Algorithmes de détection et de reconstruction en aveugle de code correcteurs d'erreurs basés sur des informations souples

Aurélien Bonvard

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Télécommunications*

Par
Aurélien BONVARD

Algorithmes de détection et de reconstruction en aveugle de code correcteur d'erreurs basés sur des informations souples

Thèse présentée et soutenue en télésoutenance (Brest - Paris), le 25 Mai 2020
Unité de recherche : Lab-STICC UMR-CNRS 6285
Thèse N° : 2020IMTA0178

Rapporteurs avant soutenance :

Jean-Pierre Tillich	Chercheur INRIA
Pierre Loidreau	Ingénieur DGA et chercheur IRMA Rennes

Composition du Jury :

Président :	Philippe Ciblat	Professeur - Telecom ParisTech
Rapporteurs :	Jean-Pierre Tillich	Chercheur INRIA
	Pierre Loidreau	Ingénieur DGA et chercheur IRMA Rennes
Examinateurs :	Mélanie Marazin	Maître de conférences – Université de Bretagne Occidentale
Dir. de thèse :	Sébastien Houcke	Professeur – IMT Atlantique
Co-dir. de thèse :	Roland Gautier	Maître de conférences (HDR) - Université de Bretagne Occidentale

N° d'ordre : 2020IMTA0178

Sous le saut de l'Université de Bretagne Loire

IMT Atlantique
Bretagne - Pays de la Loire

En accréditation conjointe avec l'École Doctorale MathSTIC

Algorithmes de détection et de reconstruction en aveugle de code correcteur d'erreurs basés sur des informations souples

---

Thèse de Doctorat
Spécialité : Télécommunications

Présenté par : Aurélien Bonvard

Département : Signal et Communications
Laboratoire : UMR CNRS 6285 Lab-STICC

Directeur de thèse : Sébastien Houcke

Soutenue le : 25 mai 2020

Jury :

Président :	Philippe Ciblat	Professeur - Telecom ParisTech
Rapporteurs :	Jean-Pierre Tillich	Chercheur INRIA
	Pierre Loidreau	Ingénieur DGA et chercheur IRMA Rennes
Examinateurs :	Mélanie Marazin	Maître de conférences – UBO
Dir. de thèse :	Sébastien Houcke	Professeur – IMT Atlantique
Co-dir. de thèse :	Roland Gautier	Maître de conférences (HDR) - UBO

Remerciements

Je tiens à remercier chaleureusement Sébastien Houcke, Roland Gautier et Mélanie Marazin avec qui j'ai eu le plaisir de travailler à l'IMT Atlantique et à l'UBO pendant toutes ces années. Je tiens à leur faire part de ma gratitude pour leur soutien, tant sur le plan scientifique que sur les plans moral et psychologique. Ce sont leurs conseils avisés, leur rigueur scientifique, et de manière générale la qualité de leur encadrement qui m'ont permis de mener ces travaux à bien. De même, je tiens à remercier Gilles Burel pour son expertise et ses remarques qui m'ont permis de me sortir de ce qui me semblait être une impasse.

J'adresse mes remerciements aux membres du jury pour l'intérêt qu'ils ont porté à mes travaux. Merci à Philippe Ciblat d'avoir présidé le jury. Je remercie Jean-Pierre Tillich et Pierre Loidreau pour le temps et le soin qu'ils ont porté à la lecture du présent manuscrit.

Merci à mes parents, à mes soeurs et à mon frère pour leur présence. Merci à Stéphanie pour le soutien et la patience dont elle fait preuve toutes ces années. Finalement, un grand merci aux doctorants du département SC : Antony, Manuel, Nicolas A. et Thomas pour nos agréables réunions musicales et pour les grandes digressions lors de nos nombreuses discussions, Nicolas G. avec qui j'ai parcouru un grand nombre de kilomètres en courant le midi. De même, Amine, Carole, Rédouane et Saïd avec qui j'ai partagé le bureau et beaucoup de pensées au détour de longues conversations. Enfin, aux membres du département SC avec qui j'ai passé ces quelques années et à qui je témoigne ma plus sincère amitié.

Table des matières

Résumé

Mots clés : *Internet des Objets - Radio cognitive - Codes linéaire - Identification aveugle - Reconstruction de codes.*

Les dernières décennies ont connu l'essor des communications numériques. Elles sont présentes dans un grand nombre d'applications, qu'il s'agisse de la téléphonie mobile, de la télévision numérique ou encore de l'internet. Ceci a donné lieu à la prolifération des standards de communication, ce qui demande une plus grande adaptabilité des systèmes de communication. Une manière de rendre ces systèmes plus flexibles consiste à concevoir un récepteur intelligent qui serait capable, à partir du signal reçu, de retrouver l'ensemble des paramètres de l'émetteur. Dans ce manuscrit, nous nous intéressons à l'identification en aveugle des codes correcteurs d'erreurs. Nous proposons des méthodes originales, qui utilisent l'information souple (la fiabilité des bits est conservée), basées sur le calcul de distances euclidiennes entre des séquences de symboles bruités.

Toute l'étude proposée s'appuie sur l'observation des distributions de distances euclidiennes en présence d'un code correcteur d'erreurs. Tout d'abord, un premier algorithme de classification utilisant les distances euclidiennes permet la détection d'un code puis l'identification de la longueur de ses mots de code. Un second algorithme basé sur le nombre de collisions permet quand à lui d'identifier la longueur des mots d'informations. Ensuite, nous proposons une autre méthode utilisant cette fois les distances euclidiennes minimales pour l'identification de la longueur d'un code en bloc. Enfin, une méthode de reconstruction du code dual d'un code correcteur d'erreurs est présentée. Cette méthode repose sur l'observation de la distribution des distances en présence d'un code de parité.

Notations

Notations mathématiques

x	Scalaire
$\mathbf{x}$	Vecteur
$x(i)$	i-ième bit du vecteur $\mathbf{x}$
$\mathbf{X}$	Matrice
$\mathbf{x}_{k,l}$	Élément à la k-ième ligne et la l-ième colonne de la matrice $\mathbf{X}$
$\mathbf{X}_{.,l}$	l-ième colonne de la matrice $\mathbf{X}$
$\mathcal{X}$	Ensemble générique
$\mathbb{N}$	Ensemble des entiers naturels
GF(2)	Corps de Galois de cardinal 2

Symboles et opérateurs

$\mathbf{I}_n$	Matrice identité de taille $(n \times n)$
$\mathbf{X}^T$	Transposée de la matrice $\mathbf{X}$
$\text{Card}(\mathcal{X})$	Cardinal de l'ensemble $\mathcal{X}$, *i.e.*, le nombre d'éléments dans l'ensemble $\mathcal{X}$
$d_H(\mathbf{x}, \mathbf{y})$	Distance de Hamming entre les vecteurs $\mathbf{x}$ et $\mathbf{y}$, *i.e.* $\text{Card}(\{i \vert x(i) \neq y(i)\})$
$d_E(\mathbf{x}, \mathbf{y})$	Distance Euclidienne entre les vecteurs $\mathbf{x}$ et $\mathbf{y}$, *i.e.*, $\sqrt{\sum_i (\mathbf{x} - \mathbf{y})^2}$
$\lfloor x \rfloor$	Arrondi à l'entier inférieur de x
$\lceil x \rceil$	Arrondi à l'entier supérieur de x
$\lfloor x \rceil$	Arrondi à l'entier le plus proche de x
$\binom{m}{i}$	Coefficient binomial (i parmi m)

Probabilité, statistique et variables aléatoires

X	Variable aléatoire
$\mathbb{P}(x)$	Probabilité d'un événement x

$\mathbb{E}(X)$	Espérance de X
$\mathcal{N}(\mu, \sigma^2)$	Distribution de loi normale de moyenne μ et de variance σ^2
$\mathcal{U}(\mathcal{X})$	Distribution de loi uniforme sur l'ensemble $\mathcal{X}$
$\chi^2(n)$	Distribution de loi du khi 2 à n degrés de liberté
$X \sim \mathcal{L}$	Variable aléatoire X distribuée selon la loi $\mathcal{L}$

Notations spécifiques à la transmission numérique

n_c	Longueur du code
k_c	Dimension du code
K_c	Longueur de contrainte
r_c	Rendement du code
$\mathcal{C}(n_c, k_c)$	Code en bloc de paramètres n_c et k_c
$\mathcal{C}(n_c, k_c, K_c)$	Code convolutif de paramètres n_c, k_c et K_c
$\mathcal{C}^T$	Code dual du code $\mathcal{C}$
$\mathbf{m}$	Vecteur de symboles d'information (mots d'information)
$\mathbf{c}$	Vecteur de symboles encodés (mots de code)
$\tilde{\mathbf{c}}$	Vecteur de symboles encodés entaché d'erreurs par un canal binaire symétrique
$\mathbf{y}$	Vecteur de symboles encodés et bruités par un canal à bruit blanc additif gaussien
$w(\mathbf{a})$	Poids de Hamming du vecteur $\mathbf{a}$
$\mathbf{G}$	Matrice génératrice d'un code
$\mathbf{H}$	Matrice de parité d'un code
$\mathbf{y}$	Séquence de symboles bruités reçue
N	Nombre de symboles dans la séquence reçue
L_n	Nombre de blocs de taille n reçus, $L_n = \lfloor \frac{N}{n} \rfloor$
p_e	Probabilité d'erreur sur le canal binaire symétrique
σ_b^2	Variance du canal à bruit blanc additif gaussien

Notations spécifiques au chapitre 2 et 3

$Z^{(n)}$	Variable aléatoire : nombre de classes obtenu avec des blocs i.i.d. de taille n
$z_c^{(n)}$	Nombre de classes mesuré obtenu avec des blocs contigus de taille n issus d'une trame codée
$\Delta_{DNC}(n)$	Déficience du nombre de classes pour des blocs de taille n
$\mathfrak{B}_i^{(n)}$	i-ième bloc de taille n
β	Seuil de classification

β_{opt}	Seuil optimal de classification
$\mathcal{R}^{(n)}$	Ensemble des mots de référence de taille n
$\mathcal{R}^{(n,\beta)}$	Ensemble des mots de référence de taille n après classification avec un seuil β
$\varphi(n,\beta)$	Déficience du nombre de classes normalisée en fonction de la taille des blocs n et du seuil sur les distances β
$\mathcal{I}_b(k)$	Ensemble des indices des blocs dont la distance euclidienne au k-ième bloc est inférieure ou égale à b
$\mathcal{F}_b(k)$	Ensemble des indices des blocs dont la distance euclidienne à un mot de référence est inférieure ou égale à b
β_{col}	Seuil de collision
$\mathbf{P}_{fa}$	Probabilité de fausse alarme
$\mathbf{P}_d$	Probabilité de détection
$\mathbf{P}_{col}$	Probabilité de collision
$\mathbf{B}^{(n)}$	Matrice de distance de mots de taille n (à partir de la séquence interceptée)
$\mathbf{D}^{(n)}$	Matrice de distance de mots de taille n ordonnée (à partir de la séquence interceptée)
$\mathbf{X}^{(n)}$	Matrice de distance de mots de taille n ordonnée (à partir d'une séquence i.i.d.)
$\mathbf{e}_d^{(n)}$	Vecteur des erreurs absolues composante à composante entre $\mathbb{E}(x^{(n)}(i))$ et $d^{(n)}(i)$ pour des blocs de taille n
$\varepsilon(\mathbf{e}_d^{(n)})$	Erreur absolue moyenne entre $\mathbb{E}(x^{(n)}(i))$ et $d^{(n)}(i)$ pour des blocs de taille n
$\mathbf{P}_{n_c}$	Probabilité de correctement identifier la longueur d'un code
$\mathbf{P}_{k_c}$	Probabilité de correctement identifier la dimension d'un code
$\hat{n}_c$	Estimation de la longueur d'un code n_c
$\hat{k}_c$	Estimation de la dimension d'un code k_c

Notations spécifiques au chapitre 4

$\mathbf{h}$	Relation de parité testée
w	Poids des relations $\mathbf{h}$ testées
$\mathcal{L}_w$	Ensemble de relations de parité de poids w identifiées
N_h	Nombre de relations identifiées
$\mathbf{M}_{soft}$	Matrice d'interception contenant des informations souples

$\mathbf{x}^{(\mathbf{h})}$	Vecteur de distances ordonnées mesurées à partir de la matrice $\mathbf{M}_{soft}$ pour la relation testée $\mathbf{h}$
$\mathbf{X}^{(par)}$	Échantillon ordonné de variables aléatoires suivant la distribution des distances pour des mots d'un code de parité
$\mathbf{X}^{(iid)}$	Échantillon ordonné de variables aléatoires suivant la distribution des distances pour des mots i.i.d.
N_B	Nombre de blocs contigus retenus pour l'identification
$\hat{\mathbf{x}}^{(\mathbf{h})}$	Vecteur de distances ordonnées moyennes pour la relation testée $\mathbf{h}$
$\mathcal{A}_{\mathbf{h}}^{(par)}$	Mesure d'adéquation de $\hat{\mathbf{x}}^{(\mathbf{h})}$ à la distribution de $\mathbf{X}^{(par)}$
$\mathcal{A}_{\mathbf{h}}^{(iid)}$	Mesure d'adéquation de $\hat{\mathbf{x}}^{(\mathbf{h})}$ à la distribution de $\mathbf{X}^{(iid)}$
S_a	Seuil sur le critère de décision $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$
$\mathbf{P}_{md}$	Probabilité de mauvaise détection
$\mathbf{M}$	Matrice d'interception binaire
p_s	Proportion de mots de $\mathbf{M}$ vérifiant une relation $\mathbf{h}$
S	Seuil sur le poids de $\mathbf{h} \cdot \mathbf{M}^T$

Liste des acronymes et abréviations

Par soucis de lisibilité, la signification des abréviations et acronymes définie ici est aussi rappelée à leurs premières apparitions dans le corps des chapitres.

ADSL	Asymmetric Digital Subscriber Line
AMC	Adaptive Modulation and Coding
BBAG	Bruit Blanc Additif Gaussien
BCJR	Bahl Cocke Jelinek Raviv
BCH	Bose Chaudhuri Hocquenghem
CBS	Canal Binaire Symétrique
DNC	Déficience du Nombre de Classes
DR	Déficience de Rang
ESA	European Space Agency
GF	Galois Field
i.i.d.	indépendant et identiquement distribué
IoT	Internet of Things
ISD	Information Set Decoding
LDPC	Low Density Parity Check Code
MAP	Maximum A Posteriori
MP-2	Modulation de Phase à deux états
NASA	National Aeronautics and Space Administration
SOD	Statistique d'Ordre sur les Distances Administration
SPP	Syndrome a Posteriori Probability

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction

Les dernières décennies ont connu l'essor des communications numériques. Elles sont présentes dans un grand nombre d'applications, qu'il s'agisse de la téléphonie mobile, de la télévision numérique ou encore de l'internet. Tout particulièrement, la prolifération des équipements connectés a marqué l'avènement de l'Internet des Objets (IoT pour Internet of Things). En effet, l'IoT est présent dans beaucoup de domaines, qu'il s'agisse du domaine de la santé, de la domotique, de l'industrie, etc.

Le principe d'une communication numérique est toujours le même, il s'agit de transmettre une information d'un émetteur vers un récepteur, sous forme binaire par exemple appelé *bit* (contraction de binary digit) et prenant pour valeur 0 ou 1. La transmission de l'information entre l'émetteur et le récepteur est réalisée à travers un canal où le signal (vecteur de l'information) à transmettre peut éventuellement subir des perturbations. Ce signal peut contenir des données de type : fichiers ou flux vidéo, audio, des images, du texte, etc. Pour effectuer ce transfert de données, l'émetteur et le récepteur suivent un protocole qui leur permettra de mener cette opération à bien. Dans un premier temps l'émetteur doit transformer les données en une suite d'éléments binaires qui correspond au message à transmettre. Puis, une première étape, appelée codage de source, est ensuite réalisée afin de compresser l'information à transmettre, dans le but, entre autres, d'améliorer le débit de la transmission. Ensuite, afin de protéger l'information des erreurs engendrées par le canal de transmission, une redondance contrôlée est ajoutée au message par le codage de canal. Pour finir, le message est transformé en un signal qui pourra être transmis via le canal de transmission par le modulateur. A la réception, les étapes inverses de l'émetteur sont réalisées afin de retrouver le message initial. Premièrement, le démodulateur permet de retrouver l'information sous forme de symboles à partir du signal reçu. Ensuite, le décodage de canal a pour rôle de détecter voire de corriger les erreurs dues à la transmission. Finalement, le décodage de source consiste à décompresser le signal pour retrouver le message initial.

Le bloc de codage de canal a un rôle très important dans une chaîne de transmission numérique, puisqu'il permet de protéger des données du bruit et des perturbations engendrées par le canal de transmission en ajoutant de la redondance contrôlée. De nombreuses recherches ont été effectuées afin de trouver des codes qui permettent de garantir une capacité de correction des erreurs très élevée tout en garantissant un débit important. En effet, dans cette optique, les standards de la téléphonie mobile, que l'on parle des anciennes générations ou encore de la 5G, permettent l'utilisation de codes longs pour protéger les données. Toutefois, les nouveaux

standards développés pour les objets connectés ont des contraintes différentes car ils doivent tenir compte des spécificités de ces nouvelles communications. Pour l'IoT, les capteurs doivent consommer peu d'énergie, il est donc impossible d'utiliser les mêmes standards qu'en téléphonie mobile et donc les mêmes codes. Dans ce contexte d'IoT, les codes utilisés sont plutôt des codes courts.

Contexte et objectifs

Les transformations du signal nécessaires à une transmission numérique demandent un grand nombre d'opérations décrites dans les standards de communication spécifiques aux différents champs d'application (téléphonie mobile, télévision numérique, objets connectés, etc.). L'augmentation du nombre d'utilisateurs et l'accroissement exponentiel du nombre d'objets connectés demandent une plus grande adaptabilité des systèmes de communication. Il devient nécessaire que les systèmes gagnent en flexibilité pour qu'ils puissent s'adapter à leurs conditions environnementales, telles que le nombre d'utilisateurs, la qualité du canal de transmission, etc. De plus, pour les systèmes IoT, il est nécessaire qu'ils s'adaptent aux propriétés intrinsèques des objets que ce soit en matière de consommation énergétique, de sécurité, de coût de gestion et de transmission. L'explosion du nombre de standards ainsi que du nombre d'utilisateurs pose également le problème de l'allocation de la ressource spectrale. Le domaine de la radio intelligente (Cognitive Radio) apparaît comme une solution naturelle à ces problèmes.

Une solution globale à ces problèmes consisterait donc à concevoir un récepteur intelligent qui serait capable, à partir du signal reçu, de retrouver l'ensemble des paramètres de l'émetteur. Il n'y aurait dans ce cas plus aucun problème d'incompatibilité entre toutes ses normes et l'obsolescence de ces objets diminuerait. Cependant ce contexte bien particulier est dit «non-coopératif». Le récepteur n'a aucune information sur l'émetteur et il est alors nécessaire de mettre en oeuvre des algorithmes lui permettant de retrouver l'ensemble des paramètres de l'émetteur afin de décoder le signal reçu. Pour rendre les systèmes plus adaptatifs sans pour autant subir des contraintes aussi fortes que dans le contexte non-coopératif, il est possible de se placer dans un contexte semi-coopératif. Dans ce cas, le récepteur aura accès à une liste de paramètres que l'émetteur peut utiliser et il devra, à partir du signal reçu et de cette liste, déterminer les paramètres utilisés par l'émetteur. Les techniques dites d'AMC (Adaptive Modulation and Coding) sont particulièrement adaptées à cette configuration puisqu'elles permettent de choisir le codage et la modulation utilisés en fonction de l'état du canal.

Dans ce manuscrit, nous nous intéressons à l'identification en aveugle du bloc de codage de canal. Nous développerons des méthodes qui permettent d'identifier le code correcteur d'erreurs utilisé par l'émetteur à partir de la seule connaissance du signal reçu. Les méthodes qui traitent de ce sujet dans la littérature utilisent, pour la majorité, des informations fermes (des 0 et des 1). Dans ce manuscrit, nous proposons des méthodes originales, qui utilisent l'information souple (la fiabilité des bits est conservée), basées sur le calcul de distances euclidiennes entre des séquences de symboles bruités.

