nous limiterons à des matrices M et M' de taille $l_e \times l_e$ telle que :

$$M = \begin{pmatrix} m(0) & m(1) & \cdots & m(l_e-1) \\ m(l_e) & m(l_e+1) & \cdots & m(2.l_e-1) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \quad M' = \begin{pmatrix} m'(0) & m'(1) & \cdots & m'(l_e-1) \\ m'(l_e) & m'(l_e+1) & \cdots & m'(2.l_e-1) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{4.32}$$

De ce fait, nous savons qu'il existe un vecteur de permutation, noté p, de taille l_e qui permet de passer de la matrice M à la matrice M'. Nous noterons E_p la matrice de permutation correspondant au vecteur de permutation, telle que :

$$[E_p]_{i,j} = \begin{cases} 1 \text{ si } i = p(j) \\ 0 \end{cases} \tag{4.33}$$

Alors, la relation de passage entre la matrice M et la matrice M' est donnée par l'expression suivante :

$$M' = M.E_p \tag{4.34}$$

Nous noterons $\mathbf{m}_i$ la i-ème colonne de M et $\mathbf{m}'_j$ la j-ème colonne de M'. Afin d'identifier le vecteur p, nous chercherons les indices i et j des colonnes de M et M' qui sont tels que :

$$w\left(\mathbf{m}_i + \mathbf{m}'_j\right) = 0 \tag{4.35}$$

alors, le j-ème élément du vecteur de permutation est tel que :

$$p(j) = i \tag{4.36}$$

Cette méthode d'identification du vecteur de permutation est représentée dans l'algorithme 7.

Algorithme 7 : Identification du vecteur de permutation

Entrées : les vecteurs $\mathbf{m}$ et $\mathbf{m}'$ et la taille de l'entrelaceur l_e
Sorties : le vecteur de permutation p'

Construire la matrice M avec $\mathbf{m}$;
Construire la matrice M' avec $\mathbf{m}'$;
pour $j = 1$ *à* l_e **faire**
  pour $i = 1$ *à* l_e **faire**
    si $w\left(\mathbf{m}_i + \mathbf{m}'_j\right) == 0$ **alors**
      $p'(j) = i$;
    fin
  fin
fin

Lors de la reconnaissance aveugle du CCP, nous ne connaissons pas la forme des deux codes, par conséquent il sera nécessaire d'appliquer l'algorithme 7 sur les quatre couples de vecteurs d'entrées possibles :

- $(\mathbf{m}_{(NRNSC)}, \mathbf{m}'_{(NRNSC)})$
- $(\mathbf{m}_{(NRNSC)}, \mathbf{m}'_{(RSC)})$
- $(\mathbf{m}_{(RSC)}, \mathbf{m}'_{(NRNSC)})$
- $(\mathbf{m}_{(RSC)}, \mathbf{m}'_{(RSC)})$

Lorsque le bon couple de codeurs aura été utilisé pour le décodage des mots de code, nous obtiendrons un vecteur de permutation. En revanche, si le mauvais couple de codeurs a été utilisé, nous n'obtiendrons pas de vecteur de permutation de taille l_e. Par conséquent, nous serons en mesure en sortie de cet algorithme d'identifier la forme des deux codeurs ainsi que le vecteur de permutation.

Nous avons montré dans le chapitre 2, que lors de l'identification d'un codeur de rendement k/n, il restait en sortie une indétermination sur l'ordre des lignes de la matrice génératrice. Dans le cas de l'identification d'un CCP, il restera toujours cette indétermination. En effet, si l'un des codeurs identifiés correspond au code de départ mais à une permutation près sur l'ordre des lignes, nous identifierons tout de même un vecteur de permutation mais ce vecteur ne correspondra pas au vecteur initiale. De ce fait, une fois le bon couple de codeurs identifié, si l'un des codeurs est de rendement k/n, il restera en sortie de cet algorithme une indétermination sur l'ordre des lignes de la matrice génératrice du ou des codeurs de rendement k/n et par conséquent sur l'ordre du vecteur de permutation.

Nous avons également vu dans le chapitre 2 que selon le codeur utilisé, en sortie de notre algorithme de reconnaissance nous pouvions obtenir un ensemble de matrices génératrices équivalentes. Ces matrices sont évidemment toutes de forme NRNSC et de par leurs équivalences, il était mathématiquement impossible de les distinguer les unes des autres. En revanche, lors de l'identification d'un CCP, nous serons en mesure de sélectionner parmi l'ensemble des matrices génératrices équivalentes identifiées, celle qui correspond au codeur C_1 ou C_2. Faisons l'hypothèse que nous avons identifié une matrice génératrice pour le codeur C_1 et trois matrices génératrices pour le codeur C_2. Dans ce cas, nous n'aurons plus quatre couples de codeurs à tester mais douze couples de codeurs :

- Quatre couples avec le codeur C_1 et la première matrice génératrice du codeur C_2
- Quatre couples avec le codeur C_1 et la deuxième matrice génératrice du codeur C_2
- Quatre couples avec le codeur C_1 et la troisième matrice génératrice du codeur C_2

Au final, nous obtiendrons un unique couple, parmi ces douze, qui nous permettra d'identifier un vecteur de permutation de taille l_e. Ainsi, nous obtiendrons l'ensemble des paramètres du code concaténé en parallèle.

Exemple 4.50.

Pour les CCP[1] et CCP[2], nous avons identifié dans les étapes précédentes, le rendement du CCP, la taille de l'entrelaceur et le nombre de sorties de chaque code. Nous sommes donc en mesure de séparer les pistes de codés afin d'obtenir les mots de code $\mathbf{c}$ et $\mathbf{c}'$. Les deux codes convolutifs de ces deux CCP (CCP[1] et CCP[2]) sont des codes de rendement $1/n$. Par conséquent, en appliquant l'algorithme du chapitre 2 pour identifier les codeurs, nous obtiendrons systématiquement les bonnes matrices génératrices. Par conséquent, il suffit pour ces deux CCP de tester les quatre couples de codeurs afin d'obtenir conjointement le vecteur de permutation de taille l_e et le bon couple de codeurs.

Dans le but de bien comprendre la méthode d'identification des deux codes et du vecteur de permutation, nous prendrons l'exemple d'un troisième CCP.

Exemple 4.51.

Prenons l'exemple d'un CCP[3] de paramètres :

- Code NRNSC de paramètres $C_1(3, 2, 3)$ et de matrice génératrice :

$$G_1 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 7 \end{pmatrix}$$

- Code RSC de paramètres $C_2(3, 2, 5)$ et de matrice génératrice :

$$G_2 = \frac{1}{23}.\begin{pmatrix} 23 & 0 & 31 \\ 0 & 23 & 27 \end{pmatrix}$$

- Un entrelaceur de taille $l_e = 24$ et le vecteur de permutation suivant :

$$p = (2, 4, 14, 8, 7, 16, 21, 6, 15, 22, 13, 18, 5, 11, 19, 24, 3, 1, 17, 12, 20, 23, 9, 10)$$

- Le rendement du CCP[3] $r_{par} = \frac{1}{3}$

La première étape de notre algorithme de reconnaissance consiste à calculer le rang des matrices R_l construites avec les données obtenues en sortie du CCP[3]. Cette première étape nous permettra

d'identifier le rendement du CCP[3], la taille de l'entrelaceur et le paramètre n. Sur la figure 4.10, nous avons représenté le rang des matrices R_l en fonction de l.

D'après les équations (4.18), (4.19) et (4.25), nous sommes en mesure d'identifier les paramètres suivants :

– Identification de l_e :

$$l_e = rg(R_{144}) - rg(R_{72}) = 24$$

– Identification de r_{par} :

$$r_{par} = l_e/72 = 0.333$$

– Identification de n :

$$n = 30 - 24 = 6$$

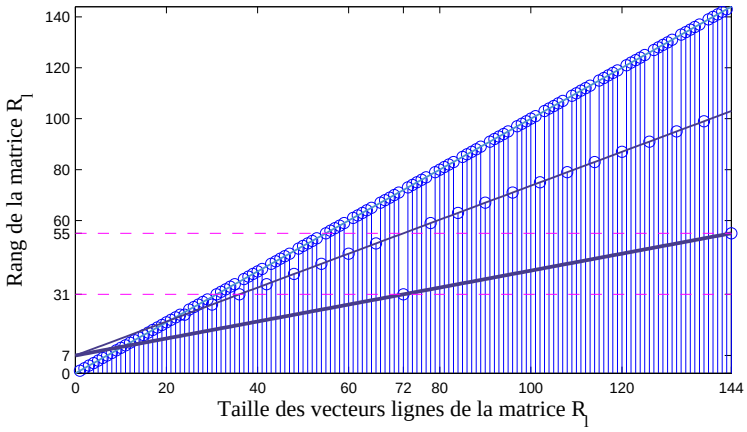


Figure 4.10 — Rang des matrices R_l du CCP[3] (cf tableau 4.1)

La seconde étape a pour objectif d'identifier la taille des mots de code de chaque code, soit les valeurs de n_1 et n_2. Sur la figure 4.11, nous avons représenté le rang normalisé des matrices $\tilde{R}_l$ en fonction de $\hat{n_1}$ (qui varie de 1 à $n - 1 = 5$).

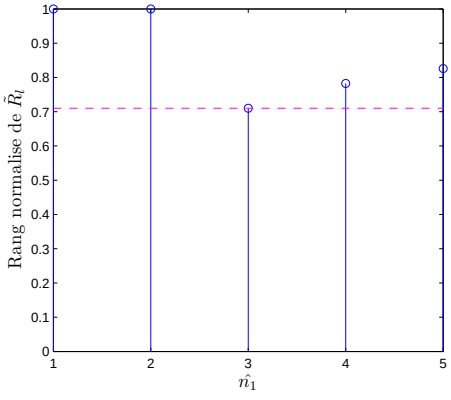


Figure 4.11 — Identification de n_1 du CCP[3] (cf tableau 4.1)

Nous remarquons sur cette figure que la matrice présentant la chute de rang maximale est obtenue pour $\hat{n_1} = 3$. Par conséquent, les paramètres n_1 et n_2 sont tels que :

$$n_1 = 3$$

et

$$n_2 = n - n_1 = 3$$

Connaissant le nombre de sorties de chaque codeur, nous sommes en mesure de séparer les bits appartenant aux mots de code de C_1, $\mathbf{c}$, de ceux appartenant au codeur C_2, $\mathbf{c}'$. Puis, en appliquant l'algorithme présenté dans le chapitre 2, nous obtenons les paramètres suivants :

– Les paramètres du codeur C_1 identifiés sont :

$$\begin{cases} n_1 = 3 \\ k_1 = 2 \\ K_1 = 3 \end{cases}$$

et nous identifions deux matrices génératrices équivalentes :

$$G_{1_{(NRNSC)}} = \begin{pmatrix} 4 & 1 & 7 \\ 1 & 2 & 3 \end{pmatrix}$$

$$G'_{1_{(NRNSC)}} = \begin{pmatrix} 1 & 2 & 3 \\ 6 & 5 & 1 \end{pmatrix}$$

– Les paramètres du codeur C_2 identifiés sont :

$$\begin{cases} n_2 = 3 \\ k_2 = 2 \\ K_2 = 3 \end{cases}$$

et nous identifions la matrice génératrice suivantes :

$$G_{2_{(NRNSC)}} = \begin{pmatrix} 2 & 5 & 7 \\ 7 & 4 & 1 \end{pmatrix}$$

Nous allons maintenant identifier conjointement le vecteur de permutation et la forme des deux codeurs. Nous réaliserons tout d'abord un essai en prenant les matrices $G_{1_{(NRNSC)}}$ et $G_{2_{(NRNSC)}}$ puis en prenant $G'_{1_{(NRNSC)}}$ et $G_{2_{(NRNSC)}}$.

– Test avec $G_{1_{(NRNSC)}}$ et $G_{2_{(NRNSC)}}$:

Avec ces deux matrices, nous devrons décoder les sorties $\mathbf{c}$ et $\mathbf{c}'$ avec le codeur sous sa forme NRNSC et RSC. Nous obtiendrons au final quatre vecteurs d'entrées possibles :

$$\begin{cases} \text{pour le code } C_1 : \mathbf{m}_{(NRNSC)} \text{ et } \mathbf{m}_{(RSC)} \\ \text{pour le code } C_2 : \mathbf{m}'_{(NRNSC)} \text{ et } \mathbf{m}'_{(RSC)} \end{cases}$$

En appliquant l'algorithme 7, sur les quatre couples de vecteurs d'entrées possible, nous obtenons au final un unique couple qui permet d'obtenir un vecteur de permutation de taille $l_e = 24$. En effet, seul le couple $\left(G_{1_{(NRNSC)}}, G_{2_{(RSC)}}\right)$ permet d'obtenir un vecteur de permutation. En prenant le code C_1 de forme NRNSC et de matrice génératrice :

$$G_{1_{(NRNSC)}} = \begin{pmatrix} 4 & 1 & 7 \\ 1 & 2 & 3 \end{pmatrix}$$

et le code C_2 de forme RSC et de matrice génératrice :

$$G_{2_{(RSC)}} = \frac{1}{23}\begin{pmatrix} 23 & 0 & 31 \\ 0 & 23 & 27 \end{pmatrix}$$

nous obtenons le vecteur de permutation suivant :

$$p' = (1, 3, 13, 7, 8, 15, 22, 5, 16, 21, 14, 17, 6, 12, 20, 23, 4, 2, 18, 11, 19, 24, 10, 9)$$

– Test avec $G'_{1_{(NRNSC)}}$ et $G_{2_{(NRNSC)}}$:

En réalisant le même traitement, en ne prenant plus la matrice $G_{1_{(NRNSC)}}$ mais la matrice $G'_{1_{(NRNSC)}}$, nous obtenons au final aucun vecteur de permutation de taille l_e.

Nous pouvons conclure que les paramètres des deux codes du CCP^3 et le vecteur de permutation sont :

– Code NRNSC de paramètres $C_1(3, 2, 3)$ et de matrice génératrice :

$$G_{1_{(NRNSC)}} = \begin{pmatrix} 4 & 1 & 7 \\ 1 & 2 & 3 \end{pmatrix}$$

– Code RSC de paramètres $C_2(3, 2, 5)$ et de matrice génératrice :

$$G_{2_{(RSC)}} = \frac{1}{23}.\begin{pmatrix} 23 & 0 & 31 \\ 0 & 23 & 27 \end{pmatrix}$$

– Un entrelaceur de taille $l_e = 24$ et le vecteur de permutation suivant :

$$p' = (1, 3, 13, 7, 8, 15, 22, 5, 16, 21, 14, 17, 6, 12, 20, 23, 4, 2, 18, 11, 19, 24, 10, 9)$$

– Le rendement du CCP est tel que $r_{par} = \frac{1}{3}$

Comparons maintenant ces résultats aux paramètres du CCP^3. Tout d'abord, nous pouvons vérifier que les paramètres et la forme des deux codeurs ont bien été identifiés ainsi que la taille de l'entrelaceur et le rendement du CCP. Nous pouvons également noter que la matrice génératrice $G_{2_{(RSC)}}$ correspond à la matrice génératrice de C_2 (G_2). En revanche, si nous comparons la matrice $G_{1_{(NRNSC)}}$ avec la matrice G_1, nous notons qu'elles sont identiques mais à une permutation sur les lignes près. De ce fait, le vecteur de permutation p' ne correspond pas au vecteur p. Mais, si nous permutons l'ordre des lignes de la matrice $G_{1_{(NRNSC)}}$ et que nous décodons les sorties $\mathbf{c}$ avec ce codeur, alors en comparant ces mots d'information au vecteur $\mathbf{m}'_{(RSC)}$, nous obtiendrons le bon vecteur de permutation.

Nous avons donc réussi à identifier l'ensemble des paramètres du CCP^3. Il restera en sortie de cet algorithme une indétermination sur l'ordre des lignes de la matrice génératrice du codeur C_1.

Nous venons de vous présenter dans cette partie un algorithme qui nous a permis, dans le cas d'une transmission sans bruit, d'identifier l'ensemble des paramètres d'un code concaténé en parallèle, c-à-d le code C_1, l'entrelaceur et le code C_2. Il reste en sortie de cet algorithme uniquement une indétermination sur l'ordre des lignes des matrices génératrices (comme dans le cadre de la reconnaissance aveugle d'un code convolutif dans le chapitre 2).

4.4 Reconnaissance aveugle des turbocodes

Dans cette partie, nous nous intéresserons à la reconnaissance aveugle des turbocodes. Rappelons que nous appelons turbocode, un code concaténé en parallèle dont le codeur C_2 est un code RSC et ses voies systématiques sont poinçonnées. Dans cette configuration, la méthode d'identification d'un code concaténé en parallèle pourra être partiellement utilisée afin d'identifier le turbocode. En effet, nous montrerons que l'identification des paramètres du turbocode ainsi que l'identification du premier codeur restent identiques. En revanche, les voies systématiques du second codeur ayant été poinçonnées, nous ne serons pas en mesure d'identifier ce codeur et l'entrelaceur avec les méthodes précédentes.

4.4.1 Identification des paramètres du turbocode et du premier codeur

Nous avons montré que le rendement d'un turbocode était :

$$r_t = \frac{k_1.k_2}{k_1.n_2 + k_2.n_1 - k_1.k_2} \tag{4.37}$$

En notant n_2' le nombre de sorties du second codeur après le poinçonnage de ses voies systématiques :

$$n_2' = n_2 - k_2, \tag{4.38}$$

le rendement du turbocode est tel que :

$$r_t = \frac{k_1.k_2}{k_1.n_2' + k_2.n_1} \tag{4.39}$$

La méthode permettant d'identifier les paramètres du turbocode est identique à celle utilisée pour les codes concaténés en parallèle. En effet, le calcul du rang des matrices construites avec les données turbocodées nous permettra d'identifier les paramètres de celui-ci. Comme précédemment, la combinaison des deux codes et de l'entrelaceur va engendrer des chutes de rang maximales et nous verrons également des chutes de rang moyennes qui seront uniquement liées à la présence des deux codes.

Afin d'identifier les deux codeurs et l'entrelaceur d'un CCP, nous pouvons préciser que les paramètres du CCP essentiels à identifier sont :

- La taille de l'entrelaceur l_e
- La valeur de n
- Le nombre de sorties des deux codes n_1 et n_2

En effet, la méthode que nous vous avons présenté permettait également d'identifier le rendement du CCP, la somme des mémoires des codes duaux ainsi que la valeur de k. Or, ces paramètres ne sont pas essentiels pour l'identification des deux codes et de l'entrelaceur. En effet, une fois les paramètres n_1, n_2 et l_e identifiés, nous sommes en mesure de reconnaître les deux codes et l'entrelaceur. De ce fait, dans le cadre de la reconnaissance aveugle d'un turbocode, nous nous intéresserons plus particulièrement à l'identification de ses paramètres l_e, n, n_1 et n_2'.

- Les chutes de rang maximales :

Comme dans le cas d'un CCP, il apparaîtra des chutes de rang maximales tous les multiples de la taille de l'entrelaceur divisée par le rendement du turbocode, soit $\alpha.\frac{l_e}{r_t}$. La différence entre le rang de deux matrices correspondant à des chutes de rang maximales successives correspond à la taille

de l'entrelaceur et la pente de la droite passant par ces chutes de rang est égale au rendement du turbocode. En revanche, l'ordonnée à l'origine de l'équation de la droite passant par ces chutes de rang est variable et sa valeur exacte en fonction des deux codes est actuellement en cours d'étude. Néanmoins, connaissant le rang de deux matrices correspondant à des chutes de rang maximales successives, nous seront en mesure d'identifier le rendement du turbocode et la taille de l'entrelaceur.

– Les chutes de rang moyennes :

L'équation exacte de la droite passant par les chutes de rang moyenne est actuellement en cours d'étude. En effet, de par le poinçonnage des voies systématiques, cette équation est plus complexe que dans le cas d'un CCP. Mais, comme dans le cas d'un CCP, il apparaîtra des chutes de rang moyennes tous les multiples de $\alpha.n$. Ainsi, connaissant deux matrices de rang déficient consécutives, nous serons en mesure d'identifier la valeur de n. Comme dans le cas d'un CCP, pour la valeur de n, on peut avoir deux cas différents : $n = n_1 = n'_2$ ou $n = n_1 + n'_2$.

Exemple 4.52.

Prenons l'exemple d'un turbocode de paramètres :

– $C_1(3, 2, 3)$ de forme NRNSC et de matrice génératrice :

$$G_1 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 7 \end{pmatrix}$$

– $C_2(3, 2, 5)$ de forme RSC et de matrice génératrice :

$$G_2 = \frac{1}{23}. \begin{pmatrix} 23 & 0 & 31 \\ 0 & 23 & 27 \end{pmatrix}$$

– Un entrelaceur de taille $l_e = 12$:

$$p = (10, 11, 4, 2, 6, 3, 9, 12, 7, 8, 5, 1)$$

– le rendement du turbocode est :

$$r_t = \frac{2}{4} = \frac{1}{2}$$

Par la suite, nous noterons turbocode[1] ce turbocode et ses paramètres sont récapitulés dans le tableau 4.1 (voir page 118).

Nous avons représenté sur la figure 4.12 le rang des matrices R_l construites avec les données obtenues en sortie du turbocode[1], pour l variant de 1 à 72. Nous remarquons sur cette figure une première chute de rang maximale pour $l = 24$ et une seconde pour $l = 48$. Connaissant le rang de ces deux matrices, nous sommes en mesure d'identifier l_e et r_t :

$$l_e = rg(R_{48}) - rg(R_{24}) = 31 - 19 = 12$$

$$r_t = \frac{12}{24} = \frac{1}{2}$$

La différence de taille entre deux matrices de rang déficient successives, nous permet d'identifier la valeur de n :

$$n = 4$$

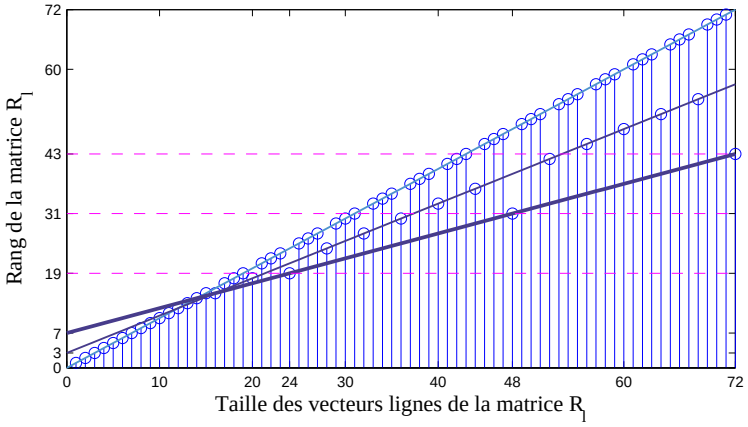


Figure 4.12 — Rang des matrices R_l du turbocode[1] (cf tableau 4.1)

Une fois la valeur de n identifiée, nous nous intéresserons à l'identification du nombre de sorties de chaque codeur, soit les valeurs de n_1 et n_2'. Pour les identifier, nous utiliserons la même méthode que dans le cas d'un CCP (voir la sous-section 4.3.2).

Exemple 4.53.

Suite de l'exemple 4.52.

Connaissant le paramètre n, nous pourrons identifier à l'aide de la méthode décrite dans la partie 4.3.2, le nombre de sorties de chaque code. Sur la figure 4.13, nous avons représenté le rang normalisé des matrices $\tilde{R}_l$ en fonction de $\hat{n_1}$ (qui varie de 1 à $n-1=3$).

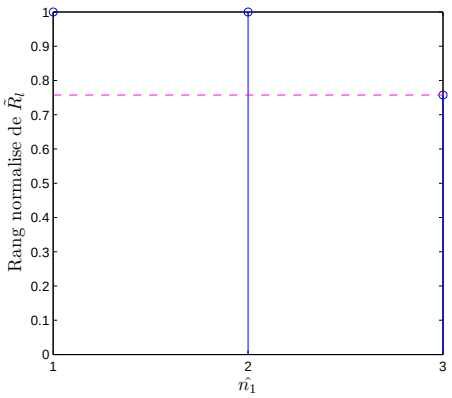


Figure 4.13 — Identification de n_1 du turbocode[1] (cf tableau 4.1)

Nous remarquons sur cette figure que nous obtenons une chute de rang maximale pour $\hat{n_1}=3$. Nous pouvons donc conclure que :

$$\begin{cases} n_1 = 3 \\ n_2' = n - n_1 = 1 \end{cases}$$

Nous pouvons vérifier que ces paramètres sont corrects puisque après le poinçonnage des 2 voies systématiques du second codeur, nous obtenons une unique voie de sortie donc $n_2' = 1$.

Connaissant le nombre de sorties des deux codes, nous pouvons aisément séparer les bits appartenant aux mots de code de C_1 et les bits appartenant aux mots de code de C_2. De ce fait, nous connaîtrons le vecteur $\mathbf{c}$ et $\mathbf{c}'$. A partir des mots de code de C_1, nous pouvons appliquer les algorithmes présentés dans le chapitre 2 pour identifier le codeur. En revanche, de par le poinçonnage

des voies systématiques, il sera impossible d'utiliser ces mêmes méthodes pour identifier le codeur C_2.

Faisons l'hypothèse que le codeur C_2 est de rendement $(n_2 - 1)/n_2$. Dans cette configuration $k_2 = n_2 - 1$, ce qui implique qu'après le poinçonnage des voies systématiques, il restera en sortie du codeur une unique voie. Nos méthodes de reconnaissance étant basées sur la redondance introduite par le code, nous ne pourrons les utiliser dans cette configuration puisque nous n'aurons plus aucune redondance après le poinçonnage des $(n_2 - 1)$ voies systématiques. Nous proposerons dans la prochaine partie de ce mémoire une approche qui permettra de pallier ce problème.

4.4.2 Identification du second codeur

4.4.2.1 Les hypothèses et notations

Dans cette partie, nous ferons l'hypothèse que le codeur C_1 du turbocode a été parfaitement identifié par la méthode précédente. Ainsi, nous serons en mesure de décoder ses sorties ($\mathbf{c}$) afin d'obtenir le message informatif $\mathbf{m}$. L'idée que nous avons proposée dans [MGB09a] et [MGB09c] est d'utiliser ce message informatif afin d'identifier les paramètres du second codeur et de l'entrelaceur.

La situation la plus compliquée pour l'identification du second codeur est lorsqu'il est de rendement $(n_2 - 1)/n_2$. En effet, si $k_2 < n_2 - 1$, il restera après poinçonnage des k_2 voies systématiques au moins 2 voies de sorties, ce qui implique qu'il y aura encore de la redondance. En revanche, si $k_2 = n_2 - 1$, il restera uniquement une voie de sortie après poinçonnage, donc nous n'aurons plus aucune redondance. De ce fait, nous ferons l'hypothèse que le codeur C_2 est de rendement $(n_2 - 1)/n_2$, ce qui correspond au cas le plus difficile.

Afin de simplifier les notations dans cette partie, nous noterons n le nombre de sorties du second codeur ($n = n_2$), k le nombre d'entrées du second codeur ($k = k_2 = n_2 - 1$), K sa longueur de contrainte ($K = K_2$) et μ sa mémoire ($\mu = K - 1$). Sur la figure 4.14, nous avons représenté le schéma que nous avons pour identifier le codeur C_2 et l'entrelaceur.

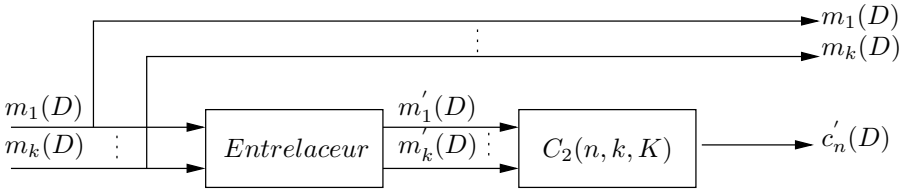


Figure 4.14 — Entrelaceur suivi du codeur C_2

Afin d'identifier le codeur $C_2(n, k, K)$ et l'entrelaceur nous aurons connaissance des séquences d'entrée $[m_1(D), \cdots, m_k(D)]$, de la séquence de sortie du second codeur $c'_n(D)$ et de la taille de l'entrelaceur.

- **Hypothèses fixées :**
  - Les données sont non-entachées d'erreurs
  - Le train binaire est synchronisé
  - Le code convolutif RSC est de rendement $\frac{n-1}{n}$
  - La taille de l'entrelaceur $l_e > (n-1).K$

Nous noterons $\mathbf{m}$ le vecteur contenant les mots d'information, $\mathbf{m}'$ la version entrelacée du vecteur $\mathbf{m}$ et $\mathbf{c'}_n$ le vecteur contenant les éléments binaires de la n-ième sortie du second codeur.

- **Le codage :**

Nous noterons $G(D)$ la $(n-1)\times n$ matrice génératrice du codeur $C_2(n,k,K)$ et $f_{i,j}(D)$ ces polynômes générateurs :

$$G(D) = \frac{1}{f_{1,1}(D)} \begin{bmatrix} f_{1,1}(D) & & f_{1,n}(D) \\ & \ddots & \vdots \\ & f_{1,1}(D) & f_{n-1,n}(D) \end{bmatrix} \tag{4.40}$$

Le lien entre les séquences d'entrée et de sortie est défini par :

$$c'(D) = m'(D).G(D) \tag{4.41}$$

où $c'(D) = \left[c'_1(D), \cdots, c'_{n-1}(D), c'_n(D)\right]$ et $m'(D) = \left[m'_1(D), \cdots, m'_{n-1}(D)\right]$.

Les $(n-1)$ premières sorties $\left[c'_1(D), \cdots, c'_{n-1}(D)\right]$ étant poinçonnées, nous nous intéresserons uniquement à l'expression de la n-ième sortie :

$$c'_n(D) = m'(D).\frac{1}{f_{1,1}(D)}.\begin{bmatrix} f_{1,n}(D) \\ \vdots \\ f_{n-1,n}(D) \end{bmatrix} \tag{4.42}$$

Dans le chapitre 1, nous avons défini la matrice de codage F (1.24) qui était composée des K sous-matrices de codage F_l (1.22) ($\forall l = 0, \cdots, K-1$). Ces sous-matrices étaient composées des l-ièmes coefficients binaires des $k \times n$ polynômes générateurs de la matrice génératrice. Dans notre contexte, la partie de la matrice génératrice du codeur C_2 qui nous intéresse, est simplement composée des $(n-1)$ polynômes générateurs suivants : $(f_{1,n}(D), \cdots, f_{n-1,n}(D))$. De ce fait, nous définirons la matrice de codage F qui sera telle que :

$$F = \begin{pmatrix} F_\mu & & \\ F_{\mu-1} & F_\mu & \\ \vdots & F_{\mu-1} & \ddots \\ F_0 & \vdots & \ddots \\ & F_0 & \ddots \\ & & \ddots \end{pmatrix} \tag{4.43}$$

avec les sous-matrices de codage qui sont dans cette configuration de simples vecteurs de taille $n-1$:

$$F_l = \begin{pmatrix} f_{1,n}(l) & \cdots & f_{n-1,n}(l) \end{pmatrix}^T, \quad \forall l = 0, \cdots, \mu \tag{4.44}$$

Nous définirons $F_{1,1}$ la matrice qui sera composée des coefficients binaires du polynôme de feedback $f_{1,1}(D)$, telle que :

$$F_{1,1} = \begin{pmatrix} f_{1,1}(\mu) & & \\ f_{1,1}(\mu-1) & f_{1,1}(\mu) & \\ \vdots & f_{1,1}(\mu-1) & \ddots \\ f_{1,1}(0) & \vdots & \ddots \\ & f_{1,1}(0) & \ddots \\ & & \ddots \end{pmatrix} \tag{4.45}$$

Ainsi, il est possible d'écrire l'expression de la n-ième sortie du codeur en fonction des bits

d'entrée successifs sous forme matricielle :

$$\mathbf{c}_n^{'}.F_{1,1} = \mathbf{m}^{'}.F \tag{4.46}$$

Or, nous savons que le vecteur $\mathbf{m}^{'}$ est tel que :

$$\mathbf{m}^{'} = \mathbf{m}.E_g \tag{4.47}$$

où E_g est une matrice bloc-diagonale qui est composée sur sa diagonale de la matrice d'entrelacement E :

$$E_g = \begin{pmatrix} E & & \\ & \ddots & \\ & & E \end{pmatrix} \tag{4.48}$$

De ce fait, nous pouvons écrire que :

$$\mathbf{c}_n^{'}.F_{1,1} = \mathbf{m}.E_g.F \tag{4.49}$$

En notant F_e la version entrelacée de la matrice F :

$$F_e = E_g.F \tag{4.50}$$

nous obtenons au final l'équation suivante :

$$\mathbf{c}_n^{'}.F_{1,1} = \mathbf{m}.F_e \tag{4.51}$$

Nous pouvons noter que dans cette expression, les données $\mathbf{c}_n^{'}$ et $\mathbf{m}$ sont connues. L'objectif sera à partir de cette équation de réussir à identifier la matrice $F_{1,1}$ et la matrice F_e.

Le but de notre méthode de reconnaissance sera au final d'identifier l'ensemble des polynômes générateurs du codeur C_2 ainsi que l'entrelaceur. Or, la matrice de codage F est composée des coefficients des polynômes générateurs $[f_{1,n}(D), \cdots, f_{n-1,n}(D)]$ et la matrice $F_{1,1}$ des coefficients binaires du polynôme de feedback $f_{1,1}(D)$. De ce fait, nous devrons identifier les matrices de codage F et $F_{1,1}$ afin d'obtenir la matrice génératrice de C_2. Cependant d'après l'équation (4.51), nous ne pourrons pas directement obtenir la matrice F mais sa version entrelacée, F_e. Nous remarquons que la matrice F (4.43) est composée d'un même vecteur qui est simplement décalé de $(n-1)$ lignes entre chaque colonne. En revanche, sa version entrelacée, $F_e = E_g.F$, ne sera pas composée d'un unique vecteur. Cette matrice sera composée d'une sous-matrice de taille $(n-1).l_e \times \frac{l_e}{n-1}$ qui sera décalée de $\frac{le}{(n-1)}$ colonnes et de l_e lignes. Par conséquent, avant de pouvoir identifier le vecteur de la matrice F, nous devrons tout d'abord identifier la sous-matrice F_e qui est de taille $(n-1).l_e \times \frac{l_e}{n-1}$. Un fois cette sous-matrice identifiée ainsi que la matrice de feedback, $F_{1,1}$, nous mettrons en oeuvre une méthode qui permettra d'identifier conjointement les polynômes générateurs et le vecteur de permutation.

Exemple 4.54.

Suite de l'exemple 4.53.

Dans l'exemple précédent, le second codeur était un codeur RSC de paramètres $C_2(3, 2, 5)$ et de matrice génératrice $F(D)$:

$$F(D) = \frac{1}{f_{1,1}(D)} \begin{bmatrix} f_{1,1}(D) & 0 & f_{1,3}(D) \\ 0 & f_{1,1}(D) & f_{2,3}(D) \end{bmatrix}$$

Avec ses polynômes générateurs représentés sous forme polynomiale :

$$f_{1,1}(D) = [1 + D^3 + D^4], \qquad f_{1,3}(D) = [1 + D + D^4] \qquad f_{2,3} = [1 + D^2 + D^3 + D^4]$$

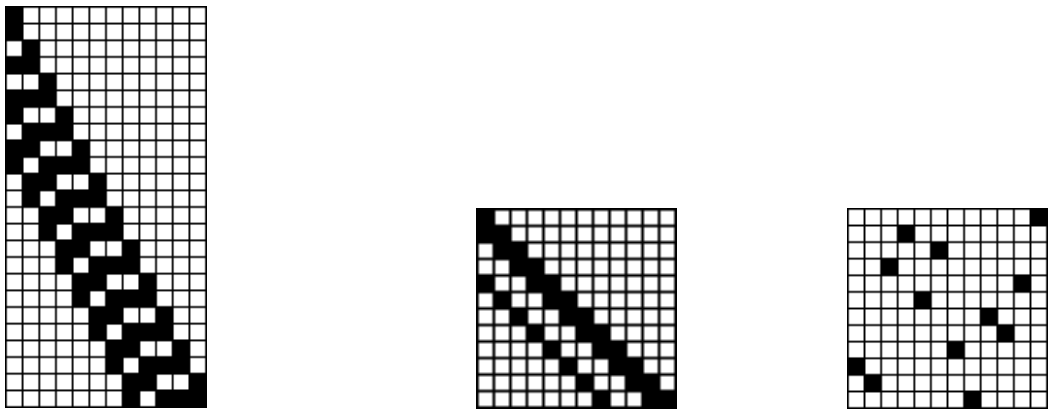
Leurs valeurs en octal puis en binaire sont :

$$\mathbf{f}_{1,1} = 23 = (1\ 0\ 0\ 1\ 1), \qquad \mathbf{f}_{1,3} = 31 = (1\ 1\ 0\ 0\ 1) \qquad \mathbf{f}_{2,3} = 27 = (1\ 0\ 1\ 1\ 1)$$

Sur les figures 4.15, nous avons représenté graphiquement les matrices F, $F_{1,1}$ et E où les carrés noirs correspondent aux bits à "1" et les blancs à ceux à "0".

Nous pouvons vérifier sur la figure 4.15(a) que la matrice F est composée du même vecteur colonne qui est simplement décalé de $(n-1)$ bits.

Sur la figure 4.16, nous avons représenté la version entrelacée de la matrice de codage, soit la matrice $F_e = E_g.F$. Nous pouvons voir sur cette figure que la première partie de F_e (la partie haute grisée) qui est de taille $(l_e \times \frac{l_e}{n-1})$ se retrouve également dans la seconde partie de F_e (partie basse grisée). Par conséquent, afin d'identifier l'ensemble des motifs de la matrice F_e, nous devrons identifier les $\frac{l_e}{n-1}$ premières colonnes de cette matrice. Soit une matrice de taille $(n-1).l_e \times \frac{l_e}{n-1}$.



(a) Matrice de codage F

(b) Matrice de feedback $F_{1,1}$

(c) Matrice de permutation E

Figure 4.15 — Les matrices de codage du turbocode[1] (cf tableau 4.1)

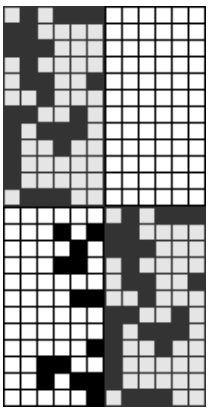


Figure 4.16 — Matrice de codage entrelacée du turbocode[1] (cf tableau 4.1)

4.4.2.2 Méthode d'identification

Le principe de notre méthode sera de construire un système homogène afin d'identifier les polynômes générateurs du codeur RSC. Nous devrons mettre l'équation (4.51) sous la forme d'un système d'équations.