Organisation du manuscrit

Pour répondre à ces problématiques, la suite de ce manuscrit est organisée de la manière suivante :

Charitre 1

Ce premier chapitre présente le contexte de l'étude. Nous définirons les différents modèles de canaux utilisés et les familles de codes correcteurs d'erreurs que recouvrent cette étude. Enfin, un état de l'art sur l'identification de paramètres et sur la reconstruction de codes sera fait. Tous ces éléments nous permettront de situer notre étude.

Charitre 2

Dans ce chapitre nous proposons une méthode permettant l'identification des paramètres d'un code correcteur d'erreurs à partir d'une trame bruitée. Un premier algorithme de classification utilisant les distances euclidiennes permet l'identification de la longueur des mots de code. Un second algorithme basé sur le nombre de collisions permet quand à lui d'identifier la longueur des mots d'informations. Enfin, une analyse des performances de notre méthode est présentée.

Charitre 3

A l'instar du chapitre 2, nous proposons une autre méthode permettant l'identification de la longueur des mots de code en présence de bruit. La présentation de cette méthode basée sur les distances euclidiennes minimales est suivie d'une analyse sur ses performances. Enfin, une comparaison entre nos deux méthodes d'identification est réalisée.

Charitre 4

L'identification des paramètres d'un code correcteur d'erreurs ne suffit pas pour décoder les données reçues et ainsi retrouver le message émis. Dans ce chapitre, nous proposons donc une méthode de reconstruction de codes correcteurs d'erreurs. Cette méthode repose sur l'observation de la distribution des distances d'un code de parité. Nous présentons également une méthode de reconstruction connue qui est basée sur le paradoxe des anniversaires. Pour finir, nous comparons notre méthode de reconstruction à celle basée sur le paradoxe des anniversaires.

La fin de ce manuscrit apporte des conclusions et quelques perspectives sur les méthodes présentées.

Chapitre 1

État de l'art sur la reconnaissance aveugle de codes correcteurs d'erreurs

Dans ce premier chapitre, le contexte de l'étude portant sur la reconstruction en aveugle des codes correcteurs d'erreurs est présenté. Dans la première section, nous définissons les contextes non-coopératif et semi-coopératif ainsi que les canaux de transmission utilisés dans notre étude. Nous présentons ensuite quelques généralités sur les codes correcteurs d'erreurs linéaires. Dans la seconde section, nous proposons un état de l'art sur les méthodes d'identification et de reconnaissance, aveugle ou semi-aveugle, des codes correcteurs d'erreurs.

1.1 Contexte et modèle de transmission

Le rôle d'une chaîne de transmission numérique est de transmettre une information d'un émetteur à un récepteur. Dans le cas d'une transmission numérique classique, le récepteur connaît les paramètres de l'émetteur et il peut aisément retrouver l'information transmise. Toutefois, si nous nous plaçons dans un contexte plus défavorable, le récepteur peut n'avoir qu'une connaissance partielle voire nulle des paramètres de l'émetteur. Dans ce cas, retrouver l'information transmise sera difficile voire impossible pour le récepteur. Il s'agit du contexte non-coopératif ou semi-coopératif, selon que le récepteur connaît ou non quelques informations sur les paramètres de l'émetteur. La reconnaissance en aveugle des codes correcteurs d'erreurs se situe dans ce contexte. Dans notre étude, nous ferons l'hypothèse que le récepteur ne connaît pas tous les paramètres du code utilisé par l'émetteur et l'objectif sera de retrouver ses paramètres afin de pouvoir remonter à l'information transmise.

1.1.1 Contexte non-coopératif

Usuellement, une chaîne de transmission standard permet d'acheminer de l'information d'une source vers un destinataire. La Figure 1.1 détaille le principe de cette chaîne de transmission numérique. Cette chaîne se décompose en plusieurs étapes telles que le codage de source, le codage de canal et la modulation. A l'émission, le codage de source permet la compression de l'information initiale pour éliminer toute redondance et ainsi augmenter l'efficacité de la transmission. Le

codage de canal permet quant à lui de rendre la transmission robuste aux altérations qui pourraient survenir lors du passage par le canal en ajoutant de la redondance. Enfin, la modulation permet d'adapter la forme du signal que l'on veut transmettre aux conditions environnementales. Ces conditions dépendent, par exemple, du support utilisé pour la transmission (aérien, filaire, etc.), de la présence d'autres utilisateurs (coopérant ou pas), etc. A la réception, ces trois opérations sont inversées. La démodulation consiste à traduire le signal en symboles (binaires ou non-binaires) possiblement entachés d'erreurs. Le démodulateur peut donner une information souple ou ferme. Le décodage de canal permet quand à lui de détecter et éventuellement de corriger les erreurs engendrées par le canal de transmission. Et enfin le décodage de source restitue le message original.

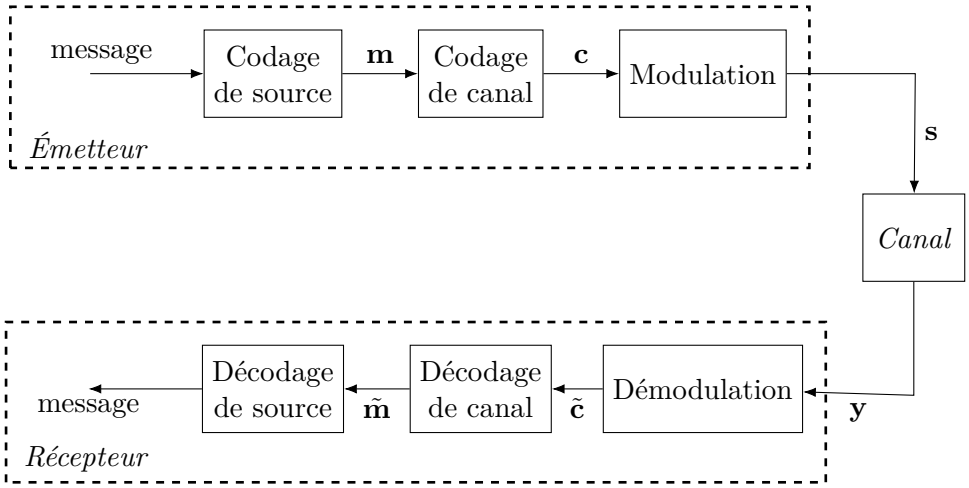


FIGURE 1.1 – Modèle d'une transmission dans un contexte coopératif

Le fonctionnement de cette chaîne de transmission standard repose sur la coopération entre l'émetteur et le récepteur. Avant de commencer à communiquer, ces deux derniers s'accordent sur le choix de la modulation et des codeurs. Ainsi, le récepteur peut «aisément» effectuer les opérations inverses de celles de l'émetteur pour obtenir le message original. Dans un contexte non-coopératif, le récepteur n'a pas nécessairement toutes ces informations à disposition. Dans ces conditions, le récepteur ne peut pas effectuer les opérations inverses de l'émetteur. Cette configuration, bien qu'issue du domaine militaire s'applique également au domaine civile. Dans le domaine militaire, lors de l'interception d'une communication, nous nous retrouvons automatiquement dans le cas du contexte non-coopératif. En effet, dans cette configuration, seul le signal intercepté est connu. Dans le domaine civil, la prolifération des normes et des standards engendrent des problèmes d'incompatibilités entre les émetteurs et les récepteurs. Il peut donc arriver qu'un récepteur ne connaisse pas les paramètres de l'émetteur, dans ces conditions le récepteur ne peut pas décoder l'information. Ces incompatibilités sont de plus en plus fréquentes, en partie à cause de la forte augmentation du nombre d'objets connectés en réseaux via l'Internet. Pour palier à ces problématiques, des nouveaux protocoles d'échange sont créés pour administrer ce réseau de connections : l'IoT (Internet of Things : Internet des Objets).

Pour rendre ces échanges viables, l'adaptation en temps réel d'un récepteur aux paramètres de différents récepteurs peut être une solution. Ce type d'adaptations est une problématique directement traitée à travers le concept de la *radio cognitive*. Par exemple, l'accès opportuniste au spectre d'émission demande de sonder le canal pour déterminer sur quelle bande il est possible de transmettre et le cas échéant, choisir les paramètres de transmission les plus adaptés. De ce fait, de plus en plus de normes utilisent des techniques de transmission adaptatives. Ces techniques consistent à adapter le codage et la modulation utilisés en fonction de l'état du canal de transmission. Ces techniques, appelées couramment AMC (Adaptive Modulation and Coding) ont pour objectif d'améliorer l'efficacité spectrale pour atteindre des débits plus élevés tout en maintenant la qualité de la transmission. Dans ces conditions, l'émetteur doit connaître l'état du canal en temps réel afin d'adapter ses paramètres de transmission. Le récepteur doit quant à lui, évaluer l'état du canal d'une part, et connaître les paramètres utilisés par l'émetteur d'autre part. Dans ce cas, nous nous situons dans le contexte semi-coopératif où le récepteur a une liste de codeurs et de modulateurs que l'émetteur peut utiliser. A partir du signal reçu et de cette liste, le récepteur doit identifier le codeur et le modulateur utilisés par l'émetteur.

La Figure 1.2 fait une synthèse des différences entre le contexte coopératif et le contexte non-coopératif. Dans le cas coopératif, le récepteur aura accès à tous les paramètres de l'émetteur (codeurs et modulateur) et au message reçu $\mathbf{y}$. Par contre, dans le cas non-coopératif, le récepteur n'aura accès qu'au signal reçu ou intercepté $\mathbf{y}$. Dans la suite de notre étude, nous considérons donc le modèle de transmission simplifié décrit par la Figure 1.2. Côté émetteur, en sortie du codage de source, le message $\mathbf{m}$ est transformé en séquence codée $\mathbf{c}$ puis modulé en un signal $\mathbf{s}$ et transmis à travers un canal. Du côté du récepteur, la seule information connue est donc le signal reçu $\mathbf{y}$. A partir de $\mathbf{y}$, l'objectif sera de retrouver les paramètres utilisés à l'émission afin de retrouver le message émis $\mathbf{m}$.

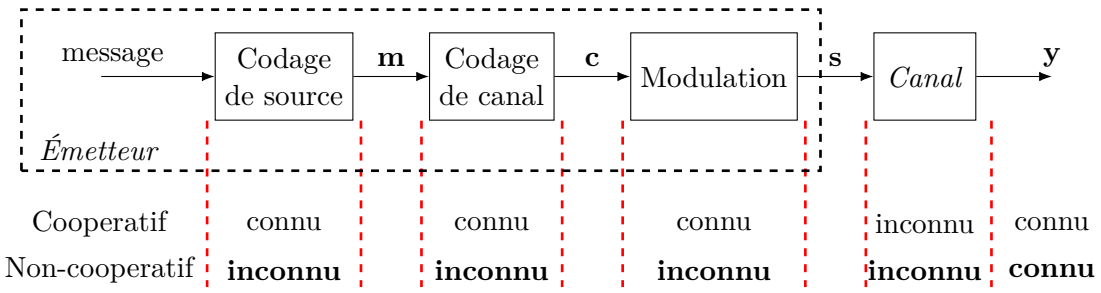


FIGURE 1.2 – Comparaison des contextes coopératif et non coopératif

1.1.2 Canaux de transmission

Le rôle du canal de transmission est de propager l'information d'une source vers un destinataire. Ce canal représente le support physique qui transmet l'information. La forme physique du signal est adaptée au milieu qui peut provoquer certaines modifications du signal, telles que des atténuations, des retards, des interférences, etc. Les canaux de transmission peuvent être

classés en deux groupes, les canaux stationnaires (paramètres fixes) et non-stationnaires (les paramètres ainsi que les propriétés statistiques du canal changent). Dans cette section, nous présentons deux modèles de canaux stationnaires qui serviront le propos de notre étude, le canal binaire symétrique (CBS) et le canal à Bruit Blanc Additif Gaussien (BBAG).

1.1.2.1 Canal binaire symétrique

Le canal binaire symétrique est un modèle de canal discret. Il prend les valeurs binaires 0 et 1 en entrée et en sortie. Ce modèle a un paramètre : une probabilité d'erreur p_e. Pour chaque valeur binaire en entrée, le canal produit une valeur binaire en sortie dépendant de cette probabilité d'erreur. La Figure 1.3 illustre ces transformations. Pour ce modèle, la nature des altérations survenant sur le signal émis n'est pas considérée. Seules sont considérées les valeurs du bit à la sortie du codeur de canal et à l'entrée du décodeur associé.

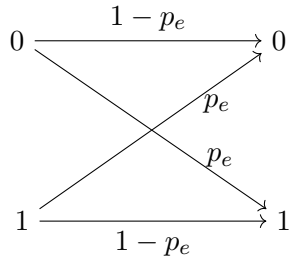


FIGURE 1.3 – Canal binaire symétrique

Si le bit c est émis et que la valeur reçue est $\tilde{c}$, alors :

$$\begin{aligned} \mathbb{P}(\tilde{c}=1|c=0) &= \mathbb{P}(\tilde{c}=0|c=1) = p_e \\ \mathbb{P}(\tilde{c}=1|c=1) &= \mathbb{P}(\tilde{c}=0|c=0) = 1-p_e \end{aligned}$$

Ce canal à entrée et à sortie ferme (uniquement des éléments binaires) permet de modéliser un ensemble de trois blocs de la chaîne de transmission numérique : la modulation, le canal de transmission et la démodulation.

1.1.2.2 Canal à Bruit Blanc Additif Gaussien

Le canal à Bruit Blanc Additif Gaussien, appelé canal BBAG, est régulièrement utilisé pour modéliser les canaux stationnaires. En effet, ce modèle de canal permet de synthétiser les phénomènes physiques intervenant lors de la transmission tout en étant facile à mettre en oeuvre. La Figure 1.4 en donne une représentation. Considérons un échantillon s du signal envoyé à travers le canal. Cet échantillon correspond à un échantillon du signal en sortie du modulateur ($\mathbf{s}$). Les altérations sont modélisées par l'addition d'un bruit blanc gaussien b à l'échantillon émis s. L'échantillon y correspondant du signal reçu est de la forme suivante :

$$y = s + b \tag{1.1}$$

Les échantillons b sont distribués selon une loi normale de moyenne nulle et de variance σ_b^2 : $b \sim \mathcal{N}(0, \sigma_b^2)$. Leur fonction de densité de probabilité est la suivante :

$$p(b) = \frac{1}{\sqrt{2 \cdot \pi \cdot \sigma_b^2}} \exp\left(-\frac{b^2}{2 \cdot \sigma_b^2}\right) \tag{1.2}$$

A partir de cette fonction de densité il est possible de déterminer la probabilité sur la valeur du bit émis au regard de la valeur reçue.

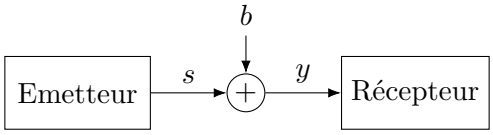


FIGURE 1.4 – Canal à Bruit Blanc Additif Gaussien

Contrairement au canal stationnaire, le canal non-stationnaire permet de prendre en compte les fluctuations de certaines caractéristiques physiques du milieu de propagation. En effet, les mouvements respectifs de l'émetteur et du récepteur entraînent des variations du milieu de propagation qui peuvent générer des erreurs. Dans ce type de canaux, il est possible qu'il y ait des pertes d'énergie (évanouissement) ou encore des trajets multiples (le signal arrive plusieurs fois au récepteur avec différents retards). Dans cette situation le canal BBAG ne suffit plus pour représenter les phénomènes physiques en jeu. Les canaux de type Rice ou Rayleigh peuvent être utilisés pour modéliser ces différents phénomènes. Cela dit, seuls des bits et leurs fiabilités seront utilisés dans les algorithmes présentés dans la suite. Le canal BBAG permet de facilement générer ces valeurs fiabilisées. Nous ne nous intéresserons donc pas à l'utilisation de données issues d'une transmission sur un canal de Rice ou de Rayleigh.

1.1.3 Codage de canal

Comme vu précédemment, les caractéristiques physiques du canal de transmission peuvent avoir un impact néfaste. Le message reçu peut être erroné ce qui rend la communication impossible. Dans ces conditions, il est d'usage de coder l'information côté émetteur avant de l'envoyer à travers le canal. Cette opération est réalisée par le codage de canal. Le récepteur pourra décoder et corriger le message si il n'est pas entaché de trop d'erreurs. En 1948, Claude Shannon énonce un «théorème de codage» dans [80]. Il établit que pour tout canal sans mémoire d'une capacité donnée, il est possible de coder l'information de telle manière que la probabilité d'erreur soit aussi faible que souhaité après décodage. Le codage de canal a donc un rôle majeur dans les systèmes de communication puisqu'il peut augmenter la fiabilité de la transmission. En effet, il

permet de prévenir les éventuelles altérations sur le signal émis dues au canal de transmission. Ces altérations peuvent avoir un effet direct sur la valeur des symboles reçus et rendre l'accès au message original impossible. Pour palier à cela, le codage de canal est composé de un ou plusieurs code(s) correcteur(s) d'erreurs. L'idée de base d'un code correcteur d'erreurs est d'ajouter de la *redondance* de manière contrôlée au message transmis. Ainsi, à partir d'un *mot d'information* comportant le message utile à transmettre, noté $\mathbf{m}$, un *mot de code*, noté $\mathbf{c}$, est créé par l'ajout de cette redondance. Elle est ajoutée au message en appliquant une règle de codage. Connaissant cette règle, le bloc de décodage peut détecter et/ou corriger les éventuelles erreurs dans les mots de code reçus. L'ajout de cette redondance permet d'étendre la même quantité d'information à transmettre sur un plus grand nombre de symboles. Ainsi, si une erreur survient, il est plus aisé de la détecter, voire de la corriger, étant donnée la relation qui existe entre les symboles codés. Les performances d'un code sont mesurées par sa capacité à détecter les erreurs et par sa capacité à les corriger.

Dans un contexte de communication en temps réel, le débit utile est une contrainte non négligeable. Un objectif est donc de trouver un code permettant de maximiser ce débit au regard des conditions de communication tout en ayant de bonnes propriétés de détection et de correction d'erreurs. D'après le théorème de codage de Shannon, il faut pour cela que le rendement du code choisi soit inférieur à la capacité du canal. La capacité du canal correspond au débit théorique maximal d'information utile qui peut être atteint sur ce canal pour une quantité de bruit donnée. Tout l'objectif est donc d'atteindre cette capacité en choisissant le code adéquat. En règle générale, plus on ajoute de redondance, plus les capacités de correction et de détection des erreurs sont élevées. Mais en contrepartie, ces ajouts réduisent le débit de la communication. Une solution est de créer des codes longs mais la complexité du décodage va en augmentant. Il faut donc trouver un compromis entre efficacité et robustesse en prenant en compte l'implémentation de l'ensemble codeur-décodeur.