Le codeur étant composé de $(n-1)$ entrées, les bits des mots d'information entrent par bloc de $(n-1)$ bits et l'entrelaceur opère par bloc de taille l_e. Par conséquent, nous réorganiserons le vecteur d'entrée $\mathbf{m}$ sous la forme d'une matrice, notée B_1, qui sera de taille $\left(\frac{L}{(n-1).l_e} \times (n-1).l_e\right)$, telle que :

$$B_1 = \begin{pmatrix} m(0) & m(1) & \cdots & m((n-1).l_e-1) \\ m((n-1).l_e) & m((n-1).l_e+1) & \cdots & m(2.(n-1).l_e) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{4.52}$$

Nous noterons $\mathbf{f}$ un vecteur composé des K sous-matrices de codage F_l $(\forall l = 0, \cdots, \mu)$ et d'un vecteur nul de taille $(n-1).(l_e-K)$, noté $\mathbf{0}_{(n-1).(l_e-K)}$:

$$\mathbf{f} = \begin{pmatrix} F_\mu & F_{\mu-1} & \cdots & F_0 & \mathbf{0}_{(n-1).(l_e-K)} \end{pmatrix}^T \tag{4.53}$$

et $\mathbf{f}_e$ la version entrelacée du vecteur $\mathbf{f}$ qui est de taille $(n-1).l_e$ et défini par :

$$\mathbf{f}_e = E_g.\mathbf{f} \tag{4.54}$$

La matrice E_g est une matrice bloc-diagonale composée de $(n-1)$ matrices E.

Dans la seconde partie de l'équation (4.51), $(\mathbf{c}_n^{'}.F_{1,1})$, l'entrelaceur n'intervient pas et la matrice $F_{1,1}$ est simplement composée du même vecteur de taille K. Nous noterons $\mathbf{f}_{1,1}$ ce vecteur :

$$\mathbf{f}_{1,1} = \begin{pmatrix} f_{1,1}(\mu) & \cdots & f_{1,1}(0) \end{pmatrix}^T \tag{4.55}$$

Nous réorganiserons le vecteur $\mathbf{c}'_n$ sous la forme d'une matrice de taille $\frac{L}{l_e} \times l_e$:

$$\begin{pmatrix} c'_n(0) & c'_n(1) & \cdots & c'_n(K-1) & \cdots & c'_n(l_e-1) \\ c'_n(l_e) & c'_n(l_e+1) & \cdots & c'_n(2.K-1) & \cdots & c'_n(2.l_e-1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{4.56}$$

L'entrelaceur n'intervenant pas, nous prendrons uniquement les K premières colonnes de cette matrice. Nous noterons B_2 cette matrice qui sera de taille $\frac{L}{l_e} \times K$:

$$B_2 = \begin{pmatrix} c'_n(0) & c'_n(1) & \cdots & c'_n(K-1) \\ c'_n(K) & c'_n(K+1) & \cdots & c'_n(2.K-1) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{4.57}$$

Une fois les matrices B_1 et B_2 construites, nous pouvons poser un système homogène qui nous permettra d'identifier les vecteurs $\mathbf{f}_e$ et $\mathbf{f}_{1,1}$:

$$\begin{bmatrix} B_1 & B_2 \end{bmatrix} . \begin{bmatrix} \mathbf{f}_e \\ \mathbf{f}_{1,1} \end{bmatrix} = 0 \tag{4.58}$$

La résolution de ce système nous permet d'identifier le polynôme de feedback ($\mathbf{f}_{1,1}$) et le vecteur $\mathbf{f}_e$ qui correspond simplement à la première colonne de la matrice F_e. Avec le polynôme $\mathbf{f}_{1,1}$, nous sommes en mesure de construire la matrice $F_{1,1}$ de l'équation (4.45). Cette matrice, nous permettra de poser un second système afin d'identifier les $\frac{l_e}{n-1}$ colonnes de la matrice F_e.

Nous noterons $\mathbf{f}_{1,1}^i$ la i-ème colonne de $F_{1,1}$ et $\mathbf{f}_e^i$ la i-ème colonne de F_e. En définissant la matrice B de taille $\frac{L}{n.l_e} \times n.l_e$:

$$B = \begin{pmatrix} m(0) & \cdots & m((n-1).l_e - 1) & c_n^{'}(0) & \cdots & c_n^{'}(l_e - 1) \\ m((n-1).l_e) & \cdots & m(2.(n-1).l_e - 1) & c_n^{'}(l_e) & \cdots & c_n^{'}(2.l_e - 1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{4.59}$$

Nous pouvons poser $\frac{l_e}{n-1}$ systèmes à résoudre :

$$B. \begin{bmatrix} \mathbf{f}_e^i \\ \mathbf{f}_{1,1}^i \end{bmatrix} = 0, \quad \forall i = 1, \cdots, \frac{l_e}{n-1} \tag{4.60}$$

Rappelons que les colonnes de la matrice $F_{1,1}$ sont connues. Par conséquent, les seules inconnues de ces systèmes sont les colonnes de la matrice F_e, soit les vecteurs $\mathbf{f}_e^i$. En utilisant les colonnes de $F_{1,1}$, nous serons en mesure, en résolvant ces systèmes, d'identifier la matrice de codage entrelacée F_e.

Une fois la matrice F_e identifiée, nous devrons mettre en place une méthode afin d'en déduire la matrice de codage F et la matrice d'entrelacement E. Nous savons que la matrice de codage est composée du même vecteur colonne. Nous noterons $\mathbf{f}$ ce vecteur qui est simplement composé des K sous-matrices de codage F_l ($\forall l = 0, \cdots, \mu$). Nous devrons donc identifier ce vecteur qui est de taille $(n-1).K$. De plus, nous avons fait l'hypothèse que les codeurs utilisés étaient optimaux. En regardant les matrices génératrices des codeurs RSC optimaux (voir annexe D), nous constatons que les polynômes de ces matrices sont tels que :

$$f_{i,j}(D) = 1 + f_{i,j}(1).D + \cdots + f_{i,j}(\mu - 1).D^{\mu-1} + 1.D^{\mu} \tag{4.61}$$

De ce fait, le vecteur $\mathbf{f}$ est tel que :

$$\mathbf{f} = \begin{pmatrix} \mathbf{1}_{n-1} & f_{1,n}(\mu-1) & \cdots & f_{n-1,n}(\mu-1) & \cdots & f_{1,n}(1) & \cdots & f_{n-1,n}(1) & \mathbf{1}_{n-1} \end{pmatrix}^T \tag{4.62}$$

où $\mathbf{1}_{n-1}$ est un vecteur composé de $(n-1)$ bits à 1.

L'entrelacement a uniquement entraîné une permutation sur les lignes de F : de ce fait le nombre de bits à "1" dans chaque colonne de F_e correspond aux nombres d'éléments à "1" dans le vecteur $\mathbf{f}$. En notant nb ce nombre de bits à "1", d'après l'équation (4.62), nous savons d'avance positionner $2.(n-1)$ bits : il restera uniquement$(n-1).(K-2)$ éléments inconnus dans les quels nous savons qu'il y a $nb - 2.(n-1)$ éléments à "1". Nous savons également que la matrice de codage F est telle que :

$$F = \begin{pmatrix} \mathbf{1}_{n-1} & & \\ ? & \mathbf{1}_{n-1} & \\ \vdots & ? & \ddots \\ \vdots & \vdots & \ddots \\ ? & \vdots & \ddots \\ \mathbf{1}_{n-1} & ? & \ddots \\ & \mathbf{1}_{n-1} & \ddots \\ & & \ddots \end{pmatrix} \tag{4.63}$$

où le symbole " ?" représente les éléments à identifier. Du fait que l'entrelaceur a uniquement entraîné une permutation sur les lignes, en réordonnant les lignes de la matrice F_e pour qu'elle soit de la même forme que F, nous obtiendrons une liste de quelques vecteurs candidats, que nous noterons

$\hat{\mathbf{f}}$. Pour l'ensemble des vecteurs $\hat{\mathbf{f}}$, nous devrons construire la matrice de codage $\hat{F}$ et rechercher une permutation telle que : $F_e = \hat{E}_g.\hat{F}$. Si nous obtenons une permutation, que nous noterons $\hat{p}$, alors avec le vecteur $\hat{\mathbf{f}}$ nous pourrons construire la matrice génératrice du code C_2. Afin de vérifier que le vecteur de permutation est correct, nous entrelacerons les données $\mathbf{m}$:

$$\hat{\mathbf{m}}' = \mathbf{m}.\hat{E}_g \tag{4.64}$$

Et nous coderons $\hat{\mathbf{m}}'$ avec le codeur identifié. Nous noterons $\hat{\mathbf{c}}'_n$ la n-ième sortie obtenue. Alors, nous validerons les paramètres identifiés en vérifiant que $\hat{\mathbf{c}}'_n$ est égale à $\mathbf{c}'_n$.

Les différentes étapes de notre méthode de reconnaissance aveugle du codeur C_2 d'un turbocode sont récapitulées dans l'algorithme 8.

Algorithme 8 : Étapes de notre algorithme de reconnaissance du code C_2 d'un turbcode

1. Identification de $F_{1,1}$ et F_e
   (a) Résoudre le système (4.58) => Identification de $\mathbf{f}_e$ et $\mathbf{f}_{1,1}$
   (b) Construire la matrice $F_{1,1}$ (4.45) avec le vecteur $\mathbf{f}_{1,1}$
   (c) Résoudre les $\frac{l_e}{n-1}$ systèmes (4.60) => Identification de F_e
2. Identification de F et p
   (a) Construire la liste des vecteurs candidats $\hat{\mathbf{f}}$
   (b) Construire les matrices $\hat{F}$ avec les vecteurs $\hat{\mathbf{f}}$
   (c) Rechercher une permutation entre F_e et $\hat{F}$
   (d) Si un vecteur de permutation a été identifié
      i. Entrelacer les données avec le vecteur de permutation identifié => $\hat{\mathbf{m}}'$
      ii. Coder les données entrelacées avec le codeur identifié => $\hat{\mathbf{c}}'_n$
      iii. Si le vecteur $\hat{\mathbf{c}}'_n$ correspond au vecteur $\mathbf{c}'_n$ alors le codeur et le vecteur de permutation identifiés sont corrects

Exemple 4.55.

Suite de l'exemple 4.54.

Les paramètres du turbocode ayant été identifiés, nous pouvons séparer les pistes de codés. Nous ferons l'hypothèse que le codeur C_1 est correctement identifié. De ce fait, en décodant les mots de code $\mathbf{c}$, nous aurons accès aux mots d'information $\mathbf{m}$. Avec ces mots d'information, les paramètres du second codeur et la taille de l'entrelaceur nous devrons identifier le second codeur et l'entrelaceur.

La première étape de reconnaissance aveugle consiste à construire le système de l'équation (4.58). En résolvant ce système, nous obtenons une unique solution non-triviale. Nous noterons d le vecteur solution qui est de taille $(n-1).l_e + K = 29$:

$$\mathbf{d} = \Big(\boxed{0\,1\,1\,0\,0\,0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0}\ \boxed{1\,1\,0\,0\,1}\Big)$$

Nous voyons que les $2.l_e$ premiers bits du vecteur $\mathbf{d}$ correspondent à la première colonne de la matrice F_e (représenté sur la figure 4.16) et les K derniers aux coefficients binaires du polynôme de feedback :

$$\mathbf{f}_{1,1} = (1\ 0\ 0\ 1\ 1)$$

Nous sommes donc en mesure de construire la matrice $F_{1,1}$ et de résoudre les $\frac{l_e}{n-1}$ systèmes de

l'équation (4.60) :

$$B.\begin{bmatrix} \mathbf{f}_e^i \\ \mathbf{f}_{1,1}^i \end{bmatrix} = 0, \quad \forall i = 1, \cdots, \frac{l_e}{n-1}$$

En résolvant ces systèmes, nous obtenons les $\frac{l_e}{n-1}$ vecteurs $\mathbf{f}_e^i$ qui sont de taille $2.(n-1).le$. Nous noterons F'_e, la matrice de codage entrelacée de taille $2.(n-1).le \times \frac{l_e}{n-1}$, qui est composée des $\frac{l_e}{n-1}$ vecteurs $\mathbf{f}_e^i$ identifiés :

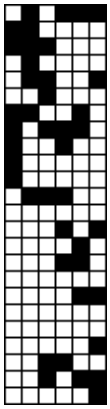


Figure 4.17 — Matrice de codage entrelacée identifiée du turbocode[1] (cf tableau 4.1)

Nous pouvons vérifier que cette matrice correspond à la première partie de la matrice de codage entrelacée F_e qui était représentée sur la figure 4.16.

Il nous faut maintenant identifier la matrice de codage et le vecteur de permutation. Nous savons que la matrice F est composée du même vecteur $\mathbf{f}$ qui est de taille $(n-1).K = 10$ et que ce vecteur est de la forme :

$$\mathbf{f} = \begin{pmatrix} 1 & 1 & f_{1,3}(3) & f_{2,3}(3) & f_{1,3}(2) & f_{2,3}(2) & f_{1,3}(1) & f_{2,3}(1) & 1 & 1 \end{pmatrix}$$

Chaque colonne de la matrice F'_e est composée de 7 bits à "1". De ce fait, nous savons que parmi les bits $(f_{1,3}(3), f_{2,3}(3), f_{1,3}(2), f_{2,3}(2), f_{1,3}(1), f_{2,3}(1))$, trois bits sont à "1" et quatre sont à "0". De plus, nous allons nous servir de la position des bits à "1" dans la première partie de la matrice F'_e, soit ses l_e premières lignes et ses $\frac{l_e}{n-1}$ premières colonnes. Nous noterons $\hat{F}_e$ cette matrice :

$$\hat{F}_e = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Nous allons tester toutes les permutations sur les lignes de cette matrice qui permettent d'obtenir

une matrice de codage $\hat{F}$ de taille $l_e \times \frac{l_e}{n-1}$ et de la forme suivante :

$$F = \begin{pmatrix} 1 & & & & & \\ 1 & & & & & \\ ? & 1 & & & & \\ ? & 1 & & & & \\ ? & ? & 1 & & & \\ ? & ? & 1 & & & \\ 1 & ? & ? & 1 & & \\ 1 & ? & ? & 1 & & \\ & 1 & ? & ? & 1 & \\ & 1 & ? & ? & 1 & \\ & & 1 & ? & ? & 1 \\ & & 1 & ? & ? & 1 \end{pmatrix}$$

où le symbole "?" représente l'emplacement des coefficients des $(n-1)$ polynômes générateurs. Nous obtenons au final huit vecteurs $\hat{\mathbf{f}}$ possibles, que nous noterons $\hat{\mathbf{f}}_i$ $(\forall i = 1, \cdots, 8)$:

$$\hat{\mathbf{f}}_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$
$$\hat{\mathbf{f}}_2 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$
$$\hat{\mathbf{f}}_3 = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$
$$\hat{\mathbf{f}}_4 = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$
$$\hat{\mathbf{f}}_5 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$
$$\hat{\mathbf{f}}_6 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$
$$\hat{\mathbf{f}}_7 = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$
$$\hat{\mathbf{f}}_8 = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Pour chacun de ces polynôme, nous allons construire une version de F (notée $\hat{F}$) et nous allons comparer cette matrice à la matrice $\hat{F}_e$, afin de rechercher un vecteur de permutation.

– Exemple en prenant le 1-er polynôme :

$$\hat{F} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad \hat{F}_e = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

La permutation faite sur la matrice de codage, entraîne une permutation sur les lignes de la matrice F. Ainsi, nous allons chercher une correspondance entre les lignes de $\hat{F}$ et de $\hat{F}_e$:

Numéro ligne de $\hat{F}$	1	2	3	4	5	6	7	8	9	10	11	12
Numéro ligne de $\hat{F}_e$	10 ou 11	10 ou 11	4	2	6	3	-	-	-	-	-	-

Nous ne retrouvons pas de permutation entre la matrice $\hat{F}$ et $\hat{F}_e$ donc le polynôme ne peut pas être celui qui a été utilisé pour le codage.

– Exemple en prenant le 4-ième polynôme :

$$\hat{F} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \qquad \hat{F}_e = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Nous recherchons une permutation :

Numéro ligne de $\hat{F}$	1	2	3	4	5	6	7	8	9	10	11	12
Numéro ligne de $\hat{F}_e$	10 ou 11	10 ou 11	2	4	3	6	12	9	8	7	1	5

Avec ce polynôme, nous obtenons deux vecteurs de permutation. D'après la matrice $\hat{F}$, les polynômes générateurs $\hat{f}_{1,3}$ et $\hat{f}_{2,3}$ sont tels que :

$$\hat{f}_{1,3} = 27$$
$$\hat{f}_{2,3} = 31$$

De ce fait, la matrice génératrice du codeur identifiée est telle que :

$$\hat{G}_{RSC} = \frac{1}{23} \begin{pmatrix} 23 & 0 & 27 \\ 0 & 23 & 31 \end{pmatrix}$$

Nous devons chercher la permutation qui permet d'obtenir les bonnes données codées avec la matrice $\hat{G}_{RSC}$. Pour chacun des deux vecteurs de permutation identifiés, nous allons entrelacer nos données $\mathbf{m}$:

$$\hat{\mathbf{m}}' = \mathbf{m}.\hat{E}_g$$

où $\hat{E}_g$ est la matrice d'entrelacement globale construite avec les vecteurs de permutation identifiés. Puis nous coderons le vecteur $\hat{\mathbf{m}}'$ avec le codeur identifié. En notant $\hat{\mathbf{c}}'_n$ la n-ième sortie obtenue, nous la comparerons à la sortie $\mathbf{c}'_n$. Si ces deux vecteurs sont identiques, alors nous en déduirons que le vecteur de permutation et la matrice génératrice du codeur C_2 sont corrects.

Dans ce cas, le bon vecteur de permutation est :

$$p' = \begin{pmatrix} 11 & 10 & 2 & 4 & 3 & 6 & 12 & 9 & 8 & 7 & 1 & 5 \end{pmatrix}$$

.

– Exemple en prenant le 5-ième polynôme :

$$\hat{F} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \qquad \hat{F}_e = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Nous recherchons une permutation :

Numéro ligne de $\hat{F}$	1	2	3	4	5	6	7	8	9	10	11	12
Numéro ligne de $\hat{F}_e$	10 ou 11	10 ou 11	4	2	6	3	9	12	7	8	5	1

Comme avec le polynôme précédent, nous trouvons deux permutations possibles. Nous allons donc procéder de la même façon. La matrice génératrice du codeur identifiée est telle que :

$$\hat{G}_{RSC} = \frac{1}{23} \begin{pmatrix} 23 & 0 & 31 \\ 0 & 23 & 27 \end{pmatrix}$$

Nous allons entrelacer les données $\mathbf{m}$ avec les deux vecteurs de permutation identifiés puis coder ces deux vecteurs avec le codeur identifié. En comparant les vecteurs $\hat{\mathbf{c}}'_n$ et $\mathbf{c}'_n$, nous obtenons au final le vecteur de permutation suivant :

$$p' = \begin{pmatrix} 10 & 11 & 4 & 2 & 6 & 3 & 9 & 12 & 7 & 8 & 5 & 1 \end{pmatrix}$$

Avec cette méthode, nous avons identifié deux matrices génératrices et deux vecteurs de permutation. Nous pouvons remarquer que les deux matrices identifiées sont identique à une permutation près sur les lignes. Dans ce cas, la permutation sur les lignes de la matrice génératrice n'est pas importante puisqu'elle sera compensée par le vecteur de permutation.

Nous venons de présenter une approche qui nous a permis d'identifier le second codeur d'un turbocode lorsqu'il était de rendement $\frac{n-1}{n}$. Cette méthode pourrait être généralisée au cas des codeur de rendement k/n. En effet, dans le cas général, après le poinçonnage des k voies systématique, nous aurons accès à $(n-k)$ voies de sorties : $\left[c'_{k+1}(D), \cdots, c'_n(D)\right]$. L'expression de ces $(n-k)$ sorties est :

$$c'_j(D) = m(D).\frac{1}{f_{1,1}(D)}.\begin{bmatrix} f_{1,j}(D) \\ \vdots \\ f_{k,j}(D) \end{bmatrix}, \quad \forall j = k+1, \cdots, n \tag{4.65}$$

Nous obtiendrions au final $(n-k)$ systèmes :

$$\mathbf{c}'_j.F_{1,1} = \mathbf{m}'.F_j \tag{4.66}$$

où F_j correspond à la matrice de codage (4.43) obtenue avec les polynômes $[f_{1,j}(D), \cdots, f_{k,j}(D)]$. Nous devrons donc appliquer l'algorithme de reconnaissance complet sur ces $(n-k)$ systèmes afin d'identifier la matrice génératrice du code C_2 et le vecteur de permutation.

4.5 Conclusions

Dans la première partie de ce chapitre, nous vous avons présenté des schémas de codage qui permettent d'améliorer de manière significative la qualité des transmissions. Cette étude sur ces schémas de codage nous a permis de mettre en avant des propriétés essentielles pour la mise en place de nos algorithme de reconnaissance. Dans la seconde partie de ce chapitre, nous avons développé deux méthodes de reconnaissance aveugle. L'une portant sur l'identification aveugle des codes concaténés en parallèle et l'autre sur les turbocodes. Pour les CCP, notre méthode nous a permis d'identifier l'ensemble de ses paramètres, soit les deux codeurs et l'entrelaceur. Dans le cas d'un turbocode, notre méthode permet également d'identifier l'ensemble de leurs paramètres à condition que le premier codeur C_1 ait été correctement identifié. Pour l'identification d'un turbocode, de par le poinçonnage des voies systématiques du second codeur, nous avons dû développer une nouvelle méthode pour identifier le second codeur, alors que les autres paramètres sont identifiés par la même méthode que pour un CCP.

Dans ce chapitre, nous avons fait l'hypothèse que la transmission était propre. La généralisation de ces méthodes au cas d'une transmission bruitée est en cours d'étude. Dans le cas d'un CCP, si nous arrivons à identifier la taille de l'entrelaceur, le paramètre n et le nombre de sorties de chaque codeur, alors en appliquant les algorithmes développés dans le chapitre 2, nous pourrions identifier les deux codes puis la matrice d'entrelacement. En revanche, dans le cas d'un turbocode, l'identification du second codeur va demander de nombreux changements afin de s'adapter au cas d'une transmission bruitée.

Conclusion

L'enjeu des technologies numériques actuelles est de garantir aux utilisateurs une qualité de transmission, en terme de vitesse et de robustesse, qui est de plus en plus grande. Afin de répondre à cette demande, les normes ou standards permettant de transmettre une information numérique sont en perpétuelle évolution. La prolifération de ces nouvelles normes peut engendrer des problèmes d'incompatibilités avec les anciennes. Le domaine de la radio cognitive offre une solution pertinente à ce problème : la conception de récepteurs intelligents. De tels récepteurs devront être capables d'identifier en aveugle les paramètres de la norme utilisée par l'émetteur. En se plaçant dans ce contexte, l'objectif de notre étude portait sur la conception de méthodes capables d'identifier en aveugle les paramètres du code correcteur d'erreurs utilisé à l'émission. Parmi l'ensemble des codes correcteurs d'erreurs, nos travaux se sont portés sur les codeurs à base de code convolutif. Afin de mettre en place des méthodes d'identification aveugle des codes convolutifs, nous avons tout d'abord réalisé une étude théorique sur les propriétés algébriques des codes convolutifs. Cette étude nous a permis d'obtenir les propriétés indispensables pour la mise en place de nos méthodes de reconnaissance.

Dans le chapitre 2, nous avons présenté deux méthodes d'identification aveugle de code convolutif. Dans ces deux méthodes, nous avons fait l'hypothèse que la seule information disponible correspondait au train binaire issu du démodulateur. La première méthode nous a permis d'identifier l'ensemble des paramètres, ainsi qu'une matrice génératrice du code convolutif utilisé à l'émission. Pour cette méthode, nous avons fait l'hypothèse que le train binaire reçu était non-entaché d'erreurs. Plus précisément, des erreurs peuvent exister mais la plage de données sur laquelle travaillent nos algorithmes ne doit pas contenir d'erreurs, ce qui signifie que le taux d'erreur doit être très faible pour que de telles plages existent. Pour simplifier, nous parlerons alors de transmission "non-bruitée". Cette première méthode fonctionne très bien lors d'une transmission non-bruitée, mais elle devient très vite inefficace lorsque la transmission est bruitée. De ce fait, nous avons conçu une seconde méthode qui permet d'identifier un code convolutif à partir d'un train binaire entaché d'erreurs. Nous avons utilisé un canal binaire symétrique afin de modéliser les erreurs introduites par le canal de transmission et la partie démodulation. Dans cette seconde méthode, nous avons également fait l'hypothèse que la trame reçue était synchronisée. En analysant les performances de détection de cet algorithme, nous avons montré qu'il était possible d'améliorer les probabilités de détecter le bon code en réalisant plusieurs itérations de cet algorithme. Pour chaque nouvelle itération, nous avons utilisé les mêmes données que nous avons judicieusement mélangées, ceci afin de réaliser virtuellement un nouveau jeu de données. Afin de montrer la pertinence de nos résultats en terme de probabilité de détection en fonction de la probabilité d'erreur du canal, nous nous sommes intéressés au taux d'erreur binaire résiduel théorique des codes convolutifs obtenus après le décodage des mots de code. En effet, il serait inutile d'essayer d'identifier en aveugle un code convolutif pour une probabilité d'erreur du canal pour laquelle ce même code ne serait jamais utilisé en pratique, car incapable de corriger les erreurs de transmission. En fixant un TEB résiduel théorique après décodage inférieur à 10^{-5}, nous avons

montré que notre algorithme de reconnaissance offrait d'excellentes performances de détection.

Afin de satisfaire une demande en débit qui est de plus en plus grande, la plupart des standards utilisent une technique simple après le codage : le poinçonnage des mots de code. Cette technique consiste simplement à ne pas transmettre tous les bits des mots de code. Dans le chapitre 3, nous avons tout d'abord réalisé une étude théorique sur cette technique de poinçonnage. Dans cette étude, nous avons montré qu'un code convolutif dont les mots de code avaient été poinçonnés pouvait être décrit par un autre code convolutif, nommé code convolutif poinçonné, construit à partir du code parent et d'un motif de poinçonnage. Nous avons vu que la reconnaissance aveugle de ce code convolutif poinçonné ne différait pas de celle d'un code convolutif classique. En revanche, dans le but d'obtenir le code parent et le motif de poinçonnage, nous avons développé un nouvel algorithme. Cet algorithme permet à partir de la seule connaissance des paramètres et de la matrice génératrice du code poinçonné d'identifier conjointement le code parent et le motif de poinçonnage. Ainsi, en utilisant l'algorithme présenté dans le chapitre 2 et ce nouvel algorithme, nous avons montré que nous étions capables, aussi bien dans le cas d'une transmission non-bruitée ou bruitée, d'identifier le code parent et le motif de poinçonnage associé. En analysant les performances de cet algorithme, nous avons montré que, comme précédemment, elles étaient nettement améliorées grâce à son processus itératif. Puis en comparant les probabilités de détection obtenues avec cet algorithme de reconnaissance au taux d'erreur binaire résiduel des codes poinçonnés, nous avons montré que nous obtenions d'excellentes performances de détection. En effet, pour un TEB résiduel théorique après décodage inférieur à 10^{-5}, les probabilités de détecter le bon code sont proche de 1.

Si le poinçonnage des mots de code permet d'augmenter le rendement d'un code et donc d'améliorer le débit d'une transmission, cette technique ne permet pas d'améliorer sa robustesse. Or, les applications actuelles nécessitent des schémas de codage de plus en plus robustes. Nous avons donc présenté dans le chapitre 4 des schémas de codage qui permettent d'améliorer de manière significative la robustesse d'une transmission. Ces schémas de codage consistent à combiner plusieurs codes. Parmi l'ensemble de ces schémas de codage existant, nous nous sommes particulièrement intéressés aux turbocodes parallèles. Dans ce chapitre, nous avons nommé *turbocode*, un turbocode dont le second codeur est de forme RSC avec ses voies systématiques poinçonnées et *CCP*, un turbocode qui est composé de deux codeurs de forme NRNSC ou RSC et où il n'y a pas de poinçonnage des voies systématiques. Nous avons développé deux méthodes de reconnaissance aveugle, l'une portant sur les CCP et l'autre sur les turbocodes. Dans ces deux méthodes, nous avons fait l'hypothèse qu'il n'y avait pas d'erreur de transmission et que le train binaire reçu était synchronisé. Pour les deux turbocodes parallèles, le principe de notre algorithme de reconnaissance reste identique : identification des paramètres du code (rendement, taille de l'entrelaceur et nombre de sorties de chaque code), identification des deux codes et identification de l'entrelaceur. La méthode permettant d'identifier les paramètres du code est identique pour un turbocode et un CCP. Une fois ces paramètres identifiés, nous sommes capables de séparer les pistes de codés. De ce fait nous connaissons les mots de code du premier codeur et ceux du second codeur. Par conséquent l'identification des deux codes du CCP est identique à l'identification d'un code convolutif classique (méthode proposée dans le chapitre 2). Puis, il reste uniquement à décoder les sorties des deux codes afin de pouvoir identifier l'entrelaceur. En revanche dans le cas d'un turbocode, en utilisant les méthodes précédentes, nous avons montré qu'il était possible d'identifier le premier codeur mais que, de par le poinçonnage des voies systématiques du second codeur, nous étions incapables, avec les méthodes existantes, d'identifier le second codeur et l'entrelaceur. Par conséquent, nous avons développé une nouvelle approche afin d'identifier conjointement le second codeur RSC et l'entrelaceur d'un turbocode. Au final, ces deux algorithmes permettent d'identifier les deux codes convolutifs et la matrice d'entrelacement d'un turbocode parallèle.

Dans l'ensemble de nos méthodes d'identification aveugle à partir d'une transmission bruitée, nous avons fait l'hypothèse que la trame reçue était synchronisée. Une perspective à court terme serait de lever cette hypothèse en adaptant la méthode de synchronisation décrite dans le cas d'une transmission non-bruitée au cas d'une transmission bruitée. D'autres perspectives à court terme sont envisagées. En effet, en étudiant la matrice de parité du code poinçonné, nous devrions être capables de trouver le lien direct entre cette matrice et celle du dual du code parent. Ce lien nous permettrait de réduire de manière significative le temps de calcul de notre algorithme de reconnaissance. Dans le cas d'un CCP, il nous reste à développer une méthode qui permettrait d'identifier en aveugle ses paramètres dans le cas d'une transmission bruitée. Le point crucial de cette méthode sera d'être capable d'identifier le nombre de sorties des deux codes afin de pouvoir séparer les pistes de codés et d'identifier les deux codes convolutifs avec les méthodes présentées dans le chapitre 2. A plus long terme, il sera nécessaire de réadapter complètement notre algorithme qui permet d'identifier conjointement le codeur RSC et l'entrelaceur d'un turbocode. De plus, il serait intéressant de se pencher sur une approche statistique afin de développer de nouvelles méthodes de reconnaissance des codeurs. Une telle approche permettrait de diminuer les probabilités de non détection en réalisant des tests, au fur et à mesure de l'identification des codes, afin de valider ou non les paramètres identifiés. Finalement, le développement de méthodes de reconnaissance aveugle avec décision souple pourrait être envisagé afin de prendre en compte les caractéristiques itératives intrinsèques des nouveaux schémas de codage et judicieusement mis à profit par les algorithmes de turbodécodage itératifs.

ANNEXE

A Les polynômes générateurs d'un codeur RSC

Cette annexe a pour objectif de démontrer le lien entre les polynômes générateurs de la matrice génératrice d'un codeur RSC et les mineurs d'ordre k de la matrice génératrice du codeur NRNSC équivalent. Avant de montrer ce lien, quelques rappels sur les déterminants et les mineurs d'une matrice sont proposés.

A.1 Déterminant et mineurs d'une matrice

Définissons une matrice A de taille $k \times k$ par :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ \vdots & \vdots & \cdots & \vdots \\ a_{k,1} & a_{k,2} & \cdots & a_{k,k} \end{pmatrix} \tag{A.1}$$

On note $A_{i,j}$ la $(k-1) \times (k-1)$ matrice obtenue en supprimant la i-ème ligne et la j-ème colonne de A.

Définition A.1. *On appelle cofacteur de l'élément $a_{i,j}$ le scalaire :*

$$Cof_{i,j} = (-1)^{i+j}. \det A_{i,j} \tag{A.2}$$

Travaillant dans le corps de Galois à deux éléments ("1" et "0"), le cofacteur de $a_{i,j}$ est :

$$Cof_{i,j} = \det A_{i,j} \tag{A.3}$$

Théorème A.1. *Le déterminant de A peut être calculé en le développant sur une colonne ou sur une ligne quelconque :*

1. *Suivant la i-ème ligne :* $\det A = \sum_{j=1}^{k} a_{i,j}.Cof_{i,j}$
2. *Suivant la j-ème colonne :* $\det A = \sum_{i=1}^{k} a_{i,j}.Cof_{i,j}$

Définition A.2. *Le déterminant d'une matrice A est nul si l'une (ou plusieurs) des conditions suivantes est (sont) remplie(s) :*

1. *A possède une colonne (ou une ligne) nulle.*
2. *A possède deux colonnes (ou deux lignes) identiques.*
3. *A possède deux colonnes (ou deux lignes) proportionnelles.*

4. A possède une colonne qui est combinaison linéaire des autres colonnes (ou une ligne qui est une combinaison linéaire des autre lignes).

Définition A.3. *Les mineurs d'ordre k d'une $m \times n$ matrice sont les déterminants des sous-matrices obtenues en supprimant $m-k$ lignes et $n-k$ colonnes.*

En notant $M_{i,j}$ les mineurs d'ordre $k-1$ de la matrice A, ces mineurs sont obtenus en supprimant la i-ème ligne et la j-ème colonne de A :

$$M_{i,j} = \det A_{i,j} = Cof_{i,j} \tag{A.4}$$

A.2 Polynômes générateurs d'un codeur RSC

La matrice génératrice d'un codeur RSC est obtenue à partir de la matrice du codeur NRNSC équivalent définie par :

$$G_{NRNSC}(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) & \cdots & g_{1,n}(D) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) & \cdots & g_{k,n}(D) \end{bmatrix} \tag{A.5}$$

On note $L(D)$ la matrice composée des $k \times k$ premiers polynômes générateurs de la matrice $G_{NRNSC}(D)$:

$$L(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) \\ \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) \end{bmatrix} \tag{A.6}$$

Le lien entre la matrice d'un codeur RSC et la matrice du codeur NRNSC équivalent est donné par l'équation ci-dessous.

$$G_{RSC}(D) = L^{-1}(D).G_{NRNSC}(D) = \frac{1}{\det L(D)}.\text{adj } L(D).G_{NRNSC}(D) = \frac{1}{\det L(D)}.G_{sys}(D). \tag{A.7}$$

avec $\det L(D)$ qui correspond au déterminant de la matrice $L(D)$, adj $L(D)$ la matrice adjointe (ou comatrice transposée) de $L(D)$ et $G_{sys}(D)$ la matrice systématique du codeur. La matrice $G_{sys}(D)$ est une matrice de taille $k \times n$ définie par :

$$G_{sys}(D) = \text{adj } L(D).G_{NRNSC}(D) = \begin{bmatrix} f_{1,1}(D) & \cdots & f_{1,k+1}(D) & \cdots & f_{1,n}(D) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ f_{k,1}(D) & \cdots & f_{k,k+1}(D) & \cdots & f_{k,n}(D) \end{bmatrix} \tag{A.8}$$

où les $f_{i,j}(D)$, $\forall i = 1, \cdots, k$ et $\forall j = 1, \cdots, n$, sont les polynômes générateurs d'un codeur RSC. La matrice adjointe de $L(D)$ est une matrice composée des cofacteurs, définie par :

$$\text{adj } L(D) = \begin{bmatrix} M_{1,1}(D) & \cdots & M_{k,1}(D) \\ \vdots & \cdots & \vdots \\ M_{1,k}(D) & \cdots & M_{k,k}(D) \end{bmatrix} \tag{A.9}$$

où $M_{l,i}(D)$ correspond au cofacteur du polynôme $g_{l,i}(D)$, soit le mineur d'ordre $k-1$ obtenu en

supprimant la l-ième ligne et la i-ème colonne de $L(D)$ tel que :

$$M_{l,i}(D) = \det \begin{pmatrix} g_{1,1}(D) & \cdots & g_{1,i-1}(D) & g_{1,i+1}(D) & \cdots & g_{1,k}(D) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ g_{l-1,1}(D) & \cdots & g_{l-1,i-1}(D) & g_{l-1,i+1}(D) & \cdots & g_{l-1,k}(D) \\ g_{l+1,1}(D) & \cdots & g_{l+1,i-1}(D) & g_{l+1,i+1}(D) & \cdots & g_{l+1,k}(D) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,i-1}(D) & g_{k,i+1}(D) & \cdots & g_{k,k}(D) \end{pmatrix} \tag{A.10}$$

D'après l'équation (A.7), la matrice génératrice d'un codeur RSC dépend du déterminant de $L(D)$ et de la matrice systématique $G_{sys}(D)$. On peut calculer le déterminant de la matrice $L(D)$ en le développant sur la k-ième colonne de $L(D)$, d'après le théorème A.1, soit :

$$\det L(D) = \sum_{i=1}^{k} g_{i,k}(D).M_{i,k}(D). \tag{A.11}$$

D'après (A.8), la matrice systématique du codeur est telle que :

$$\begin{aligned} G_{sys}(D) &= \begin{bmatrix} f_{1,1}(D) & \cdots & f_{1,k+1}(D) & \cdots & f_{1,n}(D) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ f_{k,1}(D) & \cdots & f_{k,k+1}(D) & \cdots & f_{k,n}(D) \end{bmatrix} \\ &= \begin{bmatrix} M_{1,1}(D) & \cdots & M_{k,1}(D) \\ \vdots & \cdots & \vdots \\ M_{1,k}(D) & \cdots & M_{k,k}(D) \end{bmatrix} . \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) & \cdots & g_{1,n}(D) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) & \cdots & g_{k,n}(D) \end{bmatrix} \end{aligned} \tag{A.12}$$

Le lien entre un polynôme $f_{i,j}(D)$ du RSC et les polynômes $g_{i,j}(D)$ du NRNSC équivalent peut s'exprimer par l'équation ci-dessous :

$$f_{i,j}(D) = \sum_{l=1}^{k} g_{l,j}(D).M_{l,i}(D) \tag{A.13}$$

D'après le théorème A.1, le polynôme $f_{i,j}(D)$ correspond au déterminant de la sous-matrice $L(D)$ où la i-ème colonne a été supprimée et la j-ème colonne de $L(D)$ a été ajoutée, tel que :

$$f_{i,j}(D) = \det \begin{pmatrix} g_{1,1}(D) & \cdots & g_{1,i-1}(D) & g_{1,i+1}(D) & \cdots & g_{1,k}(D) & g_{1,j}(D) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,i-1}(D) & g_{k,i+1}(D) & \cdots & g_{k,k}(D) & g_{k,j}(D) \end{pmatrix} \tag{A.14}$$

D'après la définition A.2, le déterminant d'une matrice est nul si la matrice est composée de deux colonnes (ou deux lignes) identiques. Lorsque $j \leq k$, nous aurons deux cas de figure possibles pour les polynômes $f_{i,j}(D)$:

- Si $j \neq i$:
  La colonne $\begin{bmatrix} g_{1,j} & \cdots & g_{k,j} \end{bmatrix}^T$ se trouvera deux fois dans la matrice, donc le déterminant sera nul.

$$f_{i,j}(D) = 0 \qquad \forall j = 1, \cdots, k \text{ et } i \neq j \tag{A.15}$$

- Si $j = i$:

Le déterminant de la matrice à calculer devient :

$$f_{j,j}(D) = \sum_{l=1}^{k} g_{l,j}(D).M_{l,j}(D) = \det L(D) \tag{A.16}$$

D'après ces différentes hypothèses, la matrice systématique d'un codeur RSC est une matrice de la forme :

$$G_{sys}(D) = \begin{bmatrix} f_{1,1}(D) & & & f_{1,k+1}(D) & \cdots & f_{1,n}(D) \\ & \ddots & & \vdots & \cdots & \vdots \\ & & f_{1,1}(D) & f_{k,k+1}(D) & \cdots & f_{k,n}(D) \end{bmatrix} \tag{A.17}$$

A.3 Mineurs d'ordre k de la matrice génératrice d'un codeur NRNSC

Nous allons maintenant montrer le lien entre les mineurs d'ordre k de la matrice génératrice d'un codeur NRNSC et les polynômes générateur du codeur RSC équivalent que nous venons de calculer.