Il existe différentes familles de codes, les codes en blocs, les codes convolutifs et les codes concaténés. Dans la suite nous donnerons une définition de ces différentes familles de codes. Nous restreindrons la présentation aux codes binaires dont les mots sont composés de symboles dans $\mathrm{GF}(2) = \{0, 1\}$ (GF pour Galois Field). Pour l'ensemble de ces familles de codes, nous utiliserons les notations suivantes :

- k_c : nombre de symboles en entrée du codeur
- n_c : nombre de symboles en sortie du codeur
- $r_c = k_c/n_c$ rendement du code
- $\mathbf{m} = [m(0), \cdots, m(k_c - 1)]$: un mot d'information de longueur k_c
- $\mathbf{c} = [c(0), \cdots, c(n_c - 1)]$: un mot de code de longueur n_c

1.1.3.1 Codes en bloc linéaires

Une première approche est de coder les symboles par bloc de sorte qu'à k_c symboles d'information soient associés n_c symboles codés, avec $n_c > k_c$. Notons alors $\mathbf{G} \in \mathrm{GF}(2)^{k_c \times n_c}$ la *matrice*

génératrice de rang k_c qui engendre l'espace codé $\mathcal{C}(n_c, k_c)$. Alors à chaque mot d'information $\mathbf{m}$ est associé un mot de code $\mathbf{c} = [c(0), c(1), \cdots, c(n_c - 1)]$ à travers l'opération matricielle suivante :

$$\mathbf{c} = \mathbf{m} \cdot \mathbf{G} \tag{1.3}$$

Les lignes de la matrice $\mathbf{G}$ forment une base du code $\mathcal{C}(n_c, k_c)$. Les mots de code $\mathbf{c}$ générés appartiennent à l'espace codé $\mathcal{C}(n_c, k_c)$ et sont tels que $\mathbf{c} \in \mathrm{GF}(2)^{n_c}$. Par conséquent, $\mathcal{C}(n_c, k_c) \subset \mathrm{GF}(2)^{n_c}$ et au regard de l'équation 1.3 le cardinal de $\mathcal{C}(n_c, k_c)$ est tel que : $\mathrm{Card}(\mathcal{C}(n_c, k_c)) = 2^{k_c}$. En effet, l'application suivante est une bijection :

$$\begin{array}{rcl} \mathrm{GF}(2)^{k_c} & \rightarrow & \mathcal{C}(n_c, k_c) \\ \mathbf{m} & \longmapsto & \mathbf{m} \cdot \mathbf{G} \end{array} \tag{1.4}$$

Cette particularité soulève la question de la distribution des mots de code dans cet espace codé. Il est possible d'avoir un aperçu de cette distribution en mesurant la distance entre les mots de code deux à deux grâce à la *distance de Hamming*.

Définition 1. *Soit $\mathbf{c}_1$ et $\mathbf{c}_2$ deux mots de même longueur composés de symboles prenant leurs valeurs dans un corps de Galois. La distance de Hamming entre $\mathbf{c}_1$ et $\mathbf{c}_2$, notée $d_H(\mathbf{c}_1, \mathbf{c}_2)$, correspond au nombre de symboles dont diffèrent les deux mots composante à composante.*

La distribution des distances dans un espace codé est un indicateur de l'efficacité d'un code correcteur d'erreurs. Cette mesure se fait par la *distance minimale*. Cette distance indique que tous les mots de cet espace diffèrent d'un minimum de symboles.

Définition 2. *La distance minimale (au sens de Hamming) correspond à la plus petite distance de Hamming qui existe entre entre ses mots de code :*

$$d_{min}(\mathcal{C}) = \min_{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}, \mathbf{c}_1 \neq \mathbf{c}_2} d_H(\mathbf{c}_1, \mathbf{c}_2)$$

A partir de la distance minimale il est possible de quantifier l'efficacité du codage en terme de capacité de détection et de correction d'erreurs. Quand la distance minimale est grande, il est plus facile de discriminer les mots de code, et ce, même si ils contiennent des erreurs.

Définition 3. *La capacité de détection correspond au nombre d'erreurs maximal qui peuvent être détectées pour un code $\mathcal{C}$ et elle vaut $d_{min}(\mathcal{C}) - 1$.*

Définition 4. *La capacité de correction correspond au nombre d'erreurs maximal qui peuvent être corrigées pour un code $\mathcal{C}$ et elle vaut $\lfloor \frac{d_{min}(\mathcal{C})-1}{2} \rfloor$.*

Deux codes $\mathcal{C}$ et $\mathcal{C}'$ sont équivalents si ils génèrent le même espace codé et qu'il existe une matrice inversible $\mathbf{P}$ telle que : $\mathbf{G} = \mathbf{G}'.\mathbf{P}$. Dans ce cas, la distribution des distances est la même et les codes ont la même distance minimale. Par conséquent, il auront aussi les mêmes capacités de correction et de détection des erreurs. L'Exemple 1.1.1. illustre l'opération de codage équivalent.

Exemple 1.1.1.

Prenons l'exemple du code à répétition qui consiste en la répétition de chaque bit d'information pour obtenir un mot de code. Considérons des mots d'information $\mathbf{m}$ de longueur 2. Chaque bit est répété 3 fois pour obtenir le mot de code $\mathbf{c}$:

$$\begin{aligned}
\mathbf{m} &= [0\ 0] \rightarrow \mathbf{c} = [0\ 0\ 0\ 0\ 0\ 0] \\
\mathbf{m} &= [0\ 1] \rightarrow \mathbf{c} = [0\ 0\ 0\ 1\ 1\ 1] \\
\mathbf{m} &= [1\ 0] \rightarrow \mathbf{c} = [1\ 1\ 1\ 0\ 0\ 0] \\
\mathbf{m} &= [1\ 1] \rightarrow \mathbf{c} = [1\ 1\ 1\ 1\ 1\ 1]
\end{aligned}$$

Ce code est de longueur $n_c = 6$, de dimension $k_c = 2$ ($\mathcal{C}(6, 2)$) et donc de rendement $\frac{1}{3}$. La matrice génératrice correspondante est la suivante :

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Cette matrice génératrice permet de générer l'ensemble des quatre mots de code définis ci-dessus ($\mathbf{c} = \mathbf{m} \cdot \mathbf{G}$). En lui appliquant la matrice :

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

on obtient la matrice génératrice suivante permettant de générer un code équivalent $\mathcal{C}'(6, 2)$:

$$\mathbf{G}' = \mathbf{G} \cdot \mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Le codage résultant $\mathbf{c}' = \mathbf{m} \cdot \mathbf{G}'$ est le suivant :

$$\begin{aligned} \mathbf{m} &= [0\ 0] \rightarrow \mathbf{c}' = [0\ 0\ 0\ 0\ 0\ 0] \\ \mathbf{m} &= [0\ 1] \rightarrow \mathbf{c}' = [1\ 1\ 1\ 0\ 0\ 0] \\ \mathbf{m} &= [1\ 0] \rightarrow \mathbf{c}' = [0\ 0\ 0\ 1\ 1\ 1] \\ \mathbf{m} &= [1\ 1] \rightarrow \mathbf{c}' = [1\ 1\ 1\ 1\ 1\ 1] \end{aligned}$$

L'espace codé obtenu est exactement le même pour $\mathcal{C}$ et $\mathcal{C}$'. Les deux codes sont donc équivalent.

Pour tout code linéaire, il est possible de trouver un code équivalent particulier : le *code systématique*. Le codage systématique est très utilisé car les étapes de codage et de décodage sont peu complexes.

Définition 5. *La matrice génératrice d'un code $\mathcal{C}(n_c, k_c)$ est dite systématique si ses k_c premières (ou dernières) colonnes forment la matrice identité $\mathbf{I}_{k_c}$, i.e. les k_c premiers (ou derniers) symboles dans chaque mot de code correspondent au mot d'information dont il est issu.*

Pour ce type de codage, à chaque mot de k_c bits on ajoute $(n_c - k_c)$ bits de redondance résultant chacun d'une combinaison linéaire des k_c bits d'information. L'Exemple 1.1.2. illustre ce type de codage.

Exemple 1.1.2.

Considérons un $\mathcal{C}(3, 2)$ code de rendement $\frac{2}{3}$ qui ajoute un bit de redondance à des mots d'information de 2 bits. La matrice génératrice systématique de ce code est la suivante :

$$\mathbf{G}_{sys} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

et les mots de code générés, $\mathbf{c} = \mathbf{m} \cdot \mathbf{G}_{sys}$, sont les suivants :

$$\begin{aligned} \mathbf{m} &= [0\ 0] \rightarrow \mathbf{c} = [0\ 0\ 0] \\ \mathbf{m} &= [0\ 1] \rightarrow \mathbf{c} = [0\ 1\ 1] \\ \mathbf{m} &= [1\ 0] \rightarrow \mathbf{c} = [1\ 0\ 1] \\ \mathbf{m} &= [1\ 1] \rightarrow \mathbf{c} = [1\ 1\ 0] \end{aligned}$$

Dans l'Exemple 1.1.1., le $\mathcal{C}(6, 2)$ code de rendement $\frac{1}{3}$ a une distance minimale de 3. D'après les définitions 3 et 4, ce code est capable de détecter au maximum 2 erreurs et d'en corriger 1 sur des mots de code de taille 6. Comparativement, le $\mathcal{C}(3, 2)$ code de l'Exemple 1.1.2. de rendement $\frac{2}{3}$ a une distance minimale de 2. Il est capable de détecter 1 erreur sur des mots de

code de taille 3 et de n'en corriger aucune. Ces deux codes sont équivalents en terme de capacité de détection d'erreurs mais le code à répétition ($\mathcal{C}(6,2)$) permet une meilleure correction des erreurs. Toutefois, le code de l'Exemple 1.1.2. sera plus efficace en terme de débit utile puisqu'il a un rendement plus élevé. Si le canal n'occasionne pas trop d'erreurs, il peut être plus intéressant d'utiliser ce code malgré son manque de robustesse.

Le choix d'un code dépend donc de l'application et du besoin en correction et/ou en détection d'erreurs. Ce choix dépend aussi de contraintes telles que la complexité du codage et de celle du décodage. Pour le codage par bloc, il est fréquent que l'étape de décodage utilise le *code dual* du code utilisé.

Définition 6. *Un code $\mathcal{C}^T$ est le dual du $\mathcal{C}(n_c, k_c)$ code ssi :*

$$\forall \mathbf{x} \in \mathcal{C}, \forall \mathbf{y} \in \mathcal{C}^T \begin{cases} \mathbf{x} \cdot \mathbf{y}^T = \mathbf{0} \\ \mathbf{y} \cdot \mathbf{x}^T = \mathbf{0} \end{cases} \tag{1.5}$$

$\mathcal{C}^T$ est un espace de dimension $(n_c - k_c)$. Le code $\mathcal{C}^T$ est orthogonal à $\mathcal{C}$.

On note $\{\mathbf{h}_0, \mathbf{h}_1, \cdots, \mathbf{h}_{n_c-k_c-1}\}$ une base de $\mathcal{C}^T$ et nous formons la matrice $\mathbf{H}$ tel que ses lignes sont les éléments de cette base :

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n_c-k_c-1} \end{bmatrix}$$

$\mathbf{H}$ est une matrice génératrice du code dual $\mathcal{C}^T$. Cette matrice est appelée *matrice de parité* du code $\mathcal{C}$. Les mots de $\mathcal{C}^T$ sont les relations de parité du code $\mathcal{C}$.

Proposition 1. *Soit $\mathbf{G}$ et $\mathbf{H}$ les matrices génératrices et de parité du code $\mathcal{C}$ respectivement alors :*

$$\mathbf{G}.\mathbf{H}^T = \mathbf{0} \tag{1.6}$$

Par voie de conséquence, d'après la définition 6, pour tout mot de code $\mathbf{c}$ de $\mathcal{C}$:

$$\mathbf{c}.\mathbf{H}^T = \mathbf{0} \tag{1.7}$$

La matrice de parité du code $\mathcal{C}$ n'est pas unique. Étant elle-même la matrice génératrice du code dual $\mathcal{C}^T$, il existe plusieurs matrice de parité engendrant le même code dual. Ces matrices de parité permettent la détection voire la correction d'erreurs par contrôle de parité. La détection et la correction d'une erreur peut être effectuée à l'aide du *syndrome* $\mathbf{s}$ de cette erreur. Si un

mot de code bruité $\tilde{\mathbf{c}}$ est reçu, alors le syndrome de l'erreur $\mathbf{e}$ est défini comme suit :

$$\begin{aligned} \mathbf{s} &= \tilde{\mathbf{c}}.\mathbf{H}^T \\ &= (\mathbf{c}+\mathbf{e}).\mathbf{H}^T \\ &= \mathbf{c}.\mathbf{H}^T + \mathbf{e}.\mathbf{H}^T \\ &= \mathbf{e}.\mathbf{H}^T \end{aligned} \tag{1.8}$$

Si le syndrome n'est pas nul, alors le mot reçu est entaché d'erreurs. De plus, si le poids de $\mathbf{e}$ est inférieur à la capacité de correction du code, la valeur du syndrome renseigne sur la position des erreurs et il est alors possible de les corriger. L'Exemple 1.1.3. permet d'illustrer le contrôle de la présence d'erreurs par parité.

Exemple 1.1.3.

Un exemple très connu de codage détecteur/correcteur d'erreurs est le code de Hamming de paramètres $n_c = 7$ et $k_c = 4$. Ce code a une distance minimale de 3 : ses capacités de détection et de correction d'erreurs sont donc respectivement 2 et 1. Voici une matrice génératrice du code :

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Cette matrice génératrice étant systématique, chaque mot de code est la concaténation de 4 bits d'information et de 3 bits de parité. Si le mot d'information $\mathbf{m} = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}$ est codé grâce à $\mathbf{G}$, on obtient le mot de code suivant :

$$\mathbf{c} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Une matrice de parité de ce code est :

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Cette matrice vérifie bien les équations 1.5 et 1.7. Supposons maintenant qu'une erreur soit survenue sur le 6-ième bit de $\mathbf{c}$ ($\mathbf{e} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$) :

$$\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

La distance minimale de ce code étant de 3, l'erreur est détectée car $\tilde{\mathbf{c}}.\mathbf{H}^T \neq \mathbf{0}$. La valeur du syndrome pour ce motif d'erreur est :

$$\mathbf{s} = \tilde{\mathbf{c}}.\mathbf{H}^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Ce code étant capable de corriger 1 erreur par mot de code, le calcul du syndrome va également permettre de corriger l'erreur intervenu sur le bit 6 du mot de code.

Avec l'idée de toujours améliorer le pouvoir de correction des codes, les soixante-dix dernières années ont vu naître divers méthodes de codage en blocs comme par exemple les codes LDPC [36], les codes Reed-Solomon [77], les codes Reed-Muller [69, 76] ou les codes BCH [11, 42]. Les codes BCH et Reed-Solomon sont des codes cycliques dont les méthodes de codage et de décodage tirent parti de la représentation polynomiale. Cette représentation permet de réduire la complexité de ces opérations en utilisant le décodage d'interpolation Berlekamp-Welch [106] par exemple. Les codes Reed-Solomon sont très utilisés dans les systèmes de stockages de données pour la protection des fichiers contres les erreurs ou contre l'effacement (dans les CD par exemple). Dans l'idée de réduire la complexité du décodage, les codes LDPC (Low Density Parity Check) sont caractérisés par une matrice de parité creuse. En effet, les relations de parité utilisées pour le décodage sont de poids faibles afin de minimiser la quantité de calculs nécessaires. Par leur structure, les codes LDPC sont propices à l'utilisation d'algorithmes de décodage itératifs. Étant donné les relations entre les bits dans les mots de code (relations de parité), il est possible de faire passer des messages sur la valeur des bits à travers un *graphe de Tanner* [93]. Initialement, cet algorithme par passage de message permet le renversement de bit : il opère sur le canal binaire symétrique. Toutefois, d'autres types d'algorithmes à passage de messages existent : les algorithmes à propagation de croyance pouvant opérer sur le canal BBAG. Dans ce cas, les valeurs transmises sont des probabilités. Tous les noeuds du graphe échangent des probabilités sur les valeurs a priori et a posteriori des bits reçus pendant un certain nombre d'itérations ou jusqu'à convergence pour prendre une décision sur ces valeurs. Un exemple de ce genre d'algorithmes itératifs est l'algorithme BCJR [1] (Bahl, Cocke, Jelinek et Raviv).

Le codage canal peut utiliser des symboles non-binaires comme cela peut être le cas pour les codes Reed-Solomon par exemple. Toutefois, comme dit précédemment, nous traiterons exclusivement des codes binaires dans la suite de notre propos.

1.1.3.2 Codes convolutifs

Les codes convolutifs sont un autre type de codage linéaire que l'on oppose souvent au codage en blocs. Ils ont été introduit par Elias [72] en 1955. Dans le cas des codes convolutifs, le codage s'effectue en flux, ce qui permet la création de mots de code de très grande taille (même infinie théoriquement). Ceci permet de s'affranchir de la limite qu'impose les codes en blocs pour des longueurs trop élevées. En effet, le décodage est moins complexe et aussi efficace que pour les codes en blocs. Comme pour le codage en bloc, à un mot d'information est associé un mot de

code. De la même manière que pour les codes en blocs, n_c bits codés sont générés à partir de k_c bits d'information. Cependant, la particularité du codage convolutif est qu'un effet de mémoire se produit. En effet, si nous appelons K_c la *longueur de contrainte* du code, alors un mot de code dépend d'un mot d'information et des $(K_c - 1)$ mots d'information précédents. Un code convolutif est donc représenté par ses 3 paramètres et nous le noterons $\mathcal{C}(n_c, k_c, K_c)$.

Pour faire état de cet effet de mémoire, il est possible d'utiliser une représentation polynomiale de l'opération de codage. Soit un $\mathcal{C}(n_c, k_c, K_c)$ code convolutif. Ce code possède $n_c \cdot k_c$ polynômes générateurs appartenant à GF(2)$[D]$ (ensemble des polynômes d'indéterminée D à coefficients binaires) de degré au plus K_c. Soit $g_{i,j}(D)$ un de ces polynômes générateurs :

$$\forall i \in [\![1,\, k_c]\!],\ \forall j \in [\![1,\, n_c]\!],\ g_{i,j}(D) = \sum_{t=0}^{L} g_{i,j}(t) \cdot D^t \tag{1.9}$$

où $g_{i,j}(t)$ est le t-ième coefficient du polynôme $g_{i,j}(D)$ et L est un entier inférieur ou égal à K_c. $\mathbf{g}_{i,j}$ est donc un vecteur binaire de longueur K_c :

$$\mathbf{g}_{i,j} = [g_{i,j}(0)\, g_{i,j}(1)\, \cdots\, g_{i,j}(K_c - 1)] \tag{1.10}$$

Étant donné que la transformation en mots de code repose sur des combinaisons linéaires de bits d'information, le codage convolutif peut être vu comme un codage en bloc de longueur infinie. A partir des vecteurs binaires $\mathbf{g}_{i,j}$ nous pouvons créer la matrice génératrice $\mathbf{G}$ de taille infinie :

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K_c-1} & & & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K_c-1} & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K_c-1} & \\ & & & & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \tag{1.11}$$

où $\forall l \in [\![0,\, K_c - 1]\!]$ $\mathbf{G}_l$ est une matrice de taille $(k_c \times n_c)$ de la forme :

$$\mathbf{G}_l = \begin{pmatrix} \mathbf{g}_{1,1}(l) & \cdots & \mathbf{g}_{1,n_c}(l) \\ \vdots & \ldots & \vdots \\ \mathbf{g}_{k_c,1}(l) & \cdots & \mathbf{g}_{k_c,n_c}(l) \end{pmatrix} \tag{1.12}$$

Dans $\mathbf{G}$, les parties vides sont des zéros. Cette écriture n'est toutefois pas très concise, un autre moyen de décrire le codage convolutif passe par la matrice polynomiale de taille $(k_c \times n_c)$ suivante :

$$\mathbf{G}(D) = \begin{pmatrix} g_{1,1}(D) & \cdots & g_{1,n_c}(D) \\ \vdots & \ldots & \vdots \\ g_{k_c,1}(D) & \cdots & g_{k_c,n_c}(D) \end{pmatrix} \tag{1.13}$$

Comme dit précédemment, la description d'un code convolutif comme un code en bloc est une représentation très intuitive de l'opération de codage mais elle est peu synthétique. Elle ne rend pas forcément compte de la notion de mémoire utilisée pour ajouter de la redondance. Une autre manière de représenter et d'effectuer le codage convolutif est l'utilisation d'un *registre à décalage* de longueur K_c. Les K_c bascules du registre sont initialisés à 0 et à chaque cycle d'horloge, k_c bits informations entrent dans le codeur et n_c bits en sortent. A chaque cycle, les k_c bits d'information se décalent d'un cran dans les bascules et n_c nouveaux bits codés sortent du registre. Ces n_c bits dépendent des k_c derniers bits entrés dans le registre ainsi que des $(K_c - 1) \cdot k_c$ bits entrés lors des $(K_c - 1)$ cycles d'horloge précédents.