Posons n matrices $G^j(D)$, $\forall j = 1, \cdots, n$, composées de la matrice $L(D)$ et de la j-ième colonne de la matrice du codeur NRNSC :

$$G^j(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) & g_{1,j}(D) \\ \vdots & \cdots & \vdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) & g_{k,j}(D) \end{bmatrix} \tag{A.18}$$

Nous allons calculer les k premiers mineurs d'ordre k de chacune de ces matrices. Nous noterons $\Delta_i^j(D)$ le i-ème mineur de la matrice $G^j(D)$ qui correspond au déterminant de la sous-matrice $G^j(D)$ lorsque la colonne i est supprimée, avec $i = 1, \cdots, k$.

$$\Delta_i^j(D) = \det \begin{pmatrix} g_{1,1}(D) & \cdots & g_{1,i-1}(D) & g_{1,i+1}(D) & \cdots & g_{1,k}(D) & g_{1,j}(D) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,i-1}(D) & g_{k,i+1}(D) & \cdots & g_{k,k}(D) & g_{k,j}(D) \end{pmatrix} \tag{A.19}$$

D'après l'équation (A.14), les mineurs d'ordre k de la matrice du codeur NRNSC correspondent aux polynômes générateurs du codeur RSC équivalent :

$$f_{i,j}(D) = \Delta_i^j(D) \tag{A.20}$$

La matrice systématique du codeur RSC, pourra également s'écrire en fonction des mineurs :

$$G_{sys}(D) = \begin{bmatrix} \Delta_1^1(D) & & & \Delta_1^{k+1}(D) & \cdots & \Delta_1^n(D) \\ & \ddots & & \vdots & \cdots & \vdots \\ & & \Delta_1^1(D) & \Delta_k^{k+1}(D) & \cdots & \Delta_k^n(D) \end{bmatrix} \tag{A.21}$$

Nous venons de démontrer dans cette annexe que les mineurs d'ordre k de la matrice génératrice d'un codeur NRNSC correspondent aux polynômes générateurs du codeur RSC équivalent.

ANNEXE B

Matrice de parité d'un code convolutif

Le but de cette annexe est de démontrer que la matrice de parité telle que nous l'avons présenté dans le chapitre 1 est une matrice génératrice du code dual.

Une $(n-k) \times n$ matrice $H(D)$ est une matrice de parité d'un code si la propriété suivante est vérifiée :

$$G(D).H^T(D) = 0 \tag{B.1}$$

Sous sa forme systématique, la matrice de parité que nous avons défini est une matrice composée des mineurs d'ordre k de la matrice génératrice du codeur NRNSC :

$$H(D) = \begin{bmatrix} \Delta_1^{k+1}(D) & \cdots & \Delta_k^{k+1}(D) & \Delta_1^1(D) & & \\ \vdots & \cdots & \vdots & & \ddots & \\ \Delta_1^n(D) & \cdots & \Delta_k^n(D) & & & \Delta_1^1(D) \end{bmatrix} \tag{B.2}$$

Nous noterons $R(D)$ la matrice résultant du produit $G(D).H^T(D)$, telle que :

$$R(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) & g_{1,k+1}(D) & \cdots & g_{1,n}(D) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) & g_{k,k+1}(D) & \cdots & g_{k,n}(D) \end{bmatrix} . \begin{bmatrix} \Delta_1^{k+1}(D) & \cdots & \Delta_1^n(D) \\ \vdots & \cdots & \vdots \\ \Delta_k^{k+1}(D) & \cdots & \Delta_k^n(D) \\ \Delta_1^1(D) & & \\ & \ddots & \\ & & \Delta_1^1(D) \end{bmatrix} \tag{B.3}$$

$R(D)$ est une $k \times (n-k)$ matrice qui sera composée d'éléments nuls si $H(D)$ est une matrice de parité du code. On note $r_{i,m}(D)$ ($\forall i = 1, \cdots, k$ et $\forall m = 1, \cdots, n-k$) un élément de $R(D)$, tel que :

$$r_{i,m}(D) = \sum_{j=1}^{k} g_{i,j}(D).\Delta_j^{k+m}(D) + g_{i,k+m}(D).\Delta_1^1(D) \tag{B.4}$$

D'après les équations (A.13) et (A.20), de l'annexe précédente, il est possible d'écrire que :

$$\Delta_j^{k+m}(D) = \sum_{l=1}^{k} g_{l,k+m}(D).M_{l,j}(D) \qquad \text{et} \qquad \Delta_1^1(D) = \sum_{l=1}^{k} g_{l,1}(D).M_{l,1}(D) \tag{B.5}$$

soit :

$$r_{i,m}(D) = \sum_{j=1}^{k} \sum_{l=1}^{k} g_{i,j}(D).g_{l,k+m}(D).M_{l,j}(D) + \sum_{l=1}^{k} g_{i,k+m}(D).g_{l,1}(D).M_{l,1}(D) \tag{B.6}$$

Afin de démontrer que les polynômes $r_{i,m}(D)$ sont nuls, nous découperons l'équation (B.6) en deux parties. Nous noterons $r^1_{i,m}(D)$ la première partie et $r^2_{i,m}(D)$ la seconde partie, telles que :

$$\begin{array}{l} r^1_{i,m}(D) = \sum_{j=1}^{k} \sum_{l=1}^{k} g_{i,j}(D).g_{l,k+m}(D).M_{l,j}(D) \\ r^2_{i,m}(D) = \sum_{l=1}^{k} g_{i,k+m}(D).g_{l,1}(D).M_{l,1}(D) \end{array} \tag{B.7}$$

Le polynôme $r^1_{i,m}(D)$ dépend des polynômes générateurs du codeur NRNSC et des cofacteurs de ces polynômes. Nous pouvons écrire que :

$$\begin{aligned} r^1_{i,m}(D) &= \sum_{j=1}^{k} \sum_{l=1}^{k} g_{i,j}(D).g_{l,k+m}(D).M_{l,j}(D) \\ &= \sum_{l=1}^{k} \sum_{j=1}^{k} g_{i,j}(D).M_{l,j}(D).g_{l,k+m}(D) \end{aligned} \tag{B.8}$$

Or, pour un l fixe, la partie $\sum_{j=1}^{k} g_{i,j}(D).M_{l,j}(D)$, correspond au déterminant d'une matrice composée de la matrice $L(D)$ en y enlevant la l-ième ligne et en ajoutant la i-ème ligne de $L(D)$, tel que :

$$\sum_{j=1}^{k} g_{i,j}(D).M_{l,j}(D) = \det \begin{pmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) \\ \vdots & \cdots & \vdots \\ g_{l-1,1}(D) & \cdots & g_{l-1,k}(D) \\ g_{l+1,1}(D) & \cdots & g_{l+1,k}(D) \\ \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) \\ g_{i,1}(D) & \cdots & g_{i,k}(D) \end{pmatrix} \tag{B.9}$$

Ce déterminant aura deux comportements différents en fonction de l.

– Si $l \neq i$: la matrice sera composée de deux lignes identiques, donc son déterminant sera nul.

$$\sum_{j=1}^{k} g_{i,j}(D).M_{l,j}(D) = 0 \qquad \forall l \neq i \tag{B.10}$$

– Si $l = i$: rappelons que la matrice $L(D)$ est telle que :

$$L(D) = \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,k}(D) \\ \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,k}(D) \end{bmatrix} \tag{B.11}$$

Le déterminant de cette matrice peut-être calculé en le développant sur la i-ème ligne :

$$\det L(D) = \sum_{j=1}^{k} g_{i,j}.M_{i,j} \tag{B.12}$$

De ce fait, nous obtenons :

$$\sum_{j=1}^{k} g_{i,j}(D).M_{l,j}(D) = \sum_{j=1}^{k} g_{i,j}(D).M_{i,j}(D) = \det L(D) \qquad \forall l = i \tag{B.13}$$

D'après ces équations, le polynôme $r^1_{i,m}(D)$ est tel que :

$$r^1_{i,m}(D) = \sum_{j=1}^{k} g_{i,j}(D).M_{i,j}(D).g_{i,k+m}(D) = \det L(D).g_{i,k+m}(D) \tag{B.14}$$

Le polynôme $r^2_{i,m}(D)$ défini à l'équation (B.7) est :

$$\begin{aligned} r^2_{i,m}(D) &= \sum_{l=1}^{k} g_{l,1}(D).M_{l,1}(D).g_{i,k+m}(D) \\ &= \det L(D).g_{i,k+m}(D) \end{aligned} \tag{B.15}$$

D'après les équations (B.14-B.15), le polynôme $r_{i,m(D)}$ est tel que :

$$r_{i,m}(D) = \det L(D).g_{i,k+m}(D) + \det L(D).g_{i,k+m}(D) = 0 \tag{B.16}$$

De ce fait, la matrice $R(D)$ qui résulte du produit $G(D).H^T(D)$ est une matrice composée de polynômes nuls. Nous pouvons en conclure que la matrice de parité $H(D)$ telle que nous l'avons défini est une matrice de parité du code.

ANNEXE C

Représentation d'état des codes convolutifs

Le but de cette annexe est de démontrer que le choix que nous avons fait afin de décrire la relation de codage des codes convolutifs sous forme de représentation d'état est correct.

C.1 Représentation d'état

En automatique, une représentation d'état permet de modéliser un système sous forme matricielle en utilisant des variables d'état. On représente un système invariant dans le temps de la manière suivante :

$$\begin{cases} \mathbf{s}(t+1) = \mathbf{s}(t).A + \mathbf{m}(t).B \\ \mathbf{c}(t) = \mathbf{s}(t).C + \mathbf{m}(t).Q \end{cases} \tag{C.1}$$

- N : nombre de variables d'état
- A : matrice de dynamique $(N \times N)$
- B : matrice de commande $(k \times N)$
- C : matrice d'observation $(N \times n)$
- Q : matrice d'action directe $(k \times n)$
- $\mathbf{s}(t)$: vecteur qui représente les N variables d'état
- $\mathbf{m}(t)$: vecteur qui représente les k entrées
- $\mathbf{c}(t)$: vecteur qui représente les n sorties

Par la suite, les matrices (A, B, C, Q) seront appelées matrices d'état. Une représentation d'état peut être transformée en une fonction de transfert du système. Cette fonction de transfert, notée $G(D)$, est une matrice de taille $k \times n$ obtenue avec l'équation suivante.

$$G(D) = B.\left(D^{-1}.\mathbf{I}_N + A\right)^{-1}.C + Q \tag{C.2}$$

Dans notre étude, cette fonction de transfert représente la matrice génératrice du codeur convolutif et le nombre de variables d'état est $N = \sum_{i=1}^{k} \mu_i$ en utilisant les notations du chapitre 1. Pour modéliser un système, les matrices d'état ne sont pas uniques et le seul critère permettant d'affirmer que les matrices d'état choisies sont correctes est que le quadruplet (A, B, C, Q) vérifie l'équation (C.2). L'objectif de cette annexe est de montrer au lecteur intéressé que la forme des matrices d'état que nous avons prise vérifie toujours l'équation (C.2). Afin de démontrer ceci, un petit rappel sur quelques propriétés matricielles est proposé ci-dessous.

C.2 Quelques définitions sur les matrices

– Matrice triangulaire :

Une matrice triangulaire supérieure (ou inférieure) est une matrice carrée dans laquelle tous les éléments situés en dessous (ou au-dessus) de la diagonale principale sont nuls :

$$L = \begin{pmatrix} l_{1,1} & l_{1,2} & \cdots & l_{1,n} \\ & l_{2,2} & \cdots & l_{2,n} \\ & & \ddots & \vdots \\ & & & l_{n,n} \end{pmatrix} \qquad L = \begin{pmatrix} l_{1,1} & & & \\ l_{2,1} & l_{2,2} & & \\ \vdots & \vdots & \ddots & \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n} \end{pmatrix} \tag{C.3}$$

Le déterminant d'une matrice triangulaire est le produit de ses éléments diagonaux :

$$\det L = l_{1,1}.l_{2,2}.\cdots.l_{n,n} \tag{C.4}$$

– Matrice bloc-diagonale :

Une matrice bloc-diagonale (également appelé matrice diagonale par blocs) est une matrice carrée qui possède des blocs de matrices carrées, notées L_i, sur sa diagonale principale :

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_n \end{pmatrix} \tag{C.5}$$

Le déterminant d'une matrice bloc-diagonale est le produit des déterminants des matrices carrées :

$$\det L = \det L_1.\det L_2.\cdots.\det L_n \tag{C.6}$$

On peut aisément montrer (définition de l'inverse d'une matrice) que l'inverse d'une matrice bloc-diagonale est telle que :

$$L^{-1} = \begin{pmatrix} L_1^{-1} & & & \\ & L_2^{-1} & & \\ & & \ddots & \\ & & & L_n^{-1} \end{pmatrix} \tag{C.7}$$

– Matrice bidiagonale :

Une matrice bidiagonale est telle que seuls les coefficients se trouvant sur la diagonale principale et la sous-diagonale sont non-nuls. De plus, une matrice bidiagonale unitaire signifie que les coefficients de la diagonale sont égaux à 1 :

$$L = \begin{pmatrix} 1 & & & & \\ b_1 & 1 & & & \\ & b_2 & 1 & & \\ & & \ddots & \ddots & \\ & & & b_{n-1} & 1 \end{pmatrix} \tag{C.8}$$

L'inverse d'une matrice bidiagonale unitaire est :

$$L^{-1} = \begin{pmatrix} 1 & & & & & \\ (-b_1) & 1 & & & & \\ (-b_1).(-b_2) & (-b_2) & 1 & & & \\ (-b_1).(-b_2).(-b_3) & (-b_2).(-b_3) & (-b_3) & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & \\ (-b_1)\cdots(-b_{n-1}) & (-b_2)\cdots(-b_{n-1}) & (-b_3)\cdots(-b_{n-1}) & \cdots & (-b_{n-1}) & 1 \end{pmatrix} \tag{C.9}$$

C.3 Matrices d'état d'un codeur NRNSC

C.3.1 Les matrices d'état d'un codeur de forme NRNSC

Au chapitre 1, nous avons représenté notre système d'état comme défini ci-dessous.

$$\begin{cases} \mathbf{s}(t+1) = [\mathbf{s}_1(t)m_1(t)\cdots\mathbf{s}_k(t)m_k(t)]\,.G_D \\ \mathbf{c}(t) = [\mathbf{s}_1(t)m_1(t)\cdots\mathbf{s}_k(t)m_k(t)]\,.G_N \end{cases} \tag{C.10}$$

La matrice G_D est une concaténation des matrices A et B et G_N des matrices C et Q. Nous allons pouvoir construire nos matrices d'état (A, B, C, Q) à partir des définitions des matrices G_D (1.56) et G_N (1.55) que nous avons vu dans le premier chapitre.

La matrice A est une matrice bloc-diagonale, elle contient k matrices A_i de taille $\mu_i \times \mu_i$ $(\forall i = 1, \cdots, k)$:

$$A = \begin{pmatrix} A_1 & & \\ & \ddots & \\ & & A_k \end{pmatrix} \quad \text{et} \quad A_i = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ 1 & & & \vdots \\ & \ddots & & \vdots \\ & & 1 & 0 \end{pmatrix} \tag{C.11}$$

La matrice B est composée de k vecteurs lignes B_i de taille $1 \times \mu_i$ $(\forall i = 1, \cdots, k)$:

$$B = \begin{pmatrix} B_1 & & \\ & \ddots & \\ & & B_k \end{pmatrix} \quad \text{et} \quad B_i = \begin{pmatrix} \mathbf{0}_{1,\mu_i-1} & 1 \end{pmatrix} \tag{C.12}$$

avec $\mathbf{0}_{1,\mu_i-1}$ qui correspond à un vecteur nul de taille $1 \times \mu_i - 1$.

Les polynômes générateurs d'un codeur NRNSC sont composés de $\mu_i + 1$ éléments binaires, tesl que :

$$\mathbf{g}_{i,j} = \begin{pmatrix} g_{i,j}(0) & g_{i,j}(1) & \cdots & g_{i,j}(\mu_i) \end{pmatrix}, \quad \forall i = 1, \cdots, k \text{ et } \forall j = 1, \cdots, n \tag{C.13}$$

La matrice C est composée des μ_i derniers éléments binaires des polynômes générateurs et la matrice

Q de leurs premiers éléments binaires :

$$C = \begin{pmatrix} g_{1,1}(\mu_1) & g_{1,2}(\mu_1) & \cdots & g_{1,n}(\mu_1) \\ \vdots & \vdots & \cdots & \vdots \\ g_{1,1}(1) & g_{1,2}(1) & \cdots & g_{1,n}(1) \\ \vdots & \vdots & \cdots & \vdots \\ g_{k,1}(\mu_k) & g_{k,2}(\mu_k) & \cdots & g_{k,n}(\mu_k) \\ \vdots & \vdots & \cdots & \vdots \\ g_{k,1}(1) & g_{k,2}(1) & \cdots & g_{k,n}(1) \end{pmatrix} \quad \text{et} \quad Q = \begin{pmatrix} g_{1,1}(0) & g_{1,2}(0) & \cdots & g_{1,n}(0) \\ \vdots & \vdots & \cdots & \vdots \\ g_{k,1}(0) & g_{k,2}(0) & \cdots & g_{k,n}(0) \end{pmatrix} \tag{C.14}$$

C.3.2 Validation de nos matrices d'état d'un codeur NRNSC

Nous savons que les matrices d'état utilisées pour modéliser un système ne sont pas uniques mais, peu importe la forme choisie, elles doivent toujours vérifier la relation suivante.

$$\begin{aligned} G_{NRNSC}(D) &= B.\left(D^{-1}.\mathbf{I}_N + A\right)^{-1}.C + Q \\ &= \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,n}(D) \\ \vdots & \cdots & \vdots \\ g_{k,1}(D) & \cdots & g_{k,n}(D) \end{bmatrix} \end{aligned} \tag{C.15}$$

Par la suite nous poserons :

$$X(D) = \left(D^{-1}.\mathbf{I}_N + A\right) \tag{C.16}$$

Afin de démontrer l'égalité (C.15), nous allons calculer étape par étape la relation $B.\left(D^{-1}.\mathbf{I}_N + A\right)^{-1}.C + Q$ afin de vérifier que, après ces calculs, nous obtenons la matrice génératrice du code.

Calcul de $X(D)^{-1}$:

La matrice A est une matrice bloc-diagonale. Il est donc possible, pour calculer son inverse, de calculer l'inverse des sous-matrices et d'obtenir l'inverse de $X(D)$ d'après l'équation (C.7). Posons $X_i(D) = \left(D^{-1}.\mathbf{I}_{\mu_i} + A_i\right)$. Cette matrice est de taille $\mu_i \times \mu_i$ et elle peut être décomposée sous la forme d'un produit de deux matrices :

$$X_i(D) = \begin{bmatrix} D^{-1} & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & D^{-1} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ D & \ddots & & \\ & \ddots & \ddots & \\ & & D & 1 \end{bmatrix} . \begin{bmatrix} D^{-1} & & & \\ & D^{-1} & & \\ & & \ddots & \\ & & & D^{-1} \end{bmatrix} \tag{C.17}$$

L'inverse du produit de deux matrices est égale au produit des deux inverses $(A.B)^{-1} = (B)^{-1}.(A)^{-1}$. La première matrice est une matrice bidiagonale unitaire, son inverse est obtenue à l'aide de l'équation (C.9) et nous obtenons l'inverse de $X_i(D)$:

$$X_i(D)^{-1} = \begin{bmatrix} D & & & \\ & \ddots & & \\ & & \ddots & \\ & & & D \end{bmatrix} . \begin{bmatrix} 1 & & & \\ D & \ddots & & \\ \vdots & \ddots & \ddots & \\ D^{\mu_i - 1} & \cdots & D & 1 \end{bmatrix} = \begin{bmatrix} D & & & \\ D^2 & \ddots & & \\ \vdots & \ddots & \ddots & \\ D^{\mu_i} & \cdots & D^2 & D \end{bmatrix} \tag{C.18}$$

L'inverse de la matrice bloc-diagonale $X(D)$ est également une matrice bloc-diagonale qui sera composée de l'inverse des blocs $X_i(D)$:

$$X(D)^{-1} = \begin{bmatrix} X_1(D)^{-1} & & & \\ & X_2(D)^{-1} & & \\ & & \ddots & \\ & & & X_\mu(D)^{-1} \end{bmatrix} \tag{C.19}$$

Calcul de $B.X(D)^{-1}$:

D'après la matrice inverse de $X(D)$ et la matrice B définie à l'équation (C.12), nous obtenons :

$$B.X(D)^{-1} = \begin{bmatrix} D^{\mu_1} & \cdots & D^2 & D & & & & \\ & & & & \ddots & & & \\ & & & & & D^{\mu_k} & \cdots & D^2 & D \end{bmatrix} \tag{C.20}$$

Calcul de $B.X(D)^{-1}.C$:

Avec le résultat que nous venons d'obtenir et la matrice C définie à l'équation (C.14), nous trouvons :

$$B.X(D)^{-1}.C = \begin{bmatrix} \sum_{m=1}^{\mu_1} g_{1,1}(m).D^m & \cdots & \sum_{m=1}^{\mu_1} g_{1,n}(m).D^m \\ \vdots & & \vdots \\ \sum_{m=1}^{\mu_1} g_{k,1}(m).D^m & \cdots & \sum_{m=1}^{\mu_1} g_{k,n}(m).D^m \end{bmatrix} \tag{C.21}$$

Calcul de $B.X(D)^{-1}.C + Q$:

La matrice obtenue au finale est telle que :

$$B.X(D)^{-1}.C + Q = \begin{bmatrix} \sum_{m=0}^{\mu_1} g_{1,1}(m).D^m & \cdots & \sum_{m=0}^{\mu_1} g_{1,n}(m).D^m \\ \vdots & & \vdots \\ \sum_{m=0}^{\mu_k} g_{k,1}(m).D^m & \cdots & \sum_{m=0}^{\mu_k} g_{k,n}(m).D^m \end{bmatrix} \tag{C.22}$$

Nous pouvons vérifier que les matrices d'état que nous avons définies respectent le critère (C.2) puisque :

$$\boxed{G_{NRNSC}(D) = B.X(D)^{-1}.C + Q \;=\; \begin{bmatrix} g_{1,1}(D) & \cdots & g_{1,n}(D) \\ \vdots & & \vdots \\ g_{k,1}(D) & \cdots & g_{k,n}(D) \end{bmatrix}} \tag{C.23}$$

De ce fait, les matrices d'état que nous avons choisies permettent de modéliser notre système.

C.4 Matrices d'état d'un codeur RSC

C.4.1 Les matrices d'état d'un codeur de forme RSC

Dans le cas d'un codeur de forme RSC, les matrices d'état que nous avons définies dans le chapitre 1 représentent également une concaténation des matrices (A, B) pour G_D et (C, Q) pour G_N. Les polynômes générateurs d'un codeur RSC sont notés $f_{i,j}(D)$ ($\forall i = 1, \cdots, k$ et $\forall j = 1, \cdots, n$) :

$$f_{i,j}(D) = f_{i,j}(0) + f_{i,j}(1).D + \cdots + f_{i,j}(\mu_i).D^{\mu_i} \tag{C.24}$$

De plus, nous avons vu que les degrés des polynômes générateurs d'un codeur RSC étaient tous identiques ($\mu_1 = \mu_2 = \cdots = \mu_k$). Nous noterons μ le degré des polynômes du codeur RSC. Le polynôme $f_{1,1}(D)$ est appelé polynôme de feedback et d'après les propriétés des codes RSC, $f_{1,1}(0) = 1$.

La matrice A est une matrice bloc-diagonale, elle possède k matrices A_i de taille $\mu \times \mu$:

$$A = \begin{pmatrix} A_1 & & \\ & \ddots & \\ & & A_k \end{pmatrix} \quad \text{et} \quad A_i = \begin{pmatrix} 0 & \cdots & 0 & f_{1,1}(\mu) \\ 1 & & & \vdots \\ & \ddots & & \vdots \\ & & 1 & f_{1,1}(1) \end{pmatrix} \tag{C.25}$$

La matrice B est une matrice composée de k vecteurs de taille μ telle que :

$$B = \begin{pmatrix} \mathbf{O}_{1,\mu-1} & f_{1,1}(0) & & \\ & & \ddots & \\ & & & \mathbf{O}_{1,\mu-1} \quad f_{1,1}(0) \end{pmatrix} = \begin{pmatrix} \mathbf{O}_{1,\mu-1} & 1 & & \\ & & \ddots & \\ & & & \mathbf{O}_{1,\mu-1} \quad 1 \end{pmatrix} \tag{C.26}$$

avec $\mathbf{O}_{1,\mu-1}$ qui correspond à un vecteur nul de taille $\mu - 1$.

La matrice C est une matrice de taille $k.\mu \times n$, elle est composée d'une matrice nulle de taille $k.\mu \times k$ et d'une matrice de taille $k.\mu \times n - k$ composée des μ derniers coefficients des polynômes $q_{i,j}(D)$ ($\forall i = 1, \cdots, k$ et $\forall j = k+1, \cdots, n$) :

$$C = \begin{pmatrix} 0 & \cdots & 0 & q_{1,k+1}(\mu) & \cdots & q_{1,n}(\mu) \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & q_{1,k+1}(1) & \cdots & q_{1,n}(1) \\ & \vdots & & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & q_{k,k+1}(\mu) & \cdots & q_{k,n}(\mu) \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ \underbrace{0 \quad \cdots \quad 0}_{k} & & & \underbrace{q_{k,k+1}(1) \quad \cdots \quad q_{k,n}(1)}_{n-k} & & \end{pmatrix} \tag{C.27}$$

Les éléments $q_{i,j}(l)$ sont tels que :

$$\begin{cases} \text{Si } f_{i,j}(0) = 1 \text{ alors } q_{i,j}(l) = f_{i,j}(l) + f_{1,1}(l) \\ \text{Si } f_{i,j}(0) = 0 \text{ alors } q_{i,j}(l) = f_{i,j}(l) \end{cases} \tag{C.28}$$

La matrice Q est une matrice composée de la matrice identité de taille k et des premiers

coefficients des polynômes $f_{i,j}(D)$:

$$Q = \begin{pmatrix} 1 & & & f_{1,k+1}(0) & \cdots & f_{1,n}(0) \\ & \ddots & & \vdots & \cdots & \vdots \\ & & 1 & f_{k,k+1}(0) & \cdots & f_{k,n}(0) \end{pmatrix} \tag{C.29}$$

C.4.2 Validation de nos matrices d'état d'un codeur RSC

Les matrices d'état que nous avons définies précédemment doivent satisfaire l'égalité suivante :

$$\begin{aligned} G_{RSC}(D) &= B.\left(D^{-1}.\mathbf{I}_N + A\right)^{-1}.C + Q \\ &= \begin{bmatrix} 1 & & & \frac{f_{1,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{1,n}(D)}{f_{1,1}(D)} \\ & \ddots & & \vdots & \cdots & \vdots \\ & & 1 & \frac{f_{k,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{k,n}(D)}{f_{1,1}(D)} \end{bmatrix} \end{aligned} \tag{C.30}$$

avec $N = k.\mu$. Comme précédemment, nous noterons :

$$X(D) = \left(D^{-1}.\mathbf{I}_N + A\right) \tag{C.31}$$

Nous allons également calculer étape par étape la relation $B.\left(D^{-1}.\mathbf{I}_N + A\right)^{-1}.C+Q$ afin de vérifier que l'on obtient la matrice génératrice du code.

Calcul de $X(D)^{-1}$:

La matrice A d'un codeur RSC est également une matrice bloc-diagonale, nous allons chercher l'inverse de $X_i(D) = \left(D^{-1}.\mathbf{I}_\mu + A_i\right)$ pour obtenir l'inverse de $X(D)$.

$$X_i(D) = \begin{bmatrix} D^{-1} & & & f_{1,1}(\mu) \\ 1 & \ddots & & \vdots \\ & \ddots & \ddots & f_{1,1}(2) \\ & & 1 & D^{-1} + f_{1,1}(1) \end{bmatrix} \tag{C.32}$$

L'inverse d'une matrice peut-être obtenue par la formule suivante :

$$X_i(D)^{-1} = \frac{1}{\det X_i(D)}.\text{adj } X_i(D) \tag{C.33}$$

ou adj $X_i(D)$ correspond à la matrice adjointe (ou comatrice transposée) de $X_i(D)$. Nous avons explicité en annexe A que la matrice adjointe était composée des déterminants des sous-matrices obtenues en enlevant une ligne et une colonne et que le déterminant d'une matrice carrée pouvait être obtenu en le développant sur une ligne (ou une colonne).

– ***Calcul de*** $\det X_i(D)$ ***:***

En posant $Y_{\mu,\mu}(D)$, la matrice obtenue en supprimant la colonne μ et la ligne μ de $X_i(D)$ et $Y_{\mu,\mu-1}(D)$, la matrice obtenue en supprimant la ligne μ et la colonne $\mu-1$ de $X_i(D)$:

$$Y_{\mu,\mu}(D) = \begin{bmatrix} D^{-1} & & & \\ 1 & D^{-1} & & \\ & \ddots & \ddots & \\ & & 1 & D^{-1} \end{bmatrix} \quad Y_{\mu,\mu-1}(D) = \begin{bmatrix} D^{-1} & & & f_{1,1}(\mu) \\ 1 & D^{-1} & & f_{1,1}(\mu-1) \\ & \ddots & \ddots & \vdots \\ & & D^{-1} & f_{1,1}(3) \\ & & 1 & f_{1,1}(2) \end{bmatrix} \tag{C.34}$$

Le déterminant de $X_i(D)$ est :

$$\det X_i(D) \quad = \left(D^{-1} + f_{1,1}(1)\right) . \det Y_{\mu,\mu}(D) + 1. \det Y_{\mu,\mu-1}(D) \tag{C.35}$$

avec les matrices $Y_{\mu,\mu}(D)$ et $Y_{\mu,\mu-1}(D)$ qui sont de taille $(\mu-1)\times(\mu-1)$.

La matrice $Y_{\mu,\mu}(D)$ est une matrice triangulaire, donc son déterminant est le produit de ses coefficients diagonaux :

$$\det Y_{\mu,\mu}(D) = D^{-(\mu-1)} \tag{C.36}$$

Le déterminant de la sous-matrice $Y_{\mu,\mu-1}(D)$ peut être développé sur la dernière colonne :

$$\det Y_{\mu,\mu-1}(D) = \sum_{j=1}^{\mu-1} f_{1,1}(\mu-(j-1)).M_{j,\mu-1}(D) \tag{C.37}$$

avec $M_{j,\mu-1}(D)$ qui correspond au déterminant de la sous-matrice (de taille $(\mu-2)\times(\mu-2)$) obtenue en supprimant la j-ème ligne et la $(\mu-1)$-ième colonne de $Y_{\mu,\mu-1}(D)$. Les déterminants $M_{j,\mu-1}(D)$ à calculer sont de la forme :

$$M_{j,\mu-1}(D) = \det \begin{pmatrix} \boxed{\begin{matrix} D^{-1} & & & \textcircled{1} \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & D^{-1} \end{matrix}} & \\ & \boxed{\begin{matrix} 1 & D^{-1} & & \textcircled{2} \\ & \ddots & \ddots & \\ & & \ddots & D^{-1} \\ & & & 1 \end{matrix}} \end{pmatrix} \tag{C.38}$$

avec le bloc ① qui est de taille $(j-1)\times(j-1)$ et le bloc ② de taille $(\mu-1-j)\times(\mu-1-j)$. Le déterminant $M_{j,\mu-1}(D)$ est le produit des deux matrices blocs ① et ②, qui sont toutes les deux des matrices triangulaires :

$$M_{j,\mu-1}(D) = D^{-(j-1)} \tag{C.39}$$

D'après l'équation (C.37), le déterminant de la sous-matrices $Y_{\mu,\mu-1}(D)$ est tel que :

$$\det Y_{\mu,\mu-1}(D) = \sum_{j=1}^{\mu-1} f_{1,1}(\mu-(j-1)).D^{-(j-1)} \tag{C.40}$$

En prenant le déterminant de la sous-matrice $Y_{\mu,\mu}(D)$ et celui de la sous-matrice $Y_{\mu,\mu-1}(D)$, nous obtenons le déterminant de $X_i(D)$:

$$\begin{aligned}\det X_i(D) &= D^{-\mu} + \sum_{j=1}^{\mu} f_{1,1}(\mu-(j-1)).D^{-(j-1)} \\ &= D^{-\mu} + \sum_{j=1}^{\mu} f_{1,1}(j).D^{-(\mu-j)} \\ &= \frac{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^{j}}{D^{\mu}}\end{aligned} \tag{C.41}$$

– ***Calcul de*** adj $X_i(D)$:

Afin d'obtenir la matrice inverse de $X_i(D)$, nous devons obtenir sa matrice adjointe. Notons $Cof_{l,j}(D)$ le mineur obtenu en supprimant la l-ième ligne et la j-ème colonne de $X_i(D)$. La matrice ajointe de $X_i(D)$ est telle que :

$$\text{adj } X_i(D) = \begin{bmatrix} Cof_{1,1}(D) & \cdots & Cof_{\mu,1}(D) \\ \vdots & \cdots & \vdots \\ Cof_{1,\mu}(D) & \cdots & Cof_{\mu,\mu}(D) \end{bmatrix} \tag{C.42}$$

Or, l'étape suivante consiste à calculer :

$$\begin{aligned} B_i.\frac{1}{\det X_i(D)}.\text{adj } X_i(D) &= \frac{1}{\det X_i(D)} \begin{bmatrix} \mathbf{0}_{\mu-1} & 1 \end{bmatrix}.\text{adj } X_i(D) \\ &= \frac{1}{\det X_i(D)}. \begin{bmatrix} Cof_{1,\mu}(D) & \cdots & Cof_{\mu,\mu}(D) \end{bmatrix} \end{aligned} \tag{C.43}$$

Il n'est pas nécessaire de calculer tous les mineurs d'ordre $\mu-1$ de $X_i(D)$ mais seulement ceux obtenus en supprimant sa dernière colonne, $Cof_{j,\mu}(D)$ avec $j = 1, \cdots, \mu$. En supprimant la dernière colonne, nous obtenons une matrice de taille $\mu \times \mu - 1$:

$$\begin{bmatrix} D^{-1} & & & \\ 1 & D^{-1} & & \\ & \ddots & \ddots & \\ & & \ddots & D^{-1} \\ & & & 1 \end{bmatrix} \tag{C.44}$$

Afin d'obtenir les mineurs, il suffit de supprimer l'une des lignes de la matrice ci-dessus et de calculer son déterminant. Les matrices obtenues seront de la même forme qu'à l'équation (**??**) et les mineurs seront :

$$Cof_{j,\mu}(D) = D^{-(j-1)} \tag{C.45}$$

La matrice $X_i(D)$ inverse sera de la forme :

$$X_i(D)^{-1} = \frac{D^{\mu}}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^{j}}. \begin{bmatrix} Cof_{1,1}(D) & Cof_{2,1}(D) & \cdots & Cof_{\mu,1}(D) \\ \vdots & \vdots & \cdots & \vdots \\ Cof_{1,\mu-1}(D) & Cof_{2,\mu-1}(D) & \cdots & Cof_{\mu,\mu-1}(D) \\ 1 & D^{-1} & \cdots & D^{-(\mu-1)} \end{bmatrix} \tag{C.46}$$

La matrice bloc-diagonale $X(D)$ est composée de k matrices $X_i(D)$ identiques, donc son inverse sera :

$$X(D)^{-1} = \begin{bmatrix} X_i(D)^{-1} & & \\ & \ddots & \\ & & X_i(D)^{-1} \end{bmatrix} = \frac{1}{\det X_i(D)} \begin{bmatrix} \text{adj } X_i(D) & & \\ & \ddots & \\ & & \text{adj } X_i(D) \end{bmatrix} \quad \text{(C.47)}$$

Calcul de $B.X(D)^{-1}$:

D'après la matrice B que nous avons défini à l'équation (C.26) et la matrice inverse de $X(D)$, nous obtenons :

$$\begin{aligned} B.X(D)^{-1} &= \frac{D^\mu}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^j}. \begin{bmatrix} 1 & D^{-1} & \cdots & D^{-(\mu-1)} & & & \\ & & & & \ddots & & \\ & & & & & 1 & D^{-1} & \cdots & D^{-(\mu-1)} \end{bmatrix} \\ &= \frac{1}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^j}. \begin{bmatrix} D^\mu & D^{\mu-1} & \cdots & D & & & \\ & & & & \ddots & & \\ & & & & & D^\mu & D^{\mu-1} & \cdots & D \end{bmatrix} \end{aligned} \quad \text{(C.48)}$$

Calcul de $B.X(D)^{-1}.C$:

La matrice C définie à l'équation (C.27) nous permet d'obtenir :

$$B.X(D)^{-1}.C = \frac{1}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^j}. \begin{pmatrix} 0 & \cdots & 0 & \sum_{j=1}^{\mu} q_{1,k+1}(j).D^j & \cdots & \sum_{j=1}^{\mu} q_{1,n}(j).D^j \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \sum_{j=1}^{\mu} q_{k,k+1}(j).D^j & \cdots & \sum_{j=1}^{\mu} q_{k,n}(j).D^j \end{pmatrix} \quad \text{(C.49)}$$

avec le bloc de zéros qui est de taille $k \times k$.