Exemple 1.1.4.

Soit un $\mathcal{C}(2, 1, 3)$ code convolutif de rendement $\frac{1}{2}$ et de longueur de contrainte $K_c = 3$. Ce codeur est donc composé de $n_c \cdot k_c = 2$ polynômes générateurs de degrés au plus $K_c - 1 = 2$. Pour cet exemple, prenons les 2 polynômes générateurs suivants : $g_{1,1}(D) = 1 + D^2$ et $g_{1,2}(D) = 1 + D + D^2$. L'implémentation en registre à décalage est décrite par la Figure 1.5. Les deux sorties du codeur sont notés $\mathbf{c}^{(1)}$ et $\mathbf{c}^{(2)}$ respectivement. Elles sont associées aux polynômes générateurs $g_{1,1}(D)$ et $g_{1,2}(D)$, respectivement.

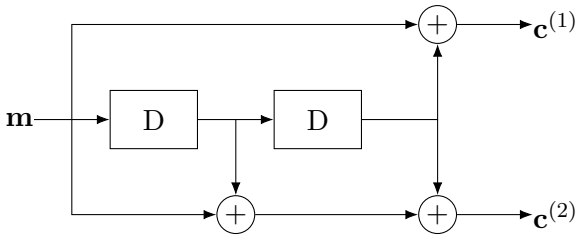


FIGURE 1.5 – Schéma d'implémentation sous forme de registre à décalage du codeur de matrice génératrice $\mathbf{G}(D) = (1 + D^2 \;\; 1 + D + D^2)$

Codons le mot d'information $\mathbf{m} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$. En écrivant $\mathbf{m}$ sous sa forme polynomiale, on obtient :

$$m(D) = 1 + D + D^3$$

La représentation polynomiale de l'opération de codage est la suivante :

$$\begin{aligned} c_1(D) &= m(D) \cdot g_{1,1}(D) = 1 + D + D^2 + D^5 \\ c_2(D) &= m(D) \cdot g_{1,2}(D) = 1 + D^4 + D^5 \end{aligned}$$

Ce qui correspond à l'écriture binaire suivante :

$$\begin{bmatrix} \mathbf{c}^{(1)} \\ \mathbf{c}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Le mot de code obtenu correspond à la concaténation des deux sorties du codeur :

$$\begin{aligned}\mathbf{c} &= \begin{bmatrix} c^{(1)}(0) & c^{(2)}(0) & c^{(1)}(1) & c^{(2)}(1) & \cdots & c^{(1)}(5) & c^{(2)}(5) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}\end{aligned} \tag{1.14}$$

Une autre représentation passe par un *diagramme d'état* du registre à décalage. Ce diagramme d'état est un graphe orienté où chaque sommet représente un état interne du registre à décalage et chaque arête est une transition d'un état vers un autre. L'état interne du registre correspond à l'ensemble des valeurs binaires à la sortie de chaque bascule. A chaque cycle d'horloge un changement d'état se produit : la transition dépend des k_c nouvelles valeurs à l'entrée du codeur. Il y a donc jusqu'à 2^{k_c} arêtes qui partent de chaque sommet. La présence d'une arête dépend de la possibilité de passer d'un état à un autre. Chaque arête est étiquetée par l'entrée $\mathbf{m}(t)$ permettant la transition, et par la sortie $\mathbf{c}(t)$. $\mathbf{m}(t)$ est un vecteur binaire contenant l'ensemble des k_c bits à l'entrée du codeur à l'instant t. De la même manière, $\mathbf{c}(t)$ est un vecteur binaire contenant l'ensemble des n_c bits générés par le codeur à l'instant t. Les étiquettes sont de la forme : $(\mathbf{m}(t), \mathbf{c}(t))$.

Exemple 1.1.5.

Reprenons le codeur de l'Exemple 1.1.4.. La Figure 1.6 est la diagramme d'état associé à ce codeur. Nous allons réaliser le codage de la séquence d'information suivante : $\mathbf{m} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$. En partant de l'état initial (00), le premier bit d'information en entrée est 1. Le codeur produit la sortie (11) (étiquette $(1, 11)$) et il passe de l'état (00) à l'état (10). Au final, nous obtenons la séquence codée suivante :

$$\mathbf{c} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

qui correspond bien au résultat obtenue précédemment.

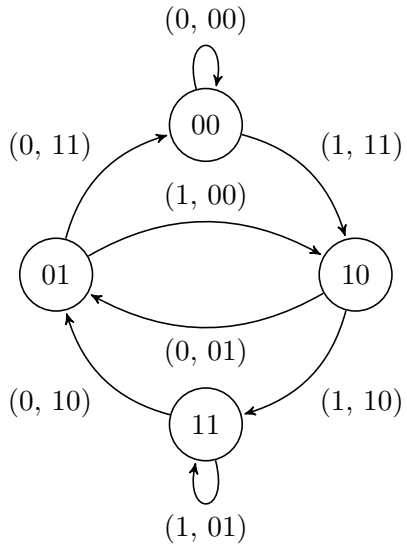


FIGURE 1.6 – Diagramme d'état du codeur de la Figure 1.5

Finalement, un autre type de graphe permet d'effectuer l'opération de codage : le *treillis*. Les différents états que peut prendre le registre sont représentés par des noeuds, et les transitions d'un état à l'autre le sont par des branches. Le codage s'effectue en suivant un chemin de gauche à droite sur le treillis. L'étiquetage des branches du treillis s'effectue comme pour les arêtes du diagramme d'état. Une fois le codage effectué, il est possible de lire le mot de code en suivant le chemin de codage. A l'instar du diagramme d'état, le mot de code se lit en concaténant les sorties du codeur inscrites sur chaque branche (deuxième composante). Les cycles d'horloge sont signalés au-dessus de chaque colonne de noeuds. La valeur des bits dans le mot d'information détermine les branches empruntées pour créer un chemin dans le treillis de gauche à droite.

Exemple 1.1.6.

La Figure 1.7 donne un exemple de treillis pour le code convolutif de rendement $\frac{1}{2}$ utilisé dans les exemples précédents. Le chemin dans le treillis pour le codage de la séquence de bits d'information de l'Exemple 1.1.5. est en pointillés pour les premiers cycles d'horloge. Comme précédemment l'état du codeur au deuxième cycle est marqué en rouge.

Le treillis est aussi un outil pour le décodage des codes convolutifs. L'algorithme de décodage le plus connu est dû à Viterbi [100]. Pour décoder, la méthode consiste à rechercher le message le plus probable à l'aide du treillis. Les arêtes du treillis sont pondérées par leur distance de Hamming au mot de code reçu, puis on décode en y recherchant le chemin le plus court.

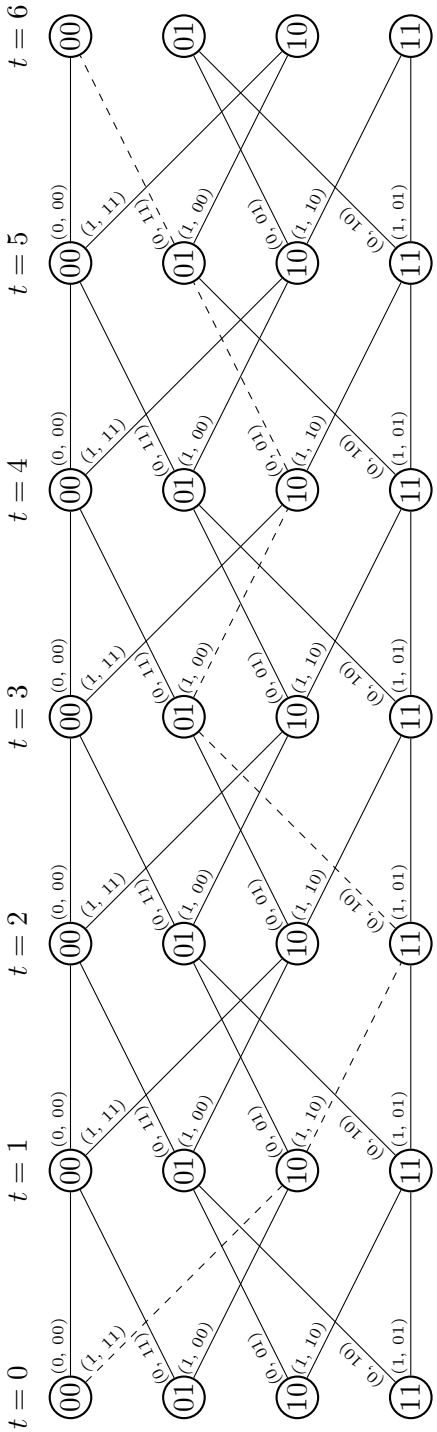


FIGURE 1.7 – Treillis du codeur de la Figure 1.5

1.1.3.3 Entrelacement

Lors d'une transmission, il est rare que les erreurs apparaissent de manière dispersée sur les trames émises. En effet, il est fréquent que les erreurs arrivent en rafale et engendre donc des paquets d'erreurs. Ces paquets d'erreurs peuvent être fatals pour le décodeur. Afin de réduire la probabilité de l'apparition de ces paquets d'erreurs, les mots de code sont la plupart du temps concaténés et les symboles qu'ils contiennent sont permutés selon un schéma d'*entrelacement* avant d'être envoyés. Ainsi, si des erreurs se produisent en rafale, les symboles affectés sont séparés à la réception par un *désentrelaceur* qui inverse la permutation. Les erreurs sont ainsi réparties de manière plus uniforme entre les mots de code, ce qui permet de rendre le décodeur plus robuste aux erreurs en rafale. Tout comme les codes, il existe plusieurs familles d'entrelaceur, les entrelaceurs par bloc et les entrelaceurs convolutifs.

1.1.3.4 Codes poinçonnés

Comme dit précédemment, le théorème de codage de Shannon [80] impose de trouver, pour un niveau de bruit donné, un compromis entre le débit utile et la complexité des étapes de codage et de décodage. Pour répondre à cette problématique, une solution consiste à poinçonner les mots de code. Le poinçonnage a été introduit par Solomon *et al.* au milieu des années 1960 [84,85] pour les codes cycliques. L'opération consiste à retirer certains bits de la trame codée afin d'augmenter le rendement du code et le débit utile par la même occasion. Plus particulièrement, le poinçonnage des codes convolutifs a été proposé par Cain *et al.* en 1979 [15]. Les codes poinçonnés voient leur rendement augmenter tout en conservant des performances de décodage proches de celles du code non poinçonné. L'opération de poinçonnage est réalisée selon un motif de poinçonnage qui est appliqué sur la trame codée. Le décodeur connaît la position des bits poinçonnés grâce à ce motif de poinçonnage et peut le prendre en compte dans ses opérations. L'Exemple 1.1.7. illustre le principe du poinçonnage pour le code convolutif de la Figure 1.5.

Exemple 1.1.7.

Reprenons le code convolutif $\mathcal{C}(2,1,3)$ de la Figure 1.5. Le codage des mots d'information suivant : $\mathbf{m} = \begin{bmatrix}1 & 0 & 1 & 1 & 0 & 0 & 1 & 1\end{bmatrix}$, nous donne les mots de code suivant (pour les sorties 1 et 2 du codeur) :

$$\begin{bmatrix}\mathbf{c}^{(1)} \\ \mathbf{c}^{(2)}\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0\end{bmatrix}$$

Considérons le motif de poinçonnage suivant :

$$\begin{bmatrix}1 & 0 \\ 1 & 1\end{bmatrix}$$

Les valeurs 0 dans le motif de poinçonnage correspondent aux bits poinçonnée (non transmis). En l'appliquant sur la trame codée, on obtient :

$$\begin{bmatrix} 1 & X & 0 & X & 0 & X & 1 & X \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Ainsi, le codage équivalent obtenu est de rendement $\frac{2}{3}$.

1.1.3.5 Codes concaténés

Une autre solution permettant cette fois d'améliorer la capacité de correction des codes réside dans l'utilisation des codes concaténés. En 1965, David Forney établit que les codes concaténés peuvent permettre de diminuer la probabilité d'erreur en augmentant la longueur du code pour un décodage aussi complexe [35]. L'opération consiste alors à coder l'information avec un premier code en bloc, (un code Reed-Solomon dans les travaux exposés par Forney), puis à coder la trame codée par un second code pour que le tout forme un *super codeur*. La Figure 1.8 donne le modèle de la transmission dans son acception originale. La structure est composée d'un codeur interne et d'un codeur externe en série à l'émission (le super codeur) et des décodeurs associés en série à la réception (le super décodeur). Du point point de vue de l'ensemble codeur-décodeur externe, l'ensemble canal et codeur-décodeur interne forme un *super canal* avec une probabilité d'erreur donné. Un exemple d'utilisation est la concaténation d'un code de Reed-Solomon en codeur externe avec un code convolutif en codeur interne dans un modem ADSL. Entre les deux codeurs est souvent placé un entrelaceur pour éviter le problème des erreurs en rafale.

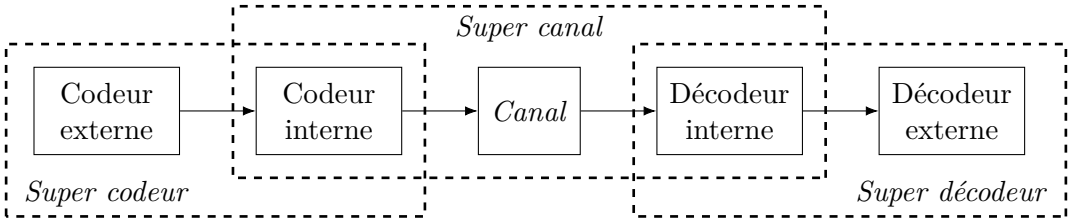


FIGURE 1.8 – Modèle de Forney d'une transmission utilisant des codes concaténés

Le modèle de la Figure 1.8 décrit la concaténation en série de codes correcteurs. En 1993, Berrou *et al.* proposent de concaténer deux codes convolutifs en parallèle : le *Turbo-code* [6]. La Figure 1.9 décrit le schéma de codage d'un tel codeur. Le Codeur 1 code le message informatif **m** et le codeur 2 code une version entrelacée du message informatif **m**. En règle générale, les deux codeurs sont identiques et sont systématiques. Dans ce cas, les sorties systématiques du second codeur sont poinçonnées, comme illustré par la Figure 1.9 pour ne conserver que la partie redondante générée par le codeur. Le Turbo-code est très prisé pour son codage et son décodage rapide. Il connaît de multiples applications : le téléphonie mobile, l'ESA et la NASA à bord de leurs sondes spatiales ou encore dans les standard de transmission pour la télévision haute définition.

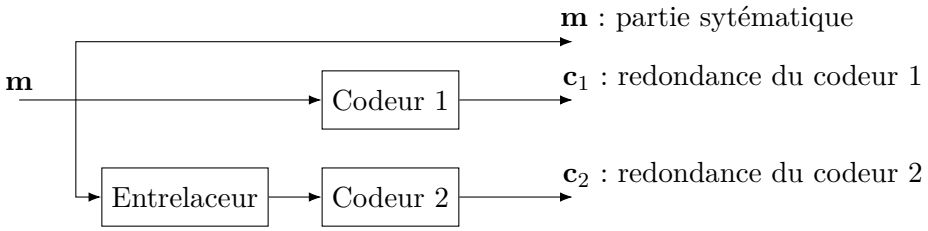


FIGURE 1.9 – Schéma de codage d'un Turbo-code parallèle

1.2 Identification des paramètres de codes correcteurs d'erreurs

Dans cette section, nous nous intéressons à la détection d'un code d'une part, et à l'identification de ses paramètres (taille des mots d'information et taille des mots de code) d'autre part. La détection consiste simplement à déterminer si un code est présent. Dans le contexte semi-coopératif, la détection et l'identification du code consiste à choisir parmi un catalogue le code utilisé par l'émetteur pour coder la trame reçue. Par contre, dans le contexte non-coopératif, il faut établir la présence d'un code et identifier ses paramètres à l'aide d'algorithmes. La détection et l'identification des codes peut se faire en utilisant des informations fermes (de l'information binaire : des 0 et des 1) si une décision a été prise en sortie du démodulateur ou des informations souples (valeurs réelles) tenant compte du niveau de fiabilité sur les bits.

1.2.1 Contexte semi-coopératif : identification sur catalogue

Le catalogue se définit comme un ensemble fini de codes correcteurs d'erreurs. La plupart du temps, les codes issus de ce catalogue ont des propriétés différentes (comme le rendement par exemple). Dans le cadre d'une communication semi-coopérative, il est envisageable qu'un émetteur s'accorde sur la teneur du catalogue avec le/les récepteurs. Dans ce cas, l'émetteur choisit un code dans la liste et le récepteur détermine lequel a été utilisé à partir de la trame reçue. Ces approches sont particulièrement bien adaptées au contexte de l'Adaptive Modulation & Coding (AMC), où l'émetteur choisit ses paramètres de codage et de modulation en fonction de la qualité du canal de transmission. Par exemple, si le canal de transmission n'occasionne pas trop d'erreurs, il sera plus judicieux de choisir un code de rendement élevé afin d'augmenter le débit utile de la transmission. Le récepteur doit alors être en mesure de suivre les modifications de paramètres de l'émetteur. Le choix du code est fait parmi un ensemble fini et de faible cardinalité pour minimiser le coût en calcul. Généralement, les méthodes à base de catalogue ont l'avantage d'être peu coûteuses, rapides et très performantes. Toutefois elles sont limitées à la reconnaissance des codes présents dans un catalogue.

Le catalogue est constitué des matrices de parité des différents codes que l'émetteur a à sa disposition pour communiquer. Pour le récepteur, une première approche consisterait à calculer les syndromes à partir des bits reçus. Le code détecté serait celui pour lequel le plus grand nombre de relations de parité est respecté. En pratique, prendre une décision sur l'information

reçue peut provoquer une perte d'information. Pour éviter cet écueil, l'utilisation d'informations souples est plus efficace. Dans ce cas, la fiabilité des bits est conservée pour identifier le codeur. Dans [68, 107–111], les auteurs proposent des méthodes utilisant ces fiabilités. Ces méthodes à base de catalogue sont basées sur le calcul d'une probabilité a posteriori sur la valeur du syndrome. Dans la suite nous appellerons cette probabilité *SPP* (pour *Syndrome a Posteriori Probability*). La SSP correspond donc à la probabilité que le syndrome soit nul connaissant la trame reçue. Dans [68], à partir de la séquence reçue, des fiabilités des bits et pour chaque matrice de parité du catalogue de codes, l'algorithme calcule une SPP. L'identification peut être faite de deux manières :

- une recherche exhaustive sur l'ensemble du catalogue. Le code sélectionné sera celui qui obtient la SPP la plus élevée.
- une approche sous-optimale : dès que la SPP est supérieure à un seuil, on arrête la recherche. On considère que le code utilisé par l'émetteur est celui correspondant à la matrice de parité ayant obtenue une SPP supérieure au seuil de décision.

Dans ces conditions, il est donc nécessaire que le récepteur connaisse les matrices de parité des codes et qu'il soit synchronisé sur la trame (c'est-à-dire que la trame commence par le premier bit d'un mot de code). Il est possible de résoudre le problème de la synchronisation de trame conjointement à l'identification comme proposé dans [110]. Toutefois, l'estimation demande une recherche exhaustive pour tous les codes et toutes les positions de synchronisation possibles. Afin de réduire la complexité, les auteurs proposent un algorithme en deux étapes pour la synchronisation : un balayage grossier sur les valeurs de synchronisation suivi d'une recherche fine. Ceci est possible car la valeur moyenne des SPP est plus grande autour de la valeur de synchronisation. Par ce biais, la première phase de recherche permet de mettre en évidence la zone de synchronisation. Ensuite, la deuxième phase permet d'estimer la valeur de synchronisation.