Calcul de $B.X(D)^{-1}.C + Q$:

$$B.X(D)^{-1}.C + Q = \begin{pmatrix} 1 & & \frac{\sum_{j=1}^{\mu} q_{1,k+1}(j).D^j}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^j} + f_{1,k+1}(0) & \cdots & \frac{\sum_{j=1}^{\mu} q_{1,n}(j).D^j}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^j} + f_{1,n}(0) \\ & \ddots & \vdots & \cdots & \vdots \\ & 1 & \frac{\sum_{j=1}^{\mu} q_{k,k+1}(j).D^j}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^j} + f_{k,k+1}(0) & \cdots & \frac{\sum_{j=1}^{\mu} q_{k,n}(j).D^j}{1+\sum_{j=1}^{\mu} f_{1,1}(j).D^j} + f_{k,n}(0) \end{pmatrix} \quad \text{(C.50)}$$

Faisons le calcul pour l'un des polynômes que nous nommerons $a_{i,l}(D)$:

$$a_{i,l}(D) = \frac{\sum_{j=1}^{\mu} q_{i,l}(j).D^j}{1 + \sum_{j=1}^{\mu} f_{1,1}(j).D^j} + f_{i,l}(0) \tag{C.51}$$

– Si $f_{i,l}(0) = 1$ alors $q_{i,l}(j) = f_{i,l}(j) + f_{1,1}(j)$:

$$\begin{aligned} a_{i,l}(D) &= \frac{\sum_{j=1}^{\mu} \left(f_{i,l}(j) + f_{1,1}(j)\right).D^j}{1 + \sum_{j=1}^{\mu} f_{1,1}(j).D^j} + 1 \\ &= \frac{1 + \sum_{j=1}^{\mu} f_{i,l}(j).D^j}{1 + \sum_{j=1}^{\mu} f_{1,1}(j).D^j} = \frac{f_{i,l}(D)}{f_{1,1}(D)} \end{aligned} \tag{C.52}$$

– Si $f_{i,l}(0) = 0$ alors $q_{i,l}(j) = f_{i,l}(j)$:

$$a_{i,l}(D) = \frac{\sum_{j=1}^{\mu} f_{i,l}(j).D^j}{1 + \sum_{j=1}^{\mu} f_{1,1}(j).D^j} + 0 = \frac{f_{i,l}(D)}{f_{1,1}(D)} \tag{C.53}$$

Nous obtenons au final :

$$B.X(D)^{-1}.C + Q = \begin{bmatrix} 1 & & \frac{f_{1,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{1,n}(D)}{f_{1,1}(D)} \\ & \ddots & \vdots & \cdots & \vdots \\ & 1 & \frac{f_{k,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{k,n}(D)}{f_{1,1}(D)} \end{bmatrix} \tag{C.54}$$

Nous pouvons vérifier que les matrices d'état que nous avons définies respectent le critère (C.2) puisque :

$$\boxed{G_{RSC}(D) = B.X(D).C + Q \;\; = \begin{bmatrix} 1 & & \frac{f_{1,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{1,n}(D)}{f_{1,1}(D)} \\ & \ddots & \vdots & \cdots & \vdots \\ & 1 & \frac{f_{k,k+1}(D)}{f_{1,1}(D)} & \cdots & \frac{f_{k,n}(D)}{f_{1,1}(D)} \end{bmatrix}} \tag{C.55}$$

De ce fait, les matrices d'état que nous avons choisies permettent de modéliser notre système.

ANNEXE D

Tables des codes convolutifs optimaux

Cette annexe présente une liste non-exhaustive de codes convolutifs optimaux.

– Table de codes convolutifs optimaux de rendement $\frac{1}{n}$:

n	K	Matrice génératrice	d_{libre}
2	2	(1 3)	3
	3	(5 7)	5
	4	(13 17)	6
	5	(23 35)	7
	6	(53 75)	8
	7	(133 171)	10
	9	(561 753)	12
	11	(2335 3661)	14
3	2	(1 3 3)	5
	3	(5 7 7)	8
	4	(13 15 17)	10
	5	(25 33 37)	12
	6	(47 53 75)	13
	7	(133 145 175)	15
	8	(225 331 367)	16
	9	(557 663 711)	18
	10	(1117 1365 1633)	20
	11	(2353 2671 3175)	22

n	K	Matrice génératrice	d_{libre}
4	2	(1 3 3 3)	7
	3	(5 7 7 7)	10
	4	(13 15 15 17)	13
	5	(25 27 33 37)	16
	6	(53 67 71 75)	18
	7	(133 135 147 163)	18
	8	(235 275 313 357)	22
	9	(463 535 733 745)	24
	9	(765 671 513 473)	24
	10	(1117 1365 1633 1653)	27
6	5	(33 25 37 33 25 37)	24

Les tableaux ci-dessous présentent différents codes optimaux de rendement k/n avec $k > 1$. Pour chaque rendement, les paramètres du codeur NRNSC et de son codeur équivalent RSC ainsi que leur distance libre sont donnés.

– Table de codes convolutifs optimaux de rendement $\frac{2}{3}$:

Codeur NRNSC		Codeur RSC			d_{libre}
K	Matrice génératrice	K	Matrice systématique	Feedback	
3	$\begin{pmatrix}1 & 2 & 3\\4 & 1 & 7\end{pmatrix}$	4	$\begin{pmatrix}11 & 0 & 15\\0 & 11 & 13\end{pmatrix}$	11	4
3	$\begin{pmatrix}7 & 4 & 1\\2 & 5 & 7\end{pmatrix}$	5	$\begin{pmatrix}23 & 0 & 31\\0 & 23 & 27\end{pmatrix}$	23	5
4	$\begin{pmatrix}3 & 6 & 7\\14 & 1 & 17\end{pmatrix}$	6	$\begin{pmatrix}53 & 0 & 45\\0 & 53 & 65\end{pmatrix}$	53	6
4	$\begin{pmatrix}13 & 6 & 13\\6 & 13 & 17\end{pmatrix}$	7	$\begin{pmatrix}121 & 0 & 147\\0 & 121 & 123\end{pmatrix}$	121	7
5	$\begin{pmatrix}3 & 6 & 15\\34 & 31 & 17\end{pmatrix}$	8	$\begin{pmatrix}143 & 0 & 227\\0 & 143 & 235\end{pmatrix}$	143	5
6	$\begin{pmatrix}25 & 30 & 17\\50 & 7 & 65\end{pmatrix}$	10	$\begin{pmatrix}1653 & 0 & 1325\\0 & 1653 & 1051\end{pmatrix}$	1653	9
6	$\begin{pmatrix}63 & 54 & 31\\26 & 53 & 43\end{pmatrix}$	11	$\begin{pmatrix}2605 & 0 & 3067\\0 & 2605 & 3763\end{pmatrix}$	2605	10

– Table de codes convolutifs optimaux de rendement $\frac{2}{4}$:

Codeur NRNSC		Codeur RSC			d_{libre}
K	Matrice génératrice	K	Matrice systématique	Feedback	
2	$\begin{pmatrix}3 & 1 & 2 & 1\\1 & 2 & 1 & 3\end{pmatrix}$	3	$\begin{pmatrix}7 & 0 & 5 & 1\\0 & 7 & 1 & 4\end{pmatrix}$	7	5

– Table de codes convolutifs optimaux de rendement $\frac{3}{4}$:

Codeur NRNSC		Codeur RSC			d_{libre}
K	Matrice génératrice	K	Matrice systématique	Feedback	
3	$\begin{pmatrix}3 & 2 & 2 & 3\\4 & 3 & 0 & 7\\0 & 2 & 5 & 5\end{pmatrix}$	6	$\begin{pmatrix}51 & 0 & 0 & 45\\0 & 51 & 0 & 71\\0 & 0 & 51 & 63\end{pmatrix}$	51	5
3	$\begin{pmatrix}3 & 4 & 0 & 7\\6 & 1 & 4 & 3\\2 & 6 & 7 & 1\end{pmatrix}$	7	$\begin{pmatrix}111 & 0 & 0 & 151\\0 & 111 & 0 & 121\\0 & 0 & 111 & 177\end{pmatrix}$	111	6
4	$\begin{pmatrix}7 & 6 & 2 & 1\\14 & 5 & 0 & 15\\10 & 4 & 17 & 1\end{pmatrix}$	9	$\begin{pmatrix}461 & 0 & 0 & 753\\0 & 461 & 0 & 575\\0 & 0 & 461 & 727\end{pmatrix}$	461	7
4	$\begin{pmatrix}1 & 14 & 16 & 3\\10 & 13 & 2 & 7\\16 & 0 & 3 & 13\end{pmatrix}$	10	$\begin{pmatrix}1161 & 0 & 0 & 1425\\0 & 1161 & 0 & 1257\\0 & 0 & 1161 & 1273\end{pmatrix}$	1161	8

ANNEXE

E Tables des codes convolutifs poinçonnés

Cette annexe présente une liste de codes poinçonnés. La première partie est consacrée au poinçonnage d'un code parent de rendement $\frac{1}{2}$ vers des codes poinçonnés de différents rendements et la deuxième partie pour des codeurs de rendement $\frac{1}{3}$.

– Table de codes poinçonnés de rendement $\frac{2}{3}$ à partir d'un code parent de rendement $\frac{1}{2}$ ($M = 2$) :

C

Code parent		Code poinçonné			Code parent		Code poinçonné		
K	G	P	K_p	d_{libre}	K	G	P	K_p	d_{libre}
3	(5 7)	$\begin{pmatrix}1 & 0\\1 & 1\end{pmatrix}$	2	3	15	[55667 63121]	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	8	10
4	(15 17)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	3	4	16	(111653 145665)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	9	10
5	(23 35)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	3	4	17	(34721 24277)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	9	12
6	(53 75)	$\begin{pmatrix}1 & 0\\1 & 1\end{pmatrix}$	4	6	18	(506477 673711)	$\begin{pmatrix}1 & 0\\1 & 1\end{pmatrix}$	9	12
7	(133 171)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	4	6	19	(1352755 1771563)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	10	12
8	(247 371)	$\begin{pmatrix}1 & 0\\1 & 1\end{pmatrix}$	5	7	20	(2451321 3546713)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	11	12
9	(561 753)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	5	7	20	(2142513 3276177)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	11	13
10	(1167 1545)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	6	7	21	(6567413 5322305)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	11	12
11	(2335 3661)	$\begin{pmatrix}1 & 0\\1 & 1\end{pmatrix}$	6	8	22	(15724153 12076311)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	12	13
12	(4335 5723)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	7	9	23	(33455341 24247063)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	12	14
13	(10533 17661)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	7	9	24	(55076157 75501351)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	13	15
14	(21675 27123)	$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}$	8	10					

– Table de codes poinçonnés de rendement $\frac{3}{4}$ à partir d'un code parent de rendement $\frac{1}{2}$ ($M = 3$) :

Code parent		Code poinçonné		
K	G	P	K_p	d_{libre}
3	(5 7)	$\begin{pmatrix}1&0&1\\1&1&0\end{pmatrix}$	2	3
4	(15 17)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	2	4
5	(23 35)	$\begin{pmatrix}1&0&1\\1&1&0\end{pmatrix}$	3	4
6	(53 75)	$\begin{pmatrix}1&0&0\\1&1&1\end{pmatrix}$	3	6
7	(133 171)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	3	6
8	(247 371)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	4	7
9	(561 753)	$\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}$	4	7
10	(1167 1545)	$\begin{pmatrix}1&0&0\\1&1&1\end{pmatrix}$	4	7
11	(2335 3661)	$\begin{pmatrix}1&0&1\\1&1&0\end{pmatrix}$	5	8
12	(4335 5723)	$\begin{pmatrix}1&0&0\\1&1&1\end{pmatrix}$	5	9
13	(10533 17661)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	5	9
14	(21675 27123)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	6	10
15	(55667 63121)	$\begin{pmatrix}1&0&1\\1&1&0\end{pmatrix}$	6	18
16	(111653 145665)	$\begin{pmatrix}1&0&0\\1&1&1\end{pmatrix}$	6	10
17	(34721 24277)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	7	12
18	(506477 673711)	$\begin{pmatrix}1&0&0\\1&1&1\end{pmatrix}$	7	12
19	(1352755 1771563)	$\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}$	7	12
20	(2451321 3546713)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	8	12
20	(2142513 3276177)	$\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$	8	13
21	(6567413 5322305)	$\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}$	8	12
22	(15724153 12076311)	$\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}$	8	13
23	(33455341 24247063)	$\begin{pmatrix}1&0&0\\1&1&1\end{pmatrix}$	9	14
24	(55076157 75501351)	$\begin{pmatrix}1&0&0\\1&1&1\end{pmatrix}$	9	15

– Table de codes poinçonnés de rendement $\frac{4}{5}$ à partir d'un code parent de rendement $\frac{1}{2}$ ($M = 4$) :

Code parent		Code poinçonné		
K	G	P	K_p	d_{libre}
5	(23 35)	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	2	3
6	(53 75)	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$	3	4
7	(133 171)	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	3	4
8	(247 371)	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	3	5
9	(561 753)	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	3	5
10	(1167 1545)	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	4	4
11	(2335 3661)	$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$	4	5
12	(4335 5723)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	4	6
13	(10533 17661)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	4	6
14	(21675 27123)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	5	7
15	(55667 63121)	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$	5	7
16	(111653 145665)	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	5	8
17	(34721 24277)	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$	5	8
18	(506477 673711)	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	6	8
19	(1352755 1771563)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	6	8
20	(2451321 3546713)	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$	6	9
20	(2142513 3276177)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	6	9

– Table de codes poinçonnés de rendement $\frac{5}{6}$ à partir d'un code parent de rendement $\frac{1}{2}$ ($M = 5$) :

Code parent		Code poinçonné		
K	G	P	K_p	d_{libre}
3	(5 7)	$\begin{pmatrix}1&0&1&1&1\\1&1&0&0&0\end{pmatrix}$	2	2
4	(15 17)	$\begin{pmatrix}1&0&1&0&0\\1&1&0&1&1\end{pmatrix}$	2	3
5	(23 35)	$\begin{pmatrix}1&0&1&1&1\\1&1&0&0&0\end{pmatrix}$	2	3
6	(53 75)	$\begin{pmatrix}1&0&0&0&0\\1&1&1&1&1\end{pmatrix}$	2	4
7	(133 171)	$\begin{pmatrix}1&1&0&1&0\\1&0&1&0&1\end{pmatrix}$	3	4
8	(247 371)	$\begin{pmatrix}1&1&1&0&0\\1&0&0&1&1\end{pmatrix}$	3	4
9	(561 753)	$\begin{pmatrix}1&0&1&1&0\\1&1&0&0&1\end{pmatrix}$	3	5
10	(1167 1545)	$\begin{pmatrix}1&0&1&1&1\\1&1&0&0&0\end{pmatrix}$	3	4
11	(2335 3661)	$\begin{pmatrix}1&1&0&0&0\\1&0&1&1&1\end{pmatrix}$	3	5
12	(4335 5723)	$\begin{pmatrix}1&1&0&0&0\\1&0&1&1&1\end{pmatrix}$	4	5
13	(10533 17661)	$\begin{pmatrix}1&0&0&1&1\\1&1&1&0&0\end{pmatrix}$	4	6
14	(21675 27123)	$\begin{pmatrix}1&1&0&1&0\\1&0&1&0&1\end{pmatrix}$	4	6
15	(55667 63121)	$\begin{pmatrix}1&1&1&1&1\\1&0&0&0&0\end{pmatrix}$	4	6
16	(111653 145665)	$\begin{pmatrix}1&0&1&0&1\\1&1&0&1&0\end{pmatrix}$	4	7
17	(34721 24277)	$\begin{pmatrix}1&1&0&0&0\\1&0&1&1&1\end{pmatrix}$	5	7
18	(506477 673711)	$\begin{pmatrix}1&0&0&0&0\\1&1&1&1&1\end{pmatrix}$	5	7
19	(1352755 1771563)	$\begin{pmatrix}1&1&1&1&1\\1&0&0&0&0\end{pmatrix}$	5	8
20	(2451321 3546713)	$\begin{pmatrix}1&1&0&1&0\\1&0&1&0&1\end{pmatrix}$	5	8
20	(2142513 3276177)	$\begin{pmatrix}1&1&1&1&0\\1&0&0&0&1\end{pmatrix}$	5	8

– Table de codes poinçonnés de rendement $\frac{6}{7}$ à partir d'un code parent de rendement $\frac{1}{2}$ ($M = 6$) :

Code parent		Code poinçonné		
K	G	P	K_p	d_{libre}
3	(5 7)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$	2	2
4	(15 17)	$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$	2	2
5	(23 35)	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$	2	3
6	(53 75)	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$	2	3
7	(133 171)	$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$	2	3
8	(247 371)	$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$	3	4
9	(561 753)	$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$	3	4
10	(1167 1545)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$	3	5
11	(2335 3661)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$	3	5
12	(4335 5723)	$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$	3	5
13	(10533 17661)	$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	3	6
14	(21675 27123)	$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$	4	6
15	(55667 63121)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	4	6
16	(111653 145665)	$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$	4	6
17	(34721 24277)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	4	7
18	(506477 673711)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$	4	7
19	(1352755 1771563)	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	4	7
20	(2451321 3546713)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$	5	7
20	(2142513 3276177)	$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$	5	7

– Table de codes poinçonnés de rendement $\frac{7}{8}$ à partir d'un code parent de rendement $\frac{1}{2}$ ($M = 7$) :

Code parent		Code poinçonné		
K	G	P	K_p	d_{libre}
3	(5 7)	$\begin{pmatrix}1&0&1&1&1&1&1\\1&1&0&0&0&0&0\end{pmatrix}$	2	2
4	(15 17)	$\begin{pmatrix}1&0&0&0&0&0&0\\1&1&1&1&1&1&1\end{pmatrix}$	2	2
5	(23 35)	$\begin{pmatrix}1&0&1&0&0&1&1\\1&1&0&1&1&0&0\end{pmatrix}$	2	3
6	(53 75)	$\begin{pmatrix}1&0&1&1&1&0&1\\1&1&0&0&0&1&0\end{pmatrix}$	2	3
7	(133 171)	$\begin{pmatrix}1&1&1&1&0&1&0\\1&0&0&0&0&0&1\end{pmatrix}$	2	3
8	(247 371)	$\begin{pmatrix}1&0&1&0&1&0&0\\1&1&0&1&0&1&1\end{pmatrix}$	2	4
9	(561 753)	$\begin{pmatrix}1&1&0&1&0&1&1\\1&0&1&0&1&0&0\end{pmatrix}$	3	4
10	(1167 1545)	$\begin{pmatrix}1&0&1&0&0&1&1\\1&1&0&1&1&0&0\end{pmatrix}$	3	4
11	(2335 3661)	$\begin{pmatrix}1&0&1&0&1&1&1\\1&1&0&1&0&0&0\end{pmatrix}$	3	4
12	(4335 5723)	$\begin{pmatrix}1&0&0&1&1&0&1\\1&1&1&0&0&1&0\end{pmatrix}$	3	5
13	(10533 17661)	$\begin{pmatrix}1&1&0&1&0&0&1\\1&0&1&0&1&1&0\end{pmatrix}$	3	5
14	(21675 27123)	$\begin{pmatrix}1&0&1&1&0&0&1\\1&1&0&0&1&1&0\end{pmatrix}$	3	5
15	(55667 63121)	$\begin{pmatrix}1&0&0&0&0&0&0\\1&1&1&1&1&1&1\end{pmatrix}$	3	6
16	(111653 145665)	$\begin{pmatrix}1&0&0&0&0&1&1\\1&1&1&1&1&0&0\end{pmatrix}$	4	6
17	(34721 24277)	$\begin{pmatrix}1&1&0&1&0&0&1\\1&0&1&0&1&1&0\end{pmatrix}$	4	6
18	(506477 673711)	$\begin{pmatrix}1&0&1&0&1&0&0\\1&1&0&1&0&1&1\end{pmatrix}$	4	6
19	(1352755 1771563)	$\begin{pmatrix}1&1&0&1&1&0&1\\1&0&1&0&0&1&0\end{pmatrix}$	4	7
20	(2451321 3546713)	$\begin{pmatrix}1&1&0&0&0&1&0\\1&0&1&1&1&0&1\end{pmatrix}$	4	7
20	(2142513 3276177)	$\begin{pmatrix}1&1&0&1&1&0&0\\1&0&1&0&0&1&1\end{pmatrix}$	4	7

– Table de différents poinçonnages d'un code de rendement $\frac{1}{3}$ pour $M = 3$ et 4 :