Il est facile d'utiliser la fiabilité des bits reçus pour évaluer la SPP. L'utilisation de cette information supplémentaire octroie un gain de performances face au bruit sans pour autant augmenter significativement la complexité calculatoire. Un autre avantage est que cette méthode peut être mise en oeuvre pour toutes les familles de code à partir du moment où l'on dispose d'une matrice de parité.

1.2.2 Contexte non-coopératif : identification aveugle

Contrairement au contexte semi-coopératif, l'a priori sur le code utilisé n'est plus présent pour une communication dans un cadre non-coopérative. L'information que le récepteur a à sa disposition est souvent restreinte à la mesure qu'il peut faire dans son environnement. Dans les méthodes décrites ci-dessous, la trame reçue est traitées après une prise de décision, on parle alors d'information ferme (0 et 1) et, dans certains cas, la fiabilité des bits peut être prise en compte, on parle alors d'information souple. La détection et l'identification des paramètres des codes est rendus possible grâce à l'ajout d'information redondante lors du codage. Cette

particularité implique que les bits dans un mot de code ne sont pas indépendants. Dans le cas des codes linéaires qui nous intéresse ici, chaque bit codé est obtenu par la combinaison linéaire de bits d'information. Toutes les méthodes décrites ci-dessous sont basés sur cette particularité. De nombreux travaux portent sur cette problématique : pour les codes linéaires de manière générale [20, 25, 97, 98], les codes LDPC [89], les codes convolutifs [34, 62, 87], les codes BCH [92, 104], les codes cycliques [50] et les codes non-binaires en général [122, 124]. Dans la suite nous détaillons particulièrement un ensemble de méthodes basées sur le critère du rang.

1.2.2.1 Méthodes basées sur le critère du rang

Les méthodes basées sur le critère du rang permettent de traiter le problème de l'identification dans son intégralité. En effet, elles nécessitent peu d'hypothèses et permettent d'estimer la taille, le rendement et les relations de parité comme nous le verrons dans la suite. Le premier article faisant mention du principe date de 2003. Dans [14], Burel et Gautier proposent un algorithme permettant d'estimer la taille et le rendement d'un code en bloc dans le cas où la trame reçue n'est pas entachée d'erreurs.

Pour une trame reçue composée de N bits, la méthode consiste a balayer différentes longueurs n pour le code. Pour chaque longueur n testée, la trame reçue est découpée en $L_n = \lfloor \frac{L}{n} \rfloor$ blocs contigus de n bits. Ces blocs sont ensuite disposés pour former les lignes d'une *matrice d'interception* notée $\mathbf{M}^{(n)}$. Cette matrice est donc de taille $(L_n \times n)$. Sur les colonnes de cette matrice est appliqué un pivot de Gauss afin de déterminer le rang de la matrice pour chaque valeur de n. Lorsque n est un multiple de la longueur du code n_c, la valeur du rang chute : ce phénomène est appelé *déficience de rang*. Pour les autres valeurs de n la matrice est théoriquement de rang plein n. Comme dit précédemment, les bits de chaque mots de code sont issus de la combinaison linéaire de bits d'information lors du codage. Ainsi, lorsque n est un multiple de n_c, il existe des colonnes de la matrice d'interception qui sont dépendantes d'autres colonnes, ce qui explique la chute de rang de la matrice d'interception. L'ampleur de cette chute de rang est directement liée au rendement du code. En effet, pour un code de rendement $\frac{k_c}{n_c}$, la déficience de rang observée est $l \cdot (n_c - k_c)$ quand la longueur testée est $n = l \cdot n_c$ pour $l \in \mathbb{N}^*$. Dans ces cas, le rang de la matrice d'interception est : $l \cdot k_c$.

En 2005, Sicot et Houcke généralisent la méthode au traitement d'une trame entachée d'erreurs dans [82]. Le principe de l'algorithme reste le même. Toutefois, la présence d'erreurs dans la trame reçue ne permet pas de détecter les déficiences de rang avec la même aisance. Pour palier à cela, les auteurs proposent de rechercher des *colonnes presque dépendantes*. En posant un seuil sur le nombre de 1 dans les colonnes de la partie inférieure de la matrice d'interception, il est alors possible de déterminer les colonnes qui sont presque dépendantes et de détecter une déficience de rang malgré la présence d'erreurs. La valeur de ce seuil dépend de la probabilité d'erreur du canal. Les erreurs de transmission n'ont pas la même influence sur l'algorithme en fonction de leur position dans la matrice d'interception. Elles peuvent empêcher l'identification si elles sont présentes dans la partie échelonnée. En effet, la présence d'erreurs dans la partie

supérieure de la matrice d'interception occasionne la propagation de celles-ci lors de l'application du pivot de Gauss. Ce problème est géré par une procédure itérative.

Par la suite, la méthode a été généralisée à d'autres types de codes. Barbier *et al.* proposent une analyse théorique de la méthode [3] et étendent son application aux codes convolutifs [4]. De même, dans [40, 123], Marazin *et al.* apportent de nouvelles analyses théoriques pour l'identification de codes convolutifs. En 2014, Liu et al. [52] généralisent la méthode du critère du rang pour qu'il puisse fonctionner sur le corps de Galois non binaire. Cela rejoint les travaux de Zrelli et al. [124] de 2015. De même, d'autres méthodes s'adaptent à la reconnaissance de code Reed-Solomon [54, 71, 91, 118] ou au code BCH [113, 119].

1.2.2.2 Améliorations utilisant les informations souples

L'algorithme basé sur le critère du rang repose sur l'algèbre binaire. Cependant il est possible d'améliorer la procédure en utilisant les fiabilités sur les bits lorsqu'elles sont disponibles. Une première implémention consiste à ordonner les lignes de la matrice d'interception selon leur fiabilité comme proposé dans [83]. Cette manipulation permet de faire remonter les bits les plus fiables dans la partie supérieure de la matrice d'interception. Ainsi, l'identification du rang de la matrice est améliorée. Cette approche a aussi été utilisée dans [87] pour les codes convolutifs.

1.3 Reconstruction de codes correcteurs d'erreurs

L'identification des paramètres d'un code ne suffit pas à retrouver le message initialement envoyé. En effet, pour cela il faut décoder le message reçu. Il est donc nécessaire d'identifier des relations de parité appartenant au code dual. A partir de ses relations de parité identifiées il sera possible de mettre en oeuvre un décodeur pour retrouver le message d'origine. Cependant, il est fréquent qu'en plus d'un code correcteur d'erreurs, le bloc de codage canal soit composé d'un entrelaceur, comme c'est le cas pour les Turbo-codes. Dans cette configuration, il est également nécessaire d'identifier l'ensemble des permutations effectuées par cet entrelaceur [2, 14, 27, 37, 38, 45–47, 56, 57, 75, 79, 82, 90, 95, 96]. Certaines approches utilisent la structure du codeur [26, 55], comme cela peut être le cas lors de l'identification des polynômes générateurs d'un code convolutif [12, 13, 31, 41, 41, 61, 101, 105, 112] ou pour les codes BCH [48, 49, 103, 119] ou les codes cycliques de manière générale [102, 114, 115, 120]. D'autres méthodes traitent des codes convolutifs et de leur code dual [40, 60, 66, 112, 121]. Des solutions ont aussi été trouvées pour reconstruire les codes convolutifs poinçonnés [18, 21, 58, 59, 65, 88] et les codes convolutifs non binaires [33, 78, 125]. Ces différents travaux ont naturellement ouvert la voie à l'étude de l'identification des Turbo-codes convolutifs [116, 117]. Dans [39, 40], les auteurs utilisent le critère du rang pour estimer les paramètres du premier codeur. Ensuite, en utilisant les résultats de cette première identification, il peuvent identifier le deuxième codeur à une matrice de permutation près. Ils obtiennent ainsi une version entrelacée de la matrice génératrice du second codeur codeur. Plusieurs articles

traitent de la reconstruction de ce second codeur [63, 116, 117]. La reconstruction d'un Turbo-code est traité dans [23, 28–30, 53, 70, 94] en adoptant parfois d'autres approches.

Pour les codes linéaires, de nombreuses méthodes consistent à reconstruire le code dual. La reconstruction du code dual consiste à identifier quelques relations de parité afin de construire une matrice de parité du code. Cette matrice est nécessaire pour réaliser le décodage de la trame reçue. Dans la suite de cette section, nous nous attardons sur deux approches. Une première approche consiste à identifier conjointement les paramètres et des relations de parité du code à partir de la méthode basée sur le critère du rang. Une autre approche consiste à la recherche de relations de poids faible en adaptant des algorithmes de décodage existant. Enfin, quelques travaux ont aussi proposé d'effectuer une correction partielle d'erreurs à partir des relations identifiées pour identifier de nouveaux éléments du code dual. Nous présenterons brièvement les avantages et les inconvénients de telles méthodes

1.3.1 Méthodes basées sur le critère du rang

La méthode du critère du rang peut être étendue à la recherche de relations de parité. Replaçons-nous un bref instant dans le contexte d'une trame reçue sans erreurs. A la création d'une matrice d'interception à n_c colonnes, la déficience de rang est maximale. Lors de l'application du pivot de Gauss, $(n_c - k_c)$ colonnes nulles apparaissent. Ces colonnes nulles sont le résultat de la combinaison linéaire de colonnes de la matrice. Ces combinaisons linéaires sont des relations de parité du code par définition. Notons $\mathbf{M}_{tri}^{(n)}$ la matrice d'interception triangularisée à n colonnes. L'opération de triangularisation de la matrice d'interception $\mathbf{M}^{(n)}$ peut se résumer ainsi :

$$\mathbf{M}_{tri}^{(n)} = \mathbf{M}^{(n)} \cdot \mathbf{P}^{(n)}$$

où $\mathbf{P}^{(n)}$ est une matrice carrée de taille $(n \times n)$. Lorsque $n = n_c$, des colonnes de $\mathbf{M}_{tri}^{(n_c)}$ sont nulles. Les indices des colonnes nulles dans $\mathbf{M}_{tri}^{(n_c)}$ correspondent aux indices des colonnes de $\mathbf{P}^{(n_c)}$ étant des relations de parité.

En présence d'erreurs dans la trame reçue, cette opération ne fonctionne plus aussi bien. Dans [83], les auteurs proposent une solution pour tirer parti de l'information disponible dans la partie non réduite de la matrice d'interception. Les erreurs de transmission n'ont pas la même influence sur l'algorithme en fonction de leurs positions dans la matrice $\mathbf{M}^{(n)}$. Elles ont peu d'importance dans la partie inférieure de la matrice. Par contre dans la partie triangulaire supérieure, elles peuvent empêcher la détection de certaines relations de parité. Néanmoins, avec les mêmes données, il est possible de construire différentes matrices $\mathbf{M}^{(n)}$ en changeant uniquement l'ordre des lignes. Les erreurs ne seront plus aux mêmes positions et on pourra alors éventuellement détecter de nouvelles relations de parité. L'algorithme proposé peut donc être implémenté de manière itérative en opérant des permutations aléatoires sur les lignes de la matrice d'interception.

Ces travaux ont naturellement donné lieu à leur adaptation aux codes convolutifs binaires [64, 87] et aux codes à symboles non-binaires [122, 125].

1.3.2 Méthodes basées sur la recherche de relations de parité de poids faible

Une recherche exhaustive des relations de parité peut être très coûteuse en calculs. Sans hypothèse supplémentaire, la recherche est un problème NP-complet d'après Valembois [99]. Cependant, si le poids des relations est constant alors le coût de recherche est polynomial. Cette hypothèse s'applique notamment aux codes convolutifs ainsi qu'aux codes LDPC réguliers. Elle peut aussi être appliquée aux codes convolutifs entrelacés et aux Turbo-codes. Dans [17], [22] et [97], les auteurs recherchent et testent des relations de petit poids. L'un des avantages de ces approches est qu'elles nécessitent peu de mots de code reçus pour fonctionner (*cf.* [24]). La probabilité qu'une équation de parité de poids w soit vérifiée après le canal binaire symétrique de probabilité d'erreur p_e est égale à $\frac{1+(1-2p_e)^w}{2}$. Donc plus w est petit plus il sera aisé de détecter si la relation est vérifiée.

Dans son manuscrit de thèse [97] Audrey Tixier traite, entre autres, de la reconnaissance des codes LDPC par recherche d'équations de parité de poids faible. Le chapitre 6 de ce manuscrit présente une méthode basée sur l'algorithme de Dumer [32]. L'approche permet de généraliser et d'unifier les méthodes existantes de Valembois [99] et de Cluzeau-Finiasz [22] mais aussi celle de Sicot et Houcke [82]. Elle permet aussi de retrouver plus efficacement un ensemble d'équations de parité de petit poids. Initialement, l'algorithme de Dumer est utilisé en décodage et fait parti des méthodes de décodage appelées Information Set Decoding (ISD)(comme les méthodes de Prange [74], de Léon [51] et de Stern [86] par exemple). Cela consiste à retrouver les motifs d'erreurs de poids faible dans un mot de code reçu connaissant la matrice de parité et le syndrome de l'erreur. Si on note $\mathbf{H}$ la matrice de parité d'un $\mathcal{C}(n_c, k_c)$ code et $\tilde{\mathbf{c}}$ un mot entaché d'erreurs. Le vecteur d'erreur $\mathbf{e}$ est tel que $\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e}$. Par définition, le syndrome de $\tilde{\mathbf{c}}$ est $\mathbf{s} = \tilde{\mathbf{c}} \cdot \mathbf{H}^T = (\mathbf{c} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T$. Les méthodes ISD permettent de déterminer un vecteur d'erreur $\mathbf{e}$ de poids au plus w pour le syndrome $\mathbf{s}$.

L'avantage de l'algorithme de Dumer par rapport à d'autres approches du même type est de rendre la recherche efficace en imposant des contraintes moins grande sur la répartition des poids dans les motifs d'erreur recherchés. Ceci augmente la probabilité de retrouver ces motifs lors d'une itération. Cette méthode peut être adaptée à la recherche de mots de poids faible. En effet, si on impose que $\mathbf{s} = \mathbf{0}$ alors par définition $\mathbf{e}$ est un mot de code de $\mathcal{C}$ car $\mathbf{e} \cdot \mathbf{H}^T = \mathbf{0}$. Pour adapter la méthode de Dumer à la recherche de relations de parité, on construit une matrice $\mathbf{M}$, de taille $(L \times (n_c + L))$, à partir de L mots reçues du code $\mathcal{C}$, noté $\tilde{\mathbf{c}}_i$ ($i \in [\![1, L]\!]$) et on lui concatène une matrice identité de taille L comme indiqué par la Figure 1.10. A partir de là, le but est de trouver les mots $\mathbf{h}_M$ tels que :

$$\mathbf{h}_M \cdot \mathbf{M}^T = \mathbf{0} \tag{1.15}$$

D'après l'équation 1.15, $\mathbf{h}_M$ correspond à une relation de parité du code ayant $\mathbf{M}$ pour matrice génératrice. Comme indiqué par la Figure 1.10, $\mathbf{h}_M = [\mathbf{h}\ \mathbf{s}_h]$ où $\mathbf{s}_h$ est de longueur L et de poids w_s. Si $\mathbf{h}_M.\mathbf{M}^T = \mathbf{0}$, alors $\mathbf{s}_h$ est égal au produit de la partie gauche de la matrice $\mathbf{M}$ (qui

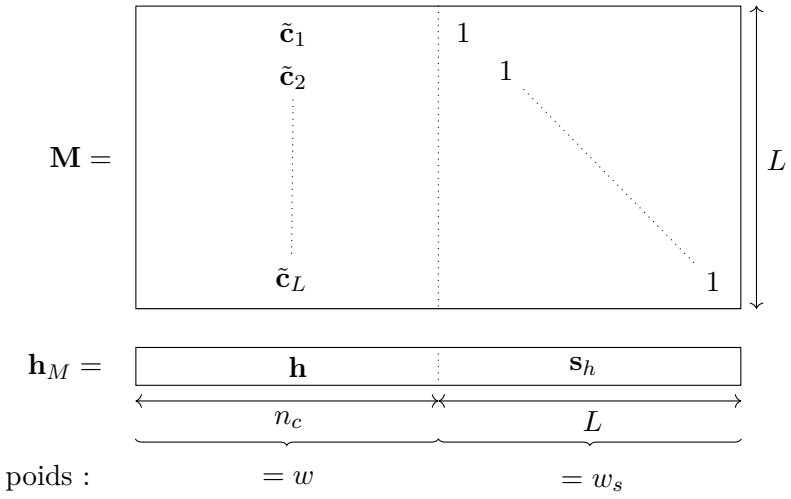


FIGURE 1.10 – Matrice des mots de code reçu concaténés à une matrice identité pour l'identification de relations de parité $\mathbf{h}$ de poids t

correspond à une matrice d'interception) et de $\mathbf{h}$. Le raisonnement pour retrouver des relations de parité à partir de $\mathbf{h}_M$ utilise la Proposition 2.

Proposition 2. *Soit* $\mathbf{h} = (h(0), ..., h(n_c - 1))$ *un vecteur de poids* w.

- *Si* $\mathbf{h}$ *est une relation de parité du code* $\mathcal{C}$*, alors pour tout* $i \in [\![1, L]\!]$ *:*

$$\mathbb{P}(\mathbf{h} \cdot \tilde{\mathbf{c}}_i^T = 1) = \frac{1 - (1 - 2p_e)^w}{2} \tag{1.16}$$

- *Sinon, pour tout* $i \in [\![1, L]\!]$ *:*

$$\mathbb{P}(\mathbf{h} \cdot \tilde{\mathbf{c}}_i^T = 1) = \frac{1}{2} \tag{1.17}$$

D'après la proposition 2 on déduit que $\mathbf{h}$ est une relation du code si $w_s \approx \frac{1-(1-2p_e)^w}{2} \cdot L$ et que $\mathbf{h}$ n'est pas une relation du code si $w_s \approx \frac{1}{2} \cdot L$. Ainsi, avec un algorithme comme celui de Dumer, il est possible de résoudre l'équation 1.15 et parmi les solutions, on sélectionne celles contenant une relation de parité en observant la Proposition 2. Lors de l'exécution de l'algorithme, des permutations aléatoires sur les colonnes de la matrice $\mathbf{M}$ sont réalisées avant de résoudre l'équation 1.15. Cela permet de faire varier la distribution des poids dans $\mathbf{h}_M$ et de trouver de nouvelles relations à chaque permutation. Dans son manuscrit de thèse Audrey Tixier [97] propose, une méthode pour trouver des relations plus rapidement à partir de l'algorithme de Dumer. L'utilisation d'une permutation aléatoire à chaque itération de l'algorithme fait perdre la maîtrise de la position des poids dans $\mathbf{h}_M$ qui est a priori connue. Elle propose donc de contraindre le choix des permutations pour accélérer la recherche. Elles sont choisies aléatoirement en vérifiant une répartition pré-établie des colonnes provenant de la matrice identité et des colonnes provenant des mots de code.

De même, Carrier *et al.* [16] proposent de directement appliquer la réduction de Gauss de manière partielle sur les colonnes de la matrice d'interception $\mathbf{M}$ formée des L mots de code de

taille n_c reçus afin de réduire la complexité. L'application du pivot est de profondeur ℓ de sorte que la partie supérieure gauche de la matrice réduite soit la matrice identité $(\ell \times \ell)$, notée $\mathbf{I}_\ell$. Une fois la réduction opérée sur quelques colonnes de $\mathbf{M}$, la recherche de relation $\mathbf{h}$ se réduit à celle de relations $\mathbf{h}_Q$ pour la sous-matrice $\mathbf{Q}$, comme présentée dans la Figure 1.11. Ensuite les relations de parité du code sont de la forme $\mathbf{h} = [\mathbf{M}'.\mathbf{h}_Q^T \ \mathbf{h}_Q]$.

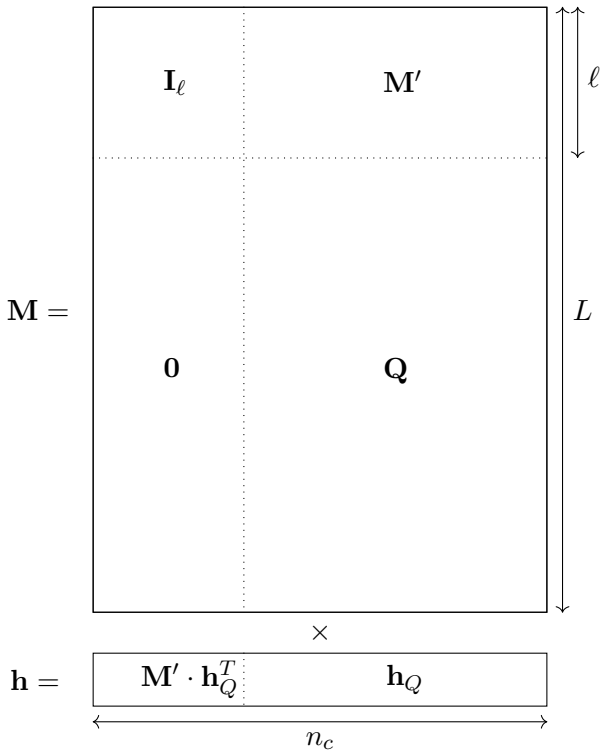


FIGURE 1.11 – Forme de la matrice après le pivot partiel

1.3.3 Méthodes avec décodage partiel

Dans l'ensemble des méthodes brieffement présentées, les données souples sont peu exploitées. Toutefois, une façon d'utiliser les données souples consiste à tirer parti des relations de parité déjà identifiées pour essayer de corriger les erreurs de la séquence reçue et d'itérer le processus pour identifier de nouvelles relations jusqu'à identification complète du code. Ce principe a été proposé par Cluzeau [19] et repris dans la thèse de Zrelli [122]. L'idée est séduisante mais possède plusieurs problèmes qui ne sont pas totalement résolus à ce jour :

- Une séquence codée très bruitée tend à être distribuée comme une séquence aléatoire. L'application d'un algorithme de reconstruction sur une telle séquence peut engendrer l'identification de mauvaise relations de parité. Dans ce cas, l'utilisation de ces mauvaises relations de parité dans le processus itératif pour corriger la séquence va engendrer des erreurs supplémentaires.

- Il est a priori nécessaire de prendre une fiabilité en compte pour chaque relation de parité. La définition de cette fiabilité et le processus de décodage tenant compte de cette fiabilité sont à définir.
- Généralement, un code est utilisé pour fonctionner à un certain niveau de bruit. Malheureusement la séquence reçue peut être obtenue avec un niveau de bruit plus élevé. Ce qui signifie que les relations de parité et le décodeur associé risquent de ne pas être en mesure de corriger toutes les erreurs. Lors des premières itérations, la restriction à un nombre limité de relations identifiées rend la correction d'erreurs encore plus difficile.

L'avantage de ce principe est qu'il peut être couplé à n'importe qu'elle autre méthode d'identification et permettre un gain de performance lorsque le niveau de bruit n'est pas trop important.

1.4 Conclusion

La majorité des méthodes présentées dans ce chapitre utilisent les informations fermes pour l'identification et la reconstruction des codes correcteurs d'erreurs dans le contexte non-coopératif. Des améliorations tirant parti des fiabilités des bits reçus ont été proposées pour s'assurer que les bits les moins entachés d'erreurs soient utilisés pour déterminer les paramètres d'un code ou pour identifier des relations de parité. Dans le contexte semi-coopératif, les méthodes basées sur l'utilisation d'un catalogue de codes correcteurs s'inspirent des algorithmes de décodage MAP (Maximum A Posteriori). En effet, ces méthodes estiment la vraisemblance de la valeur d'un syndrome à partir de la vraisemblance sur la valeur des bits. A travers les travaux présentés dans les chapitres qui vont suivre, nous proposons une alternative aux méthodes basés sur l'algèbre binaire pour l'identification et la reconstruction de codes en bloc binaires. Dans la suite, nous présentons donc des méthodes d'identification et de reconstruction à partir de symboles bruités.

Chapitre 2

Identification des paramètres par Déficience du Nombre de Classes

La plupart des méthodes d'identification de code correcteur d'erreurs utilisent des approches algébriques basées sur les données décidées. Il existe plusieurs façons de visualiser les transformations réalisées par un code correcteur d'erreurs. Le codage peut, par exemple, être caractérisé par la disposition qu'il impose aux mots de code dans l'espace vectoriel créé. Une mesure de ce phénomène peut être faite par l'observation des distances de Hamming entre les mots deux à deux et/ou par la distance minimale. La Figure 2.1 permet d'observer la différence de distribution des distances de Hamming pour des mots de code de Hamming $\mathcal{C}(7,4)$ (*cf.* Exemple 1.1.3.) et des mots i.i.d. (indépendants et identiquement distribués) de longueur 7. L'absence de bruit permet de distinguer sans ambiguïté que ce code de Hamming est de distance minimale 3. En effet, les mots distincts les plus proches sont à une distance de Hamming de 3. La probabilité que deux mots soient distants de 3 ou 4 bits est la plus élevée, alors que cette probabilité est nulle pour des distances de 1, 2, 5 ou 6 bits. Dans le cas des mots i.i.d., les distances sont distribuées selon une loi binomiale de paramètres $n = 7$ et $p = \frac{1}{2}$. La différence majeure est que dans le cas du code de Hamming certaines zones de l'espace codé sont vides car certains mots de $\mathrm{GF}(2)^7$ ne sont pas des mots du code, en effet il n'y a que 2^4 mots de code (contre 2^7 dans le cas i.i.d.). En présence de bruit, les distributions des distances peuvent être discriminées. La Figure 2.2 présente la distribution des distances de Hamming après un passage à travers un canal binaire symétrique de paramètre $p_e = 3 \cdot 10^{-3}$. Les distributions sont sensiblement les mêmes que pour la Figure 2.1. Toutefois, les probabilités que deux mots soient distants de 1, 2, 5 ou 6 bits ne sont plus nulles.

Ces distributions des distances présentent donc un caractère discriminant de la présence ou non d'un code. L'inconvénient d'utiliser la distance de Hamming est la perte d'information liée à la prise de décision sur les mots reçus. Dans ce chapitre, nous présentons une méthode d'identification des paramètres d'un code linéaire en utilisant une information souple sur les données reçues. Les mesures seront effectuées dans l'espace des réels, et nous utiliseront donc la distance euclidienne à la place de la distance de Hamming. Nous notons $d_E(\mathbf{a}, \mathbf{b})$ la distance

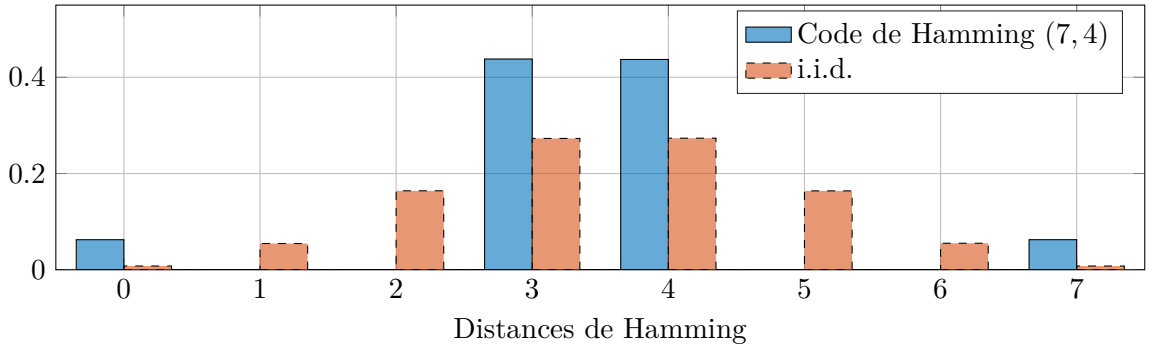


FIGURE 2.1 – Distribution des distances en l'absence de bruit pour un code de Hamming $\mathcal{C}(7,4)$ et pour des mots i.i.d. de longueur 7

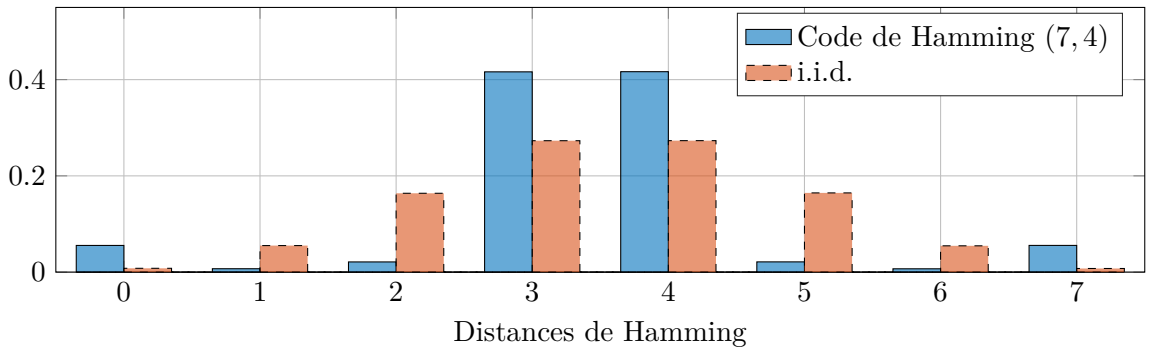


FIGURE 2.2 – Distribution des distances pour $p_e = 3 \cdot 10^{-3}$ pour un code de Hamming $\mathcal{C}(7,4)$ et pour des mots i.i.d. de longueur 7

euclidienne entre les deux vecteurs à composantes réelles **a** et **b**. Si **a** et **b** sont de taille n :

$$d_E(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=0}^{n-1} (a(i) - b(i))^2}$$

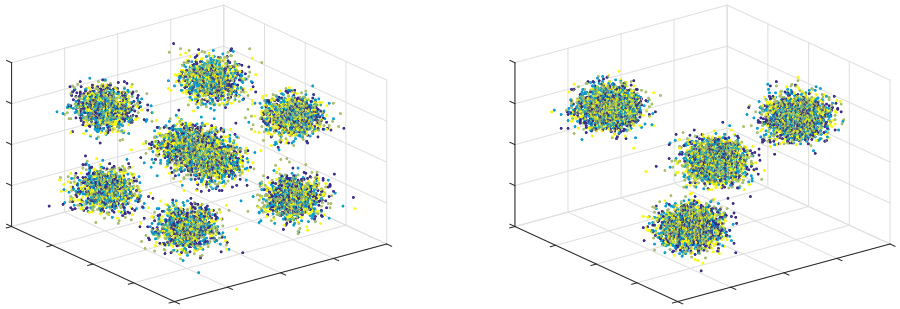
L'identification reposera sur une nouvelle notion : la *Déficience du Nombre de Classes*. Cette méthode a été publiée dans [8].

Dans la suite, nous présentons donc un moyen d'identifier la longueur d'un code et l'algorithme associé. Ensuite, nous proposons une méthode pour déterminer la dimension du code. Enfin, nous analysons les performances de ces techniques d'identification à travers des résultats obtenus par simulation.

2.1 Déficience du Nombre de Classes

La méthode présentée dans ce chapitre est fondée sur des considérations géométriques. Intuitivement, s'il était possible de représenter un nombre infini de mots de code reçus bruités dans un espace euclidien de dimension n_c, leurs positions seraient modifiées par les effets du canal à BBAG. En fait, chaque mot de code «se déplacerait vers un point proche selon une distribution gaussienne sphérique» [81]. Par conséquent, il se formerait des «agglomérats» (si l'énergie du bruit n'était pas trop élevée). Ces agglomérats se concentreraient autour des positions qu'avaient les mots de code avant l'adjonction des altérations causées par le bruit. La répartition des mots de code bruités dans cet espace suit donc une loi de mélange gaussien multivarié. De plus, comme évoqué précédemment, on peut prédire que cet espace est lacunaire par rapport à un espace rempli de mots i.i.d. bruités. En effet, en raison de la redondance introduite par le codeur, seuls 2^{k_c} mots de code sont placés dans un espace pouvant en contenir 2^{n_c}. Autour des 2^{k_c} positions de mots de code non bruités, les mots de code bruités se répartissent selon une distribution gaussienne. Pour illustrer ce phénomène, la Figure 2.3 permet de comparer deux cas de distribution dans un espace euclidien à trois dimensions pour un BBAG de variance $\sigma_b^2 = 0.1$: avec des mots de longueur 3 en l'absence d'un code correcteur d'erreurs (Figure 2.3(a)) et en présence du code $\mathcal{C}(3, 2)$ de l'Exemple 1.1.2. (Figure 2.3(b)). En présence du code (Figure 2.3(b)), et malgré les effets du bruit, il se forme des amas très denses autour des $2^{k_c} = 4$ positions initiales des mots de code. Dans le cas i.i.d., nous distinguons bien 8 agglomérats qui correspondent aux 8 mots possibles (Figure 2.3(a)). Il apparaît visuellement facile de discriminer le cas codé du cas non codé et la position initiale des mots de code grâce à leur nombre limité.

Dans ce chapitre, les algorithmes décrits visent à déterminer la longueur du code n_c et sa dimension k_c à travers un processus de classification. Chaque agglomérat est une *classe*. Chaque classe correspond à l'ensemble des mots bruités issus du même mot non bruité. Pour un espace euclidien de dimension donnée, le nombre de classes sera donc plus faible en présence d'un code correcteur d'erreurs que lorsque les mots sont i.i.d.. Par conséquent, nous souhaitons déterminer quand des classes sont manquantes pour identifier les paramètres du code. Une



(a) (b)

FIGURE 2.3 – Observation de mots bruités dans un espace euclidien de dimension 3 pour un canal BBAG de variance $\sigma_b^2 = 0.1$: (a) en l'absence de codage, (b) en présence de codage avec $n_c = 3$ et $k_c = 2$.

première définition de la *Déficience du Nombre de Classes* pourrait être : c'est un phénomène dû à la redondance induite par un code de longueur n_c où la quantité de classes est inférieure à celle attendue dans un espace euclidien donné de dimension n_c.

Définition 7. *Déficience du nombre de classes (DNC)*

Soit :

- $Z^{(n)}$ *: la variable aléatoire comptant le nombre de classes dans un espace euclidien de dimension n donné pour une classification de n-uplets i.i.d.*
- $z_c(n)$ *: le nombre de classes observées à partir du flux codé intercepté divisé en n-uplets.*

La déficience $\Delta_{DNC}(n)$ *est définie comme suit :*

$$\Delta_{DNC}(n) \triangleq \mathbb{E}(Z^{(n)}) - z_c(n) \tag{2.1}$$

Cette définition de la DNC est telle qu'elle permet de comparer le nombre de classes *observé* en présence d'un code correcteur d'erreurs ($z_c(n)$) à celui qui est *attendu* avec une trame i.i.d. ($\mathbb{E}(Z^{(n)})$).

La section suivante développe l'établissement d'un critère à partir de la Définition 7 pour identifier la longueur du code n_c.

2.2 Identification de la longueur du code

Nous considérons le modèle de transmission simplifié décrit par la Figure 1.2. Côté émetteur, le message $\mathbf{m}$ est codé par le code $\mathcal{C}(n_c, k_c)$ et transmis à travers un canal donné. Du côté du récepteur, il est supposé que l'utilisateur a accès aux bits reçus et à une mesure de fiabilité de chacun de ces bits. Nous ferons l'hypothèse que les données reçues résultent d'une modulation

MP-2 (Modulation de Phase à deux états : au bit 0 est associé le symbole -1 et au bit 1 est associé le symbole $+1$) et du passage à travers un canal à BBAG. Ce canal est caractérisé par la variance du bruit σ_b^2. De plus, la synchronisation est supposée parfaite (le flux de données reçues commence par un mot de code complet). Le signal reçu $\mathbf{y}$ est composé de N échantillons :

$$y(k) = s(k) + b(k), \forall k \in [\![0, N-1]\!] \tag{2.2}$$

où $s(k)$ (respectivement $b(k)$) est le k-ième élément du signal envoyé (respectivement du bruit). La séquence $\mathbf{s}$ résulte d'une concaténation de mots de code modulés. La modulation étant une modulation de phase à deux états, pour tout k, le symbole $s(k)$ appartient à $\{\pm 1\}$. Modélisé ainsi, les symboles reçus $y(k)$ sont proportionnels à des rapports de vraisemblance sur les bits correspondants. C'est une manière simple de générer des mesures de fiabilité sur les bits reçus.

Dans ce qui suit, la probabilité d'un événement E et l'espérance d'une variable aléatoire X seront notées $\mathbb{P}(E)$ et $\mathbb{E}(X)$, respectivement. Une variable aléatoire sera représentée par une majuscule tandis que les lettres minuscules représentent la valeur déterministe correspondante. La distribution uniforme sur un ensemble I est symbolisée par $\mathcal{U}(I)$. De même, $\mathcal{N}(\mu, \sigma^2)$ représente une distribution normale de moyenne de μ et de variance σ^2 alors que $\chi^2(n)$ décrit une distribution du khi-2 à n degrés de liberté.

2.2.1 Processus de classification

Pour estimer la longueur du code, l'objectif est de trouver la dimension n pour laquelle la déficience Δ_{DNC} est la plus grande. À cette fin, la séquence bruitée reçue de N échantillons est divisée en L_n blocs contigus, notés $\mathfrak{B}_i^{(n)}$, de longueurs n comme décrit ci-dessous :

$$\mathfrak{B}_i^{(n)} = [y(i \cdot n), \cdots, y(i \cdot n + n - 1)] \tag{2.3}$$

avec $L_n = \left\lfloor \frac{N}{n} \right\rfloor$ et $i \in [\![0, L_n - 1]\!]$.

Ensuite, pour chaque valeur de n, un processus de classification basé sur un critère de distance euclidienne est opéré sur ces blocs à travers les deux étapes suivantes :

1. Le premier bloc $\mathfrak{B}_0^{(n)}$ définit le premier élément de la première classe : on le nomme *mot de référence* R_0. Pour chaque valeur $i \neq 0$, les blocs $\mathfrak{B}_i^{(n)}$ sont comparés à R_0 : si la distance euclidienne entre R_0 et $\mathfrak{B}_i^{(n)}$ est inférieure à un seuil choisi β, alors ils appartiennent à la même classe. Sinon, si cette distance est supérieure à 2β, alors $\mathfrak{B}_i^{(n)}$ est un candidat pour être le mot de référence d'une autre classe. Autrement, lorsqu'une distance se situe entre β et 2β, le bloc concerné est mis de côté jusqu'à ce que tous les autres aient été comparés à R_0. Ensuite, R_1 est choisi parmi les candidats (qui était à une distance d'au moins 2β de R_0). Le mots R_1 agglomère tout bloc non classé étant à une distance inférieure à β (même s'il avait été mis de côté parce qu'il était trop près de R_0). Ceux à au moins 2β de R_1 sont candidats pour devenir R_2 et les autres sont mis de côté. Le processus est répété jusqu'à

ce qu'il n'y ait plus de candidats pour le prochain mot de référence. Cette façon de choisir les mots de référence permet de minimiser le phénomène de *superposition de classes*.

2. Les blocs mis de côté et non classés lors de l'étape 1 sont classés selon le même processus. Néanmoins, le choix d'un nouveau mot de référence est moins contraint. Sa distance à un mot de référence existant déjà peut prendre une valeur entre β et 2β.

Une fois qu'il entre dans une classe, un bloc est retiré de l'ensemble des blocs, i.e. chaque bloc n'appartient qu'à une seule et unique classe. Finalement, ces deux étapes nous donnent accès au nombre de classes $z_c(n)$ obtenu à partir de la trame codée :

$$z_c(n) = \text{Card}(\mathcal{R}^{(n)}) \tag{2.4}$$

où $\mathcal{R}^{(n)}$ est l'ensemble des mots de référence pour un n donné, et $\text{Card}(\mathcal{R}^{(n)})$ son cardinal.

Une première appréciation de résultats obtenus par ce processus de classification est représentée par la Figure 2.4(a). Ici, $\mathcal{C}(n_c, k_c)$ est un code LDPC dont la longueur et le rendement sont $n_c = 25$ et $\rho = \frac{2}{5}$ respectivement. Pour cet exemple, $N = 25000$ bits ont été envoyés. Pour chaque taille de bloc n, l'espérance du nombre de classes en l'absence de codage (cercles rouges) et le nombre de classes obtenues dans le cas codé (carrés noirs) sont comparés. Les quantités respectives de classes dans le cas codé et dans le cas non codé coïncident approximativement pour chaque taille de bloc n, à l'exception d'une seule. À $n = n_c = 25$, le code génère un phénomène de déficience (DNC). De même, nous remarquons quelques déficiences subsidiaires apparaissant pour différentes valeurs de n comprises entre 10 et 15. Celles-ci sont dues à la structure particulière du code LDPC considéré. De plus, en raison de la limitation du nombre de bits reçus N, pour n se situant entre 13 et 30, la valeur de $\mathbb{E}(Z^{(n)})$ diminue. Plus précisément, dans ce cas $\lfloor \frac{N}{n} \rfloor$ est une borne supérieure de $\mathbb{E}(Z^{(n)})$. Ces déficiences subsidiaires et la forme générale de ces courbes empêchent de directement exploiter ces résultats pour estimer la longueur du code. Pour détecter automatiquement la déficience prépondérante, le critère est adapté : la DNC normalisée (Définition 8). Les effets de cette normalisation sont illustrés par la Figure 2.4(b) qui révèle la déficience la plus significative et libère de la limitation en données reçues.

Définition 8. *La déficience du nombre de classes normalisée φ :*

$$\begin{aligned} \varphi(n) &= \frac{\mathbb{E}(Z^{(n)}) - z_c(n)}{\mathbb{E}(Z^{(n)})} \\ &= \frac{\Delta_{DNC}(n)}{\mathbb{E}(Z^{(n)})} \end{aligned} \tag{2.5}$$

Cette définition conduit au critère de détection suivant pour l'identification de la longueur du code :

$$\hat{n}_c = \underset{n}{\text{argmax}}(\varphi(n)) \tag{2.6}$$

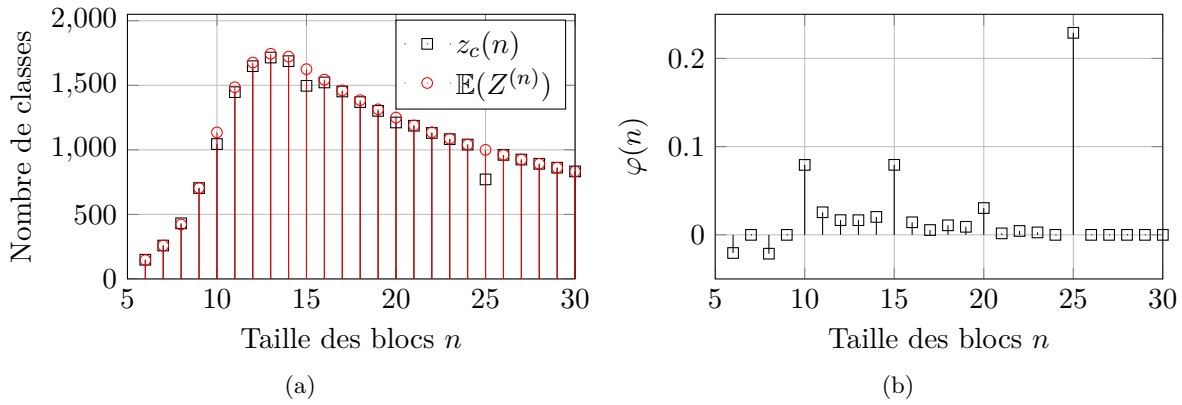


FIGURE 2.4 – Mise en évidence de la déficience : (a) comparaison de $\mathbb{E}(Z^{(n)})$ et $z_c(n)$, (b) déficiences normalisées

Dans cette présentation du principe de notre algorithme, la définition du seuil β a été mise de côté volontairement. Évidemment la valeur de ce paramètre est centrale dans cette méthode d'identification. Par conséquent, dans ce qui suit, les dépendances au seuil β sont mises en évidence grâce aux notations suivantes : le nombre de classes observées dans la trame codée $z_c(n)$ et la variable aléatoire $Z^{(n)}$ sont dénotés $z_c(n, \beta)$ et $Z_\beta^{(n)}$ respectivement. Nous adaptons aussi la notation pour l'ensemble des mots de référence $\mathcal{R}^{(n)}$: $\mathcal{R}^{(n,\beta)}$. La DNC normalisée devient une fonction de β : $\varphi(n, \beta)$.

2.2.2 Choix du seuil pour la classification

Dans l'idéal, il serait préférable d'appliquer l'algorithme de classification pour un seuil optimal β_{opt}, c'est à dire celui qui maximise la déficience du nombre de classes. Par exemple, avec le même code LDPC que précédemment ($n_c = 25$, $k_c = 10$), il est possible de déterminer empiriquement le meilleur seuil comme le montre la Figure 2.5, pour $n = n_c$. Le seuil optimal est $\beta_{opt} \approx 3.7$. Un moyen d'obtenir le seuil optimal serait de résoudre le problème suivant :

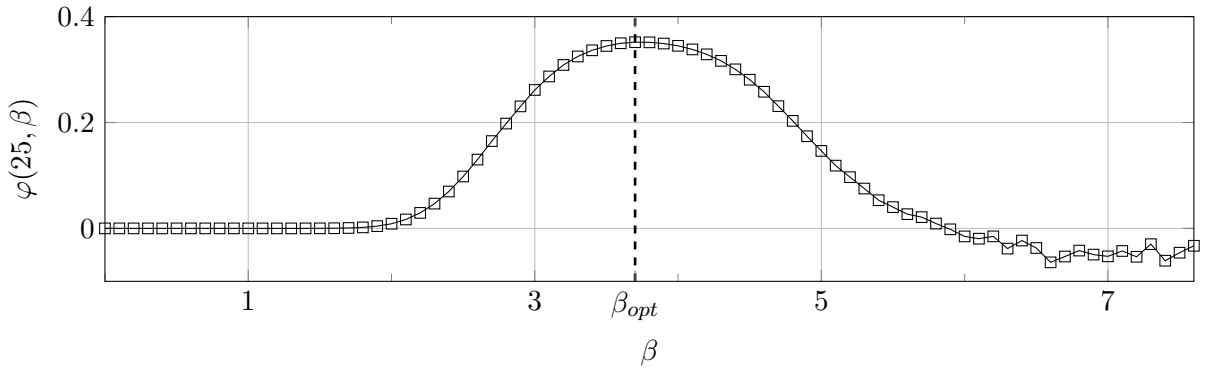


FIGURE 2.5 – Déficience normalisée lorsque $n = n_c = 25$ pour différentes valeurs de β

$$\beta_{opt} = \underset{\beta}{\operatorname{argmax}}\, \mathbb{E}\left(\varphi(n, \beta)\right) \tag{2.7}$$

Malheureusement, il est difficile d'obtenir une expression analytique pour β_{opt} puisqu'elle dépend du code lui-même. Ainsi, au lieu de choisir un seuil fixe, nous proposons de balayer une plage de seuils selon un pas choisi $\Delta\beta$. De plus, d'après la Figure 2.5, $\varphi(n = n_c, \beta)$ est faible pour des valeurs extrêmes de β. En effet, pour les seuils les plus élevés, tous les blocs sont agglomérés en une seule classe. Par contre, lorsque β tend vers 0, chaque bloc est considéré comme une classe à part entière. Par conséquent, il n'est pas nécessaire d'opérer la classification pour ces valeurs de β. Ainsi, les valeurs du seuil trop éloignées de β_{opt} ne sont pas explorées pour limiter la quantité de calculs en définissant une valeur minimale et maximale pour β.

Tout d'abord, il semble inutile de classer des blocs pour $\beta < \min_{i \neq j}(d_E(\mathfrak{B}_i^{(n)}, \mathfrak{B}_j^{(n)}))$: il y aurait autant de classes que de blocs créés. Dans ce contexte, il serait impossible d'observer une DNC. Cela nous fournit le seuil minimum :

$$\beta_{min}(n) = \min_{i \neq j}(d_E(\mathfrak{B}_i^{(n)}, \mathfrak{B}_j^{(n)})) \tag{2.8}$$

Ensuite, il est inutile de choisir un seuil β plus grand que la distance moyenne entre deux blocs. Définissons $Y(k) = S(k) + B(k)$ la variable aléatoire portant la mesure de fiabilité du k-ième bits de la séquence reçue $\mathbf{y}$ avec $S(k) \sim \mathcal{U}(\{-1, +1\})$ et $B(k) \sim \mathcal{N}(0, \sigma_b^2)$. Notons, X_{d^2} la variable aléatoire représentant la distance euclidienne au carré entre deux blocs i et j choisis au hasard :

$$X_{d^2} = \sum_{p=0}^{n-1} (Y(i \cdot n + p) - Y(j \cdot n + p))^2 \tag{2.9}$$

Si nous considérons $X(p) = (Y(i \cdot n + p) - Y(j \cdot n + p))^2$, alors $X(0), X(1), X(2), \cdots, X(n-1)$ est une séquence de variables aléatoires i.i.d.. Ces variables sont toutes indépendantes les unes des autres et chacune d'entre elle a la même distribution de probabilité que les autres. Notons respectivement μ_X et σ_X^2 la moyenne et la variance de cette distribution. A partir d'une approximation basée sur le théorème central limite [7], pour n grand, $X(0)+X(1)+\cdots+X(n-1)$ peut asymptotiquement être vu comme une variable aléatoire distribuée selon une loi normale de moyenne $n.\mu_X$ et de variance $n \cdot \sigma_X^2$. Par conséquent, si X_{d^2} tend à être normalement distribuée avec pour moyenne $\mu_{d^2} = n \cdot \mu_X$ et pour variance $\sigma_{d^2}^2 = n \cdot \sigma_X^2$, alors :

$$\mathbb{P}(d_E^2(\mathfrak{B}_i^{(n)}, \mathfrak{B}_j^{(n)}) \leqslant \mu_{d^2}) \approx \frac{1}{2} \tag{2.10}$$

Cela signifie qu'il y a une chance sur deux que deux blocs choisis au hasard appartiennent à la même classe lorsque β^2 atteint μ_{d^2}. Dans notre processus de classification, dépasser $\beta = \sqrt{\mu_{d^2}}$ donnerait trop de potentiel d'agglomération à chaque bloc. Ceci ferait tendre le nombre de classes vers 1. Cette constatation mène à une borne supérieure pour β :

$$\beta_{max}(n) = \sqrt{\mu_{d^2}} \tag{2.11}$$

Finalement, pour un n fixé :

$$\min_{i \neq j}(d_E(\mathfrak{B}_i^{(n)}, \mathfrak{B}_j^{(n)})) \leqslant \beta \leqslant \sqrt{\mu_{d^2}} \tag{2.12}$$

Malgré le fait que la valeur du seuil soit bornée, il est nécessaire de calculer plusieurs valeurs de $\varphi(n, \beta)$ pour β se situant entre β_{min} et β_{max}. Ainsi, pour chaque taille de bloc n, il faut déterminer une valeur représentative de la déficience à partir de celles obtenues pour chaque valeur de β entre $\beta_{min}(n)$ et $\beta_{max}(n)$. Pour prendre une décision sur la prépondérance d'une déficience, nous définissons une nouvelle DNC normalisée :

Définition 9. *La déficience du nombre de classes normalisée et fiabilisée φ_s :*

$$\varphi_s(n) \quad = \sum_{\beta \in [\beta_{min}(n),\, \beta_{max}(n)]} a_\beta(n) \cdot \varphi(n, \beta) \tag{2.13}$$

avec $a_\beta(n)$ le coefficient de normalisation dépendant de β et de n :

$$a_\beta(n) = \frac{Z_\beta^{(n)} - \min_\beta(Z_\beta^{(n)})}{\max_\beta(Z_\beta^{(n)}) - \min_\beta(Z_\beta^{(n)})} \tag{2.14}$$

Pour un n fixé, $\varphi_s(n)$ est la somme pondérée de plusieurs valeurs de $\varphi(n, \beta)$ (*cf.* Définition 9). Chaque coefficient de normalisation $a_\beta(n)$ permet de mettre à l'échelle chaque valeur de $\varphi(n, \beta)$ en fonction de sa fiabilité statistique. En effet, plus il y a de classes créées, plus la déficience mesurée est fiable. Cette normalisation permet de palier au fait que lorsque β tend vers β_{max} le nombre de classes créées est faible. Ceci génère des données aberrantes pour $\varphi(n, \beta)$ comme on peut en voir pour des seuils élevés sur la Figure 2.5. Ceci est d'autant plus vrai lorsque n est grand, car la limitation dans la quantité de données reçues impose de créer peu de classes. Finalement, une DNC doit être à la fois prépondérante et fiable pour être discriminante.

La Figure 2.6 représente la déficience du nombre de classes normalisée et fiabilisée ($a_\beta(n) \cdot \varphi(n, \beta)$) du code LDPC $\mathcal{C}(25, 10)$ pour $n \in [\![10, 30]\!]$ et pour $\beta \in [\beta_{min}(n), \beta_{max}(n)]$ avec un pas de $\Delta\beta = 0.5$. Les courbes blanches correspondent aux valeurs des seuils minimum (ronds) et maximum balayés (triangles). Nous pouvons remarquer que les valeurs de $a_\beta(n) \cdot \varphi(n, \beta)$ les plus grandes sont obtenues pour une taille de bloc $n = n_c = 25$ et pour un seuil β situé entre 2.5 et 5.5. Enfin, pour $\beta > \beta_{max}$ et $\beta < \beta_{min}$, $a_\beta(n) \cdot \varphi(n, \beta)$ n'est pas estimé et il est remplacé par un 0 dans cette figure. Dans cet exemple, le choix de β_{max} et β_{min} selon (2.12) réduit le temps de calcul en n'éliminant aucune valeur d'intérêt.

La Figure 2.7 représente φ_s en fonction de n. La plus grande déficience est obtenue pour $n = n_c$. Le critère de détection devient :

$$\hat{n}_c = \operatorname*{argmax}_n(\varphi_s(n)) \tag{2.15}$$

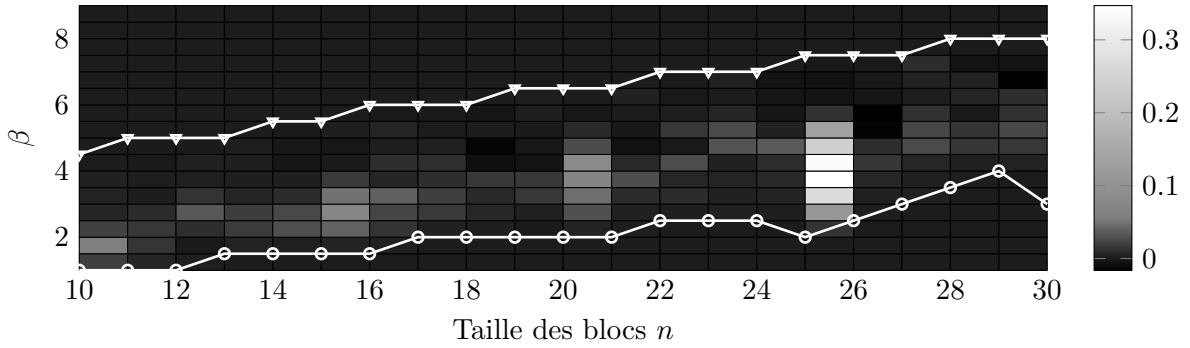


FIGURE 2.6 – Observation de $a_\beta(n) \cdot \varphi(n, \beta)$ pour une zone restreinte de valeurs de β

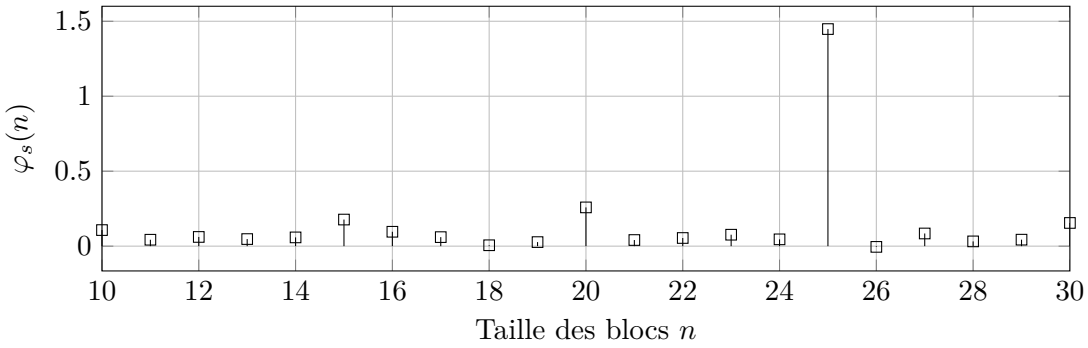


FIGURE 2.7 – Déficience normalisée et fiabilisée

2.2.3 Algorithme d'identification aveugle

Le processus d'identification aveugle est résumé par l'Algorithme 1. Cette section décrit le processus de classification de manière détaillée : spécifiquement les étapes 1 et 2 présentées précédemment. Elles correspondent aux deux étapes décrites dans la sous-section 2.2.1. Pour une taille de bloc n et un seuil sur la distance β, l'objectif est de calculer un nombre de classes $z_c(n, \beta) = \text{Card}(\mathcal{R}^{(n,\beta)})$. Comme défini précédemment, $\mathcal{R}^{(n,\beta)}$ est l'ensemble des mots de référence pour n et β donnés. Sachant que R_z est le mot de référence de la z-ième classe créée : $\mathcal{R}^{(n,\beta)} = \{R_z | z \in [\![0, z_c(n, \beta) - 1]\!]\}$. Ces mots de référence sont choisis à l'étape 1 et à l'étape 2 parmi les L_n blocs créés. De plus, ils agglomèrent d'autres blocs : cela forme des classes. Un bloc est considéré comme classé lorsqu'il est lui-même un mot de référence ou lorsqu'il est aggloméré par l'un d'entre eux. Pour un n donné, notons $\mathcal{I}_b^{(n)}(k)$ un ensemble d'indices de blocs défini comme suit :

$$\mathcal{I}_b^{(n)}(k) = \{i | \, d_E(\mathfrak{B}_k^{(n)}, \mathfrak{B}_i^{(n)}) \leqslant b\} \tag{2.16}$$

avec b qui correspond à un seuil valant β ou 2β.

A l'étape 1, le premier mot de référence est le bloc $R_0 = \mathfrak{B}_0^{(n)}$ (ligne 8 de l'Algorithme 1). $\mathcal{I}_\beta^{(n)}(0)$ contient donc les indices de tous les blocs agglomérés par $\mathfrak{B}_0^{(n)}$ pour un seuil β donné. Le

Algorithme 1 DNC : ALGORITHME D'IDENTIFICATION AVEUGLE

Entrées : $\mathbf{y}$, n_{min}, n_{max}

Pour $n = n_{min}$ à n_{max} **Faire**
  Créer L_n blocs $\mathfrak{B}_i^{(n)}$ de longueur n à partir de $\mathbf{y}$ comme défini par (2.3)
  Calculer $\beta_{min}(n)$ et $\beta_{max}(n)$
  Pour $\beta = \beta_{min}(n)$ à $\beta_{max}(n)$ **Faire**
    $\mathcal{F}_{2\beta}^{(n)} = \varnothing$
    $\mathcal{F}_{\beta}^{(n)} = \varnothing$
    $\mathcal{R}^{(n,\beta)} = \varnothing$
    $k = 0$
    Tant que $\mathrm{Card}(\mathcal{F}_{2\beta}^{(n)}) < L_n$ **Faire**
      $\mathcal{R}^{(n,\beta)} \leftarrow \mathcal{R}^{(n,\beta)} \cup \{\mathfrak{B}_k^{(n)}\}$
      $\mathcal{F}_{\beta}^{(n)} \leftarrow \mathcal{F}_{\beta}^{(n)} \cup \mathcal{I}_{\beta}^{(n)}(k)$
      $\mathcal{F}_{2\beta}^{(n)} \leftarrow \mathcal{F}_{2\beta}^{(n)} \cup \mathcal{I}_{2\beta}^{(n)}(k)$
      $k = \min(\{0, 1, \cdots, L_n - 1\} \setminus \mathcal{F}_{2\beta}^{(n)})$
    Fin Tant que
    $k = \min(\{1, \cdots, L_n\} \setminus \mathcal{F}_{\beta}^{(n)})$
    Tant que $\mathrm{Card}(\mathcal{F}_{\beta}^{(n)}) < L_n$ **Faire**
      $\mathcal{R}^{(n,\beta)} \leftarrow \mathcal{R}^{(n,\beta)} \cup \{\mathfrak{B}_k^{(n)}\}$
      $\mathcal{F}_{\beta}^{(n)} \leftarrow \mathcal{F}_{\beta}^{(n)} \cup \mathcal{I}_{\beta}^{(n)}(k)$
      $k = \min(\{0, 1, \cdots, L_n - 1\} \setminus \mathcal{F}_{\beta}^{(n)})$
    Fin Tant que
    $z_c(n, \beta) = \mathrm{Card}(\mathcal{R}^{(n,\beta)})$
    Calculer $\varphi(n, \beta)$
  Fin Pour
  Calculer $\varphi_s(n)$
Fin Pour
$\hat{n}_c = \mathrm{argmax}_n(\varphi_s(n))$

Etape 1. Etape 2.

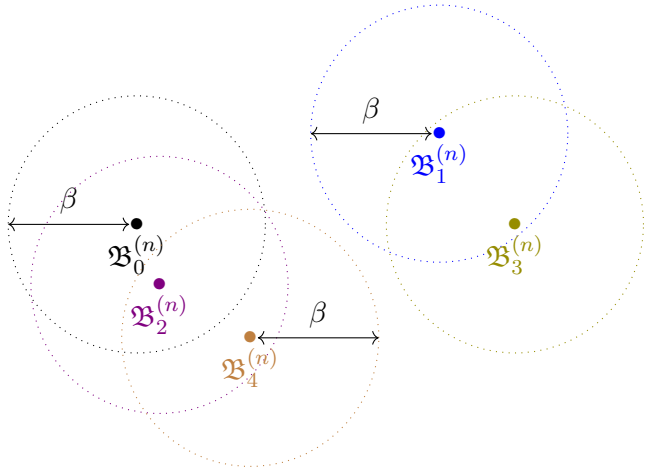
Sorties : $\hat{n}_c$

choix du nouveau mot de référence (R_1) correspond au premier bloc non classé distant d'au moins 2β de chaque mot de référence précédemment identifié (ici R_0) (ligne 13). Sous cette dernière contrainte, tous les blocs appartenant à $\mathcal{I}_{2\beta}^{(n)}(0)$ sont inéligibles pour devenir R_1 le prochain mot de référence. Pendant l'étape 1, pour chaque itération, les blocs classés ($\mathcal{I}_{\beta}^{(n)}(k)$) et les blocs compris dans $\mathcal{I}_{2\beta}^{(n)}(k)$ sont respectivement répertoriés dans $\mathcal{F}_{\beta}^{(n)}$ et dans $\mathcal{F}_{2\beta}(n)$ (ligne 11 et ligne 12 respectivement). À l'étape 2, la même classification est faite avec les blocs non classés restants après l'étape 1 (c'est-à-dire avec les blocs dans $\{0, 1, \cdots, L_n - 1\} \setminus \mathcal{F}_{\beta}^{(n)}$). Le premier mot de référence est choisi à la ligne 15 et les suivants sont sélectionnés à la ligne 19. A la fin de l'étape 2, chaque bloc a été classé dans une des $z_c(n, \beta)$ classes. Ce qui permet le calcul de $\varphi(n, \beta)$ (ligne 22). Pour clarifier ce qui se produit lors de ces deux étapes, l'Exemple 2.2.1. décrit le processus pour la classification de cinq blocs.

Ce processus de classification est répété pour chaque valeur de n et pour chaque valeur de β pour permettre le calcul de $\varphi_s(n)$ (ligne 24).

Exemple 2.2.1.

Considérons $L_n = 5$ blocs de taille n à classer selon un seuil β : $\mathfrak{B}_0^{(n)}$, $\mathfrak{B}_1^{(n)}$, $\mathfrak{B}_2^{(n)}$, $\mathfrak{B}_3^{(n)}$ et $\mathfrak{B}_4^{(n)}$. Afin de simplifier cet exemple, supposons que ces blocs ont les positions relatives suivantes dans le plan à 2 dimensions :



Chaque bloc a un rayon d'influence de β qui est représenté par un cercle en pointillés. Le tableau suivant donne un résumé des contraintes qui existent sur les positions relatives de ces blocs. Il nous permettra de suivre les étapes de la classification :

$d_E(\mathfrak{B}_i^{(n)}, \mathfrak{B}_j^{(n)})$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
$i=0$	$\leqslant \beta$	$\geqslant 2\beta$	$\leqslant \beta$	$\geqslant 2\beta$	$\geqslant \beta$ et $\leqslant 2\beta$
$i=1$	$\geqslant 2\beta$	$\leqslant \beta$	$\geqslant 2\beta$	$\leqslant \beta$	$\geqslant 2\beta$
$i=2$	$\leqslant \beta$	$\geqslant 2\beta$	$\leqslant \beta$	$\geqslant 2\beta$	$\leqslant \beta$
$i=3$	$\geqslant 2\beta$	$\leqslant \beta$	$\geqslant 2\beta$	$\leqslant \beta$	$\geqslant 2\beta$
$i=4$	$\geqslant \beta$ et $\leqslant 2\beta$	$\geqslant 2\beta$	$\leqslant \beta$	$\geqslant 2\beta$	$\leqslant \beta$

Ce tableau définit si les distances euclidiennes entre les blocs deux à deux sont inférieures ou supérieures à un seuil donné. Par exemple, $\beta \leqslant d_E(\mathfrak{B}_0^{(n)}, \mathfrak{B}_4^{(n)}) \leqslant 2\beta$. A l'aide de ce tableau, nous donnons le détail des opérations lors des étapes 1 et 2 de l'algorithme.

Étape 1 :

— $R_0 = \mathfrak{B}_0^{(n)}$ est le premier mot de référence.

Entrée dans la boucle *tant que* à la ligne 9

boucle 1

- $R_0 = \mathfrak{B}_0^{(n)}$ est ajouté à l'ensemble des mots de référence : $\mathcal{R}^{(n,\beta)}$
- D'après la première ligne du tableau ($i=0$), $\mathcal{I}_\beta(0) = \{0, 2\}$. Il en résulte que seuls $\mathfrak{B}_0^{(n)}$ et $\mathfrak{B}_2^{(n)}$ appartiennent à la classe de $\mathcal{R}_0$. $\mathfrak{B}_0^{(n)}$ de manière triviale car $R_0 = \mathfrak{B}_0^{(n)}$, et $\mathfrak{B}_2^{(n)}$ car $d_E(\mathfrak{B}_0^{(n)}, \mathfrak{B}_2^{(n)}) \leqslant \beta$. Finalement : $\mathcal{F}_\beta^{(n)} = \{0, 2\}$.
- De même, d'après la première ligne du tableau, $\mathcal{I}_{2\beta}(0) = \{0, 2, 4\}$. Il en résulte que $\mathfrak{B}_0^{(n)}$, $\mathfrak{B}_2^{(n)}$ et $\mathfrak{B}_4^{(n)}$ ne peuvent pas être un nouveau mot de référence lors de l'étape 1. Plus spécifiquement, $\mathfrak{B}_0^{(n)}$ et $\mathfrak{B}_2^{(n)}$ sont déjà classés, ils sont donc exclus. Et $\mathfrak{B}_4^{(n)}$ est trop près de R_0. Finalement : $\mathcal{F}_{2\beta}^{(n)} = \{0, 2, 4\}$.
- Le choix de l'indice du nouveau mot de référence pour la boucle suivante se fait ainsi : $\min(\{0,1,2,3,4\} \setminus \mathcal{F}_{2\beta}^{(n)}) = \min(\{0,1,2,3,4\} \setminus \{0,2,4\}) = 1$. Le nouveau mot de référence est $\mathfrak{B}_1^{(n)}$.

boucle 2 :
- $R_1 = \mathfrak{B}_1^{(n)}$ est ajouté à l'ensemble des mots de référence : $\mathcal{R}^{(n,\beta)}$
- D'après la deuxième ligne du tableau ($i = 1$), $\mathcal{I}_\beta(1) = \{1, 3\}$. Il en résulte que seuls $\mathfrak{B}_1^{(n)}$ et $\mathfrak{B}_3^{(n)}$ appartiennent à la classe de $\mathcal{R}_1$. Finalement : $\mathcal{F}_\beta^{(n)} = \{0, 2\} \cup \{1, 3\} = \{0, 1, 2, 3\}$.
- De même, d'après la deuxième ligne du tableau, $\mathcal{I}_{2\beta}(1) = \{1, 3\}$. Finalement : $\mathcal{F}_{2\beta}^{(n)} = \{0, 2, 4\} \cup \{1, 3\} = \{0, 1, 2, 3, 4\}$.
- Le cardinal de $\mathcal{F}_{2\beta}^{(n)}$ vaut $L_{n_c} = 5$, ce qui signifie que tous les blocs ont été classés ou exclus parce qu'ils était trop près d'un mot de référence (comme pour $\mathfrak{B}_4^{(n)}$). Ceci provoque la sortie de la première boucle *tant que*.

Sortie de la boucle *tant que*

Étape 2 :

— Choix du prochain mot de référence pour la deuxième étape : $\min(\{0, 1, 2, 3, 4\} \setminus \mathcal{F}_\beta^{(n)}) = \min(\{0, 1, 2, 3, 4\} \setminus \{0, 1, 2, 3\}) = 4$. Le nouveau mot de référence est $\mathfrak{B}_4^{(n)}$.

Entrée dans la boucle *tant que* à la ligne 16

boucle 1 :
- $R_3 = \mathfrak{B}_4^{(n)}$ est ajouté à l'ensemble des mots de référence : $\mathcal{R}^{(n,\beta)}$
- D'après la cinquième ligne du tableau ($i = 4$), $\mathcal{I}_\beta(4) = \{2, 4\}$. Finalement : $\mathcal{F}_\beta^{(n)} = \{0, 1, 2, 3\} \cup \{2, 4\} = \{0, 1, 2, 3, 4\}$.
- Le cardinal de $\mathcal{F}_\beta^{(n)}$ vaut $L_{n_c} = 5$, ce qui signifie que tous les blocs ont été classés. Ceci provoque la sortie de la deuxième boucle *tant que*.

Sortie de la boucle *tant que*

Finalement, trois classes ont été créées par ce processus : $\text{Card}(\mathcal{R}^{(n,\beta)}) = \text{Card}(\{R_0, R_1, R_3\}) = \text{Card}(\{\mathfrak{B}_0^{(n)}, \mathfrak{B}_1^{(n)}, \mathfrak{B}_4^{(n)}\}) = 3$.

2.3 Estimation de la dimension du code par collision

Dans cette section nous présentons une méthode d'estimation de la dimension du code k_c. Il est supposé que la longueur de code n_c a été correctement identifiée (à partir de l'algorithme d'identification aveugle présenté précédemment). Avec un processus de classification parfait, chaque classe créée serait une représentation fiable d'un mot de code donné. Cette hypothèse permettrait de voir le nombre de classes comme le nombre de mots de code différents reçus. Par conséquent, avec un temps d'observation infini, la déficience mesurée devrait être de $2^{n_c} - 2^{k_c}$ dans un espace de dimension n_c où il n'y a que 2^{k_c} de mots de code et l'estimation de k_c serait immédiate. Comme le temps d'observation du canal est fini, il est très probable que l'utilisateur

n'ait pas une représentation de chaque mot de code à sa disposition. De plus, le processus de classification ne fournit pas une valeur unique de déficience (mais plusieurs correspondant aux différentes valeurs de seuil testées). Dans ce contexte, une solution est d'utiliser certains résultats du célèbre *problème des anniversaires* en comptant un nombre de *collisions* à l'instar de Bellard et Tillich dans [5]. Une collision survient lorsque la distance entre deux blocs de taille n_c est inférieure à un *seuil de collision* donné. Dans la suite, nous noterons ce seuil β_{col}. En raison de la redondance introduite par le code, le nombre de collisions attendues est lié à k_c.

2.3.1 Espérance mathématique du nombre de collisions

En l'absence de bruit, pour $K = 2^{k_c}$ mots de code possibles, la probabilité que deux blocs choisis aléatoirement proviennent du même mot de code est $\frac{1}{K}$. Pour $L_{n_c} = \lfloor \frac{N}{n_c} \rfloor$ mots de code reçus, il y a $\frac{L_{n_c} \cdot (L_{n_c}-1)}{2}$ distances calculables pour détecter les collisions. Nous notons X_{col} la variable aléatoire comptant le nombre de collisions. Dans ce cas, l'espérance mathématique du nombre de collisions est : $\mathbb{E}(X_{col}) = \frac{L_{n_c} \cdot (L_{n_c}-1)}{2} \cdot \frac{1}{K}$. Dans le cas bruité, la probabilité que deux blocs, $\mathfrak{B}_i^{(n_c)}$ et $\mathfrak{B}_j^{(n_c)}$, entrent en collision est :

$$\mathbf{P}_{col} = \mathbb{P}(d_E(\mathfrak{B}_i^{(n_c)}, \mathfrak{B}_j^{(n_c)}) \leq \beta_{col})$$

Dans ce cas, l'espérance du nombre total de collisions X_{col} devient :

$$\mathbb{E}(X_{col}) = \frac{L_{n_c} \cdot (L_{n_c} - 1)}{2} \cdot \mathbf{P}_{col} \tag{2.17}$$

Par la suite, nous noterons $\mathbf{s}_i = [s(i \cdot n_c), \cdots, s(i \cdot n_c + n_c - 1)]$ un mot de code modulé, $\mathbf{P}_d$ la probabilité de collision attendue et $\mathbf{P}_{fa}$ la probabilité de collision inattendue. D'après le théorème de probabilité totale, la probabilité que deux blocs entrent en collision est donc :

$$\mathbf{P}_{col} \quad = \quad \mathbb{P}(\mathbf{s}_i = \mathbf{s}_j) \cdot \mathbf{P}_d + \mathbb{P}(\mathbf{s}_i \neq \mathbf{s}_j) \cdot \mathbf{P}_{fa} \tag{2.18}$$

avec :

$$\mathbb{P}(\mathbf{s}_i = \mathbf{s}_j) \quad = \quad 1 - \mathbb{P}(\mathbf{s}_i \neq \mathbf{s}_j) = \frac{1}{K}$$

$\mathbf{P}_d$ représente la probabilité que deux blocs entrent en collision étant donné qu'ils sont tous les deux des versions bruitées du même mot de code. L'expression de la probabilité $\mathbf{P}_d$ dérive d'une loi du khi-2 à n_c degrés de liberté :

$$\begin{aligned} \mathbf{P}_d &= \mathbb{P}(d_E(\mathfrak{B}_i^{(n_c)}, \mathfrak{B}_j^{(n_c)}) \leq \beta_{col} | \, \mathbf{s}_i = \mathbf{s}_j) \\ &= \frac{\gamma\left(\frac{n_c}{2}, \frac{\beta_{col}^2}{4\sigma_b^2}\right)}{\Gamma\left(\frac{n_c}{2}\right)} \end{aligned} \tag{2.19}$$

L'annexe A détaille le calcul de l'expression de $\mathbf{P}_d$. Ici, $\Gamma(\cdot)$ et $\gamma(\cdot,\cdot)$ désignent respectivement la fonction gamma et la fonction gamma incomplète. $\mathbf{P}_{fa}$ est la probabilité que deux blocs entrent en collision étant donné qu'ils ne proviennent pas du même mot de code :

$$\mathbf{P}_{fa} = \mathbb{P}(d_E(\mathfrak{B}_i^{(n_c)}, \mathfrak{B}_j^{(n_c)}) \leq \beta_{col} | \, \mathbf{s}_i \neq \mathbf{s}_j) \tag{2.20}$$

Puisque la distribution des mots de code est inconnue, il n'est pas possible d'avoir une expression pour $\mathbf{P}_{fa}$. Cependant, $\mathbf{P}_{fa}$ est supposée négligeable pour $\beta = \beta_{col}$, tout particulièrement quand le niveau de bruit est faible. Par conséquent, on en déduit l'approximation suivante :

$$\mathbf{P}_{col} \approx \frac{1}{K} \cdot \mathbf{P}_d \tag{2.21}$$

Enfin, à partir de la séquence reçue, on mesure le nombre empirique de collisions :

$$\hat{x}_{col} = \text{Card}\left(\{(i,j)_{1 \leqslant i < j \leqslant L_{n_c}} : d_E(\mathfrak{B}_i^{(n_c)}, \mathfrak{B}_j^{(n_c)}) \leqslant \beta_{col}\}\right) \tag{2.22}$$

Par conséquent, à partir des équations (2.17) et (2.21) :

$$\hat{x}_{col} \approx \frac{L_{n_c} \cdot (L_{n_c} - 1)}{2} \cdot \frac{1}{K} \cdot \mathbf{P}_d \tag{2.23}$$

Etant donné que $K = 2^{k_c}$ et que k_c est un entier, on en déduit l'estimateur suivant pour la dimension du code :

$$\hat{k}_c = \left\lfloor \log_2 \left(\frac{L_{n_c} \cdot (L_{n_c} - 1)}{2 \cdot \hat{x}_{col}} \cdot \mathbf{P}_d \right) \right\rceil \tag{2.24}$$

où $\lfloor x \rceil$ est l'entier le plus proche de x. Le nombre de collisions mesurées $\hat{x}_{col}$ dépend d'un seuil de collision β_{col} qui est relatif à la probabilité $\mathbf{P}_d$.

2.3.2 Choix du seuil de collision

D'après l'équation 2.24, l'estimation de la dimension du code dépend du nombre de collisions observées. Pour obtenir un nombre représentatif de collisions, il faut déterminer le seuil sur la distance en dessous duquel deux mots de code initialement identiques entrent effectivement en collision. Le choix de ce seuil est définie comme suit :

$$\beta_{col} \triangleq \arg \max_{\beta \in [\beta_{min}, \beta_{max}]} f\left(\frac{\beta^2}{2 \cdot \sigma_b^2}, n_c \right) \tag{2.25}$$

où $f(x,n)$ est la fonction de densité de probabilité correspondant à la distribution du khi-2 à n degrés de liberté, i.e. :

$$f(x,n) = \frac{\left(\frac{1}{2}\right)^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)} (x)^{\frac{n}{2}-1} e^{-\frac{x}{2}} \tag{2.26}$$

L'égalité (2.25) définit β_{col} comme la distance maximisant la densité des mots de code qui entrent en collision. Même si $\mathbf{P}_{fa}$ est inconnue, cette valeur pour β_{col} assure que le nombre de collisions

attendues est grand. Plus le niveau de bruit est important, plus il est nécessaire que le seuil de collision soit élevé. En effet, lorsque la quantité de bruit augmente, la probabilité qu'un mot de code s'écarte de sa position d'origine augmente. Il en résulte une augmentation de la distance moyenne entre deux mots de code identiques. Toutefois, le seuil augmentant, la probabilité de collisions inattendues augmente aussi, ce qui conduit à des erreurs d'estimation.

2.4 Résultats de simulation

Dans cette section, nous évaluons les performances de notre algorithme d'identification. Pour illustrer nos résultats, nous utilisons trois codes LDPC :

- $\mathcal{C}_1 : \mathcal{C}(25, 10)$ de rendement $\frac{2}{5}$
- $\mathcal{C}_2 : \mathcal{C}(25, 20)$ de rendement $\frac{4}{5}$
- $\mathcal{C}_3 : \mathcal{C}(15, 10)$ de