Code parent		Code poinçonné => $M = 3$				Code poinçonné => $M = 4$			
K	G	K_p	P	R	d_{libre}	K_p	P	R	d_{libre}
6	(75 53 47)	3				3	$\begin{pmatrix}1&1&1&1\\1&1&0&1\\1&1&1&1\end{pmatrix}$	4/11	11
			$\begin{pmatrix}1&1&1\\1&1&1\\0&1&1\end{pmatrix}$	3/8	11		$\begin{pmatrix}1&1&1&1\\1&1&0&1\\1&1&1&0\end{pmatrix}$	4/10	10
			$\begin{pmatrix}1&1&1\\1&0&1\\0&1&1\end{pmatrix}$	3/7	9		$\begin{pmatrix}1&1&1&1\\1&0&0&1\\1&1&1&0\end{pmatrix}$	4/9	10
			$\begin{pmatrix}1&1&1\\1&0&0\\0&1&1\end{pmatrix}$	3/6	8		$\begin{pmatrix}1&1&1&1\\1&1&0&1\\0&1&1&0\end{pmatrix}$	4/8	8
			$\begin{pmatrix}1&1&0\\1&0&0\\0&1&1\end{pmatrix}$	3/5	6		$\begin{pmatrix}1&1&1&1\\1&0&0&1\\0&1&0&0\end{pmatrix}$	4/7	6
			$\begin{pmatrix}0&1&0\\1&0&0\\0&1&1\end{pmatrix}$	3/4	4		$\begin{pmatrix}1&1&1&1\\0&0&0&1\\0&1&0&0\end{pmatrix}$	4/6	5
							$\begin{pmatrix}1&1&1&1\\0&0&0&1\\0&0&0&0\end{pmatrix}$	4/5	4
7	(171 165 133)	3				3	$\begin{pmatrix}1&1&1&1\\0&1&1&1\\1&1&1&1\end{pmatrix}$	4/11	13
			$\begin{pmatrix}1&1&1\\1&1&0\\1&1&1\end{pmatrix}$	3/8	12		$\begin{pmatrix}1&0&1&1\\0&1&1&1\\1&1&1&1\end{pmatrix}$	4/10	12
			$\begin{pmatrix}1&0&1\\1&1&0\\1&1&1\end{pmatrix}$	3/7	10		$\begin{pmatrix}1&0&1&1\\0&1&1&0\\1&1&1&1\end{pmatrix}$	4/9	10
			$\begin{pmatrix}1&0&1\\1&1&0\\1&1&0\end{pmatrix}$	3/6	9		$\begin{pmatrix}0&0&1&1\\0&1&0&0\\1&1&1&1\end{pmatrix}$	4/7	7
			$\begin{pmatrix}1&0&1\\0&1&0\\1&1&0\end{pmatrix}$	3/5	7		$\begin{pmatrix}1&0&1&1\\0&1&0&0\\1&1&1&1\end{pmatrix}$	4/8	9
			$\begin{pmatrix}1&0&1\\0&1&0\\0&1&0\end{pmatrix}$	3/4	5		$\begin{pmatrix}0&0&1&0\\0&1&0&0\\1&1&1&1\end{pmatrix}$	4/6	6
							$\begin{pmatrix}0&0&1&0\\0&1&0&0\\1&0&1&1\end{pmatrix}$	4/5	4

– Table de différents poinçonnages d'un code de rendement $\frac{1}{3}$ pour $M = 5$ et 6 :

La longueur de contrainte du code parent est : $K = 6$
La matrice génératrice du code parent est : $G = \begin{pmatrix} 75 & 53 & 47 \end{pmatrix}$

Code poinçonné => $M = 5$				Code poinçonné => $M = 6$			
K_p	P	R	d_{libre}	K_p	P	R	d_{libre}
2				2	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$	6/17	12
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/14	11		$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$	6/16	11
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/13	10		$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$	6/15	10
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	5/12	9		$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$	6/14	9
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	5/11	9		$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$	6/13	8
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$	5/10	8		$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$	6/12	7
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$	5/9	7		$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$	6/11	6
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$	5/8	6		$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$	6/10	6
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	5/7	5		$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	6/9	5
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	5/6	4		$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	6/8	4
					$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	6/7	3

– Table de différents poinçonnages d'un code de rendement $\frac{1}{3}$ pour $M = 5$ et 6 :

La longueur de contrainte du code parent est : $K = 7$
La matrice génératrice du code parent est : $G = \begin{pmatrix} 171 & 165 & 133 \end{pmatrix}$

Code poinçonné => $M = 5$				Code poinçonné => $M = 6$			
K_p	P	R	d_{libre}	K_p	P	R	d_{libre}
3				2	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6/17	13
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/14	13		$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6/16	12
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/13	12		$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6/15	11
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/12	11		$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6/14	10
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/11	10		$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6/13	10
	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/10	10		$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6/12	9
	$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/9	7		$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	6/11	8
	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/8	6		$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	6/10	7
	$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/7	5		$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	6/9	6
	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	5/6	3		$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	6/8	4
					$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$	6/7	4

Bibliographie

[3GP05a] 3GPP TS 05.03 v8.9.0. *Channel coding (release 1999).* The 3^{rd} Generation Partnership Project, Technical Specification Group GSM/EDGE Radio Access Network, January 2005. http ://www.3gpp.org.

[3GP05b] 3GPP TS 25.212 v6.5.0. *Multiplexing and channel coding (FDD) (release 6).* The 3^{rd} Generation Partnership Project, Technical Specification Group Radio Access Network, June 2005. http ://www.3gpp.org.

[3GP08] 3GPP TS 36.212 v8.3.0. *Multiplexing and channel coding (FDD) (release 8).* The 3^{rd} Generation Partnership Project, Technical Specification Group Radio Access Network, May 2008. http ://www.3gpp.org.

[3GP09] 3GPP2 C.S0002-E v0.99. *Physical layer Standard for CDMA2000 Spread Spectrum Systems (Revision E).* The 3^{rd} Generation Partnership Prject 2, February 2009. http ://www.3gpp2.org.

[AN96] E. Azzouz and A. Nandi. *Automatic modulation recognition of communication signal.* Kluwer Academic Publisher, 1996.

[Bar05] J. Barbier. Reconstruction of turbo-code encoders. In *Proceedings of SPIE Security and Defence, Space Communication Technologies Symposium*, volume 5819, pages 463–473, Orlando, FL, USA, March 2005.

[Bar07a] J. Barbier. *Analyse de canaux de communication dans un contexte non coopératif.* Thèse de doctorat, École Polytechnique, France, Novembre 2007.

[Bar07b] J. Barbier. Reconstructions des codes en blocs linéaires. Séminaire aintercom, CELAR, Département de Cryptographie, 31 mai 2007.

[BCJR74] L. R Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transaction on Information Theory*, 20(2) :284–287, March 1974.

[Ber07] C. Berrou. *Codes et turbocodes.* Springer, Paris, 2007.

[BG03] G. Burel and R. Gautier. Blind estimation of encoder and interleaver characteristics in a non cooperative context. In *IASTED International Conference on Communications, Internet and Information Technology*, Scottsdale, AZ, USA, Novembre 2003.

[BGT93] C. Berrou, A. Glavieux, and T. Thitimajshima. Near shannon limit error-correcting coding and decoding : turbo-codes. In *IEEE International Conference on Communications (ICC)*, pages 1064–1070, Geneva, Switzerland, May 1993.

[BHP90] G. Begin, D. Haccoun, and C. Paquin. Further results on high-rate punctured convolutional codes for viterbi and sequential decoding. *IEEE Transactions on Communictaions*, 38(11) :1922–1928, 1990.

[BK97] Irina E. Bocharova and Boris D. Kudryashov. Rational rate punctured convolutional codes for soft-decision viterbi decoding. *IEEE Transactions on Information Theory*, 43(4) :1305–1313, 1997.

[BRC60] AR. C Bose and D. K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1) :69–79, March 1960.

[BSH06] J. Barbier, G. Sicot, and S. Houcke. Algebraic approach for the reconstruction of linear and convolutional error correcting codes. *International journal of applied mathematics and computer sciences*, 2(3) :113–118, 2006.

[CCG79] J.B. Cain, G.C. Clark, and J.M. Geist. Punctured convolutional codes of rate (n-1)/n and simplified maximum likehood decoding. *IEEE Transactions on Information Theory*, IT-25(1) :97–100, January 1979.

[CF09] M. Cluzeau and M. Finiasz. Reconstruction of punctured convolutional codes. In *IEEE Information Theory Workshop (ITW)*, Taormina, Sicily, 11-16 October 2009.

[Cha07] C. Chabot. Recognition of a code in a noisy environment. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2211–2215, Nice, France, 24-29 June 2007.

[Clu04] M. Cluzeau. Reconstruction of a linear scrambler. In *IEEE International Symposium on Information Theory (ISIT)*, page 230, Chicago, USA, 2004.

[Clu06] M. Cluzeau. Block code reconstruction using iterative decoding techniques. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2269–2273, Seattle, USA, 2006.

[DBNS05] O. Dobre, Y. Bar-Ness, and W. Su. Blind modulation classification : A concept whose time has come. In *IEEE Sarnoff Symposium on Advances in Wired and Wireless Communication*, pages 223–228, Princeton, USA, 2005.

[DH07] J. Dingel and J. Hagenauer. Parameter estimation of a convolutional encoder from noisy observations. In *IEEE International Symposium on Information Theory (ISIT)*, pages 1776–1780, Nice, France, 24-29 June 2007.

[DVB09] DVB Document A054 Rev. 4.1. *Interaction channel for satellite distribution systems.* Digital Video Broadcasting, January 2009. http ://www.dvb.org.

[Eli55] P. Elias. Coding for noisy channels. *IRE International Convention Record*, 4 :37–46, 1955.

[Fan63] R. M Fano. A heuristic discussion of probabilistic decoding. *IEEE Transactions on Information Theory*, 9(2) :64–74, April 1963.

[Fil97] E. Filiol. Reconstruction of convolutional encoders over gf(p). In Michael DARNELL, editor, *Cryptography and Coding ; proceedings of the 6th IMA Conference, number 1355*, pages 101–109. Springer-Verlag, 1997.

[Fil00] E. Filiol. Reconstruction of punctured convolutional encoders. In *International Symposium on Information Theory and Application (ISITA)*, pages 4–7, Hawai, USA, November 2000.

[Fil01] E. Filiol. *Technique de reconstruction en cryptologie et théorie des codes.* Thèse de doctorat, École Polytechnique, France, Mars 2001.

[For66] G.D. Forney. *Concatened Codes.* PhD thesis, Massachusetts Institute of Technology, USA, 1966.

[For70] G. David Forney. Convolutional codes I : Algebraic structure. *IEEE Transactions on Information Theory*, 16(6) :720–738, November 1970.

[For71] G. David Forney. Burst-correctiong codes for the classis bursty channel. *IEEE Transactions on Communications Technology*, COM-19(5) :772–781, October 1971.

[For73] G. David Forney. Structural Analysis of Convolutional codes via Dual Codes. *IEEE Transactions on Information Theory*, 19(4) :512–5188, July 1973.

[Gal63] R. G Gallager. *Low-Density Parity-Check codes.* M.I.T Press, 1963.

[Gil60] E. N Gilbert. Capacity of a burst noise channel. *The Bell System Technical Journal*, 39 :1253–1266, September 1960.

[GMB08] R. Gautier, M. Marazin, and G. Burel. Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs are punctured. In *Proceedings of the 7-th IEEE-Communications 2008*, pages 345–348, Bucharest, Romania, 5-7 June 2008.

[Ham50] R. W Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, XXIX(2) :147–160, April 1950.

[HB89] D. Haccoun and G. Begin. High-rate punctured convolutional codes for viterbi and sequential decoding. *IEEE Transactions on Communictaions*, 37(11) :1113–1125, 1989.

[Hoc59] A. Hocquenghem. Codes correcteurs d'erreur. *Chiffres*, 2 :147–156, September 1959.

[Hol91a] Kjell Jorgen Hole. Punctured Convolutional Codes for the 1-D Partial-Reponse Channel. *IEEE Transactions on Information Theory*, 37(3) :808 – 817, 1991.

[Hol91b] Kjell Jorgen Hole. Rate k/(k+1) punctured convolutional encoders. *IEEE Transactions on Information Theory*, 37(3) :653 – 655, 1991.

[JZ99] R. Johannesson and K Sh. Zigangirov. *Fundamentals of Convolutional Coding*. IEEE series on Digital and Mobile Communication, IEE Press, 1999.

[LLZL05] P. Lu, S. Li, Y. Zou, and X. Luo. Blind recognition of punctured convolutional codes. *Science in China Ser. F Information Sciences*, 48(4) :484–498, 2005.

[LS06] L. H. C. Lee and J. Sodha. More new rate-compatible punctured convolutional codes for viterbi decoding. In *Proceedings of the 5th Workshop on Internet, Telecommunications and Signal Processing*, Hobart, Australia, December 2006.

[McE78] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report, Jet Prop. Lab., California Inst. Technol., Passadena, CA*, pages 114–116, January 1978.

[McE98] R.J. McEliece. The algebraic theory of convolutional codes. In *Handbook of coding theory*, volume 2, chapter 12, pages 1065–1138. Elsevier Science, 1998.

[MGB09a] M. Marazin, R. Gautier, and G. Burel. Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs are punctured. *MTA Review*, XIX(2) :213–232, June 2009.

[MGB09b] M. Marazin, R. Gautier, and G. Burel. Dual code method for blind identification of convolutional encoder for cognitive radio receiver design. In *the* 5^{th} *IEEE Broadband Wireless Access Workshop, IEEE GLOBECOM 2009*, Honolulu, Hawaii, USA, 30 November 2009.

[MGB09c] M. Marazin, R. Gautier, and G. Burel. Reconnaissance aveugle de turbocodes. Journées "codage et cryptographie", GDR IM (Informatique Mathématique), Fréjus, France, 04-09 Octobre 2009.

[MS67] J. L Massey and M. K Sain. Codes, automata, and continuous systems : explicit interconnections. *IEEE Transactions on Computers*, AC-12(6) :644–649, December 1967.

[MS68] J. L Massey and M. K Sain. Inverses of linear sequential circuits. *IEEE Transactions on Computers*, 17(4) :330–337, April 1968.

[PV97] G. Planquette and G. Vezzosi. Certaines propriétés des codeurs convolutifs. In *GRETSI*, pages 1077–1080, Grenoble, France, 15-19 septembre 1997.

[Ric95] B. Rice. Determining parameters of a rate $1/n$ convolutional encoder over gf(q). In *Proceedings of 3-rd International Conference on Finite Fields and Applications*, Glasgow, 1995.

[RS60] I. S Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, 8(2) :300–304, 1960.

[SH05a] G. Sicot and S. Houcke. Blind detection of interleaver parameters. In *IEEE International Conference on Acoustics, Speech and Signal Processiong (ICASSP)*, volume 3, pages 829–832, Philadelphia, Pennsylvania, Mars 2005.

[SH05b] G. Sicot and S. Houcke. Etude statistique du seuil dans la detection d'entrelaceur. In *GRETSI*, Louvain la Neuve, Belgique, 2005.

[Sha48] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27 :379–423, 623–656, July, October 1948.

[SHB09] G. Sicot, S. Houcke, and J. Barbier. Blind detection of interleaver parameters. *Signal Processing*, 89(4) :450–462, April 2009.

[Sic06] G. Sicot. *Étude du codage dans l'ADN.* Thèse de doctorat, École Nationale Supérieur des Télécommunications de Bretagne, France, Juin 2006.

[SLLZ04] Li Shen, P. Lu, X. Luo, and Y. Zou. Blind recognition of punctured convolutional codes. In *IEEE International Symposium on Information Theory (ISIT)*, page 457, Chicago, USA, July 2004.

[Val00] A. Valambois. *Décodage, détection et reconnaissance des codes linéaires binaires.* Thèse de doctorat, Université de Limoges, France, Octobre 2000.

[VF01] A. Valambois and M. Fossorier. An improved method to compute lists of binary vectors that optimize a given weight function with application to soft decision decoding. *IEEE Communications Letters*, 5(11) :456–458, November 2001.

[Vit67] A. J Viterbi. Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Transaction on Information Theory*, 13 :260–269, April 1967.

[WHZ07] F. Wang, Z. Huang, and Y. Zhou. A method for blind recognition of convolution code based on euclidean algorithm. In *Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1414–1417, Sept. 2007.

[Woz57] J. M Wozencraft. Sequential decoding for reliable communication. *IRE International Convention Record*, 5(2) :11–25, 1957.

Liste des publications

Conférences à comité de lecture

- R. Gautier, **M. Marazin** and G. Burel. "Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs are punctured". In *Procceeding of the* 7^{th} *IEEE-Communications 2008*, pages 345-348, Bucharest, Romania, 5-7 June 2008.

- **M. Marazin**, R. Gautier and G. Burel. "Reconnaissance aveugle de turbocode". Journées "codage et cryptographie", GDR IM (Informatique Mathématique), Fréjus, France, 4-9 Octobre 2009.

- **M. Marazin**, R. Gautier and G. Burel. "Dual code method for blind identification of convolutional encoder for cognitive radio receiver design". In *the* 5^{th} *IEEE Broadband Wireless Access Workshop, IEEE GLOBECOM 2009*, Honolulu, Hawaii, USA, 30 November 2009.

Revue à comité de lecture

- **M. Marazin**, R. Gautier and G. Burel. "Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs are punctured". *MTA Review*, XIX(2) :213-232, June 2009.

- V. Choqueuse, **M. Marazin**, L. Collin, K. Yao and G. Burel. "Blind recognition of linear Space-Time Block Codes : A Likelihood-based approach". *IEEE Transactions on Signal Processing*, vol. 53, no. 3, pp. 1290-1299, 2010.

Résumé

Mots clés : Radio cognitive - récepteur intelligent - identification aveugle - code convolutif - turbocode - entrelaceur - code poinçonné

Dans le but d'améliorer la qualité des transmissions numériques, les normes sont en perpétuelle évolution ce qui engendre des problèmes d'incompatibilité. Le domaine de la radio cognitive offre une solution pertinente à ce problème : la conception de récepteurs intelligents. Ces récepteurs devront être capables d'identifier en aveugle les paramètres de la norme utilisée par l'émetteur. Les travaux présentés dans ce mémoire portent sur la reconnaissance aveugle de codeur à base de code convolutif. De tels codes améliorent la fiabilité d'une transmission en permettant, côté récepteur, de détecter et/ou de corriger d'éventuelles erreurs intervenues lors de la transmission du message.

Une étude sur la théorie algébrique des codes convolutifs a été conduite afin d'obtenir les propriétés indispensables à la mise en oeuvre de méthodes d'identification aveugle. Puis, à partir de la seule connaissance d'un train binaire codé, nous avons développé des méthodes permettant d'identifier un code convolutif lors d'une transmission non-bruitée, puis bruitée. Nous avons ensuite développé un algorithme dédié à l'identification en aveugle d'un code convolutif poinçonné. Cet algorithme permet, à partir de la seule connaissance d'une trame codée, poinçonnée et bruitée, d'identifier le code convolutif ainsi que le motif de poinçonnage utilisé à l'émission. Ensuite, nous proposons deux méthodes qui permettent d'identifier un turbocode lors de la réception d'une trame présentant une plage de donnée non-bruitée. Enfin, nous montrons que nos algorithmes d'identification offrent d'excellentes performances avec des paramètres de simulation proches de ceux utilisés dans les différents standards.

Abstract

Keywords : Cognitive radio - intelligent receiver - blind identification - convolutional encoder - turbocode - interleaver - punctured code

For enhancement of the quality of digital transmissions, the standards are in continual evolution, but this generates compatibility problems. Cognitive radio systems provide a relevant solution to this problem : the design of an intelligent receiver. Such receivers must be able to blindly identify the transmitter parameters. This PhD work was focused on the blind recognition of encoders based on a convolutional code. Such codes enhance the reliability of transmissions by allowing, at the receiver side, the detection and/or correction of errors liable to occur over the data transmission

Our investigations started with a study of the algebraic theory of convolutional codes so as to get the properties essential for a blind recognition of convolutional encoders. Then, from the knowledge of only the coded data stream, methods were developed to identify a convolutional code in the case of a noiseless transmission, and then of a noisy one. An algorithm able to identify a punctured convolutional code was devised. From the knowledge of only a stream of encoded, punctured and noisy data, this algorithm permits the identification of the convolutional encoder and that of the puncturing pattern in use at the transmitter side. Then, two methods are proposed to identify a turbocode with noiseless parts of the received data stream. Finally, application of our blind identification algorithms to simulation parameters close to those used by the available standards gave good performances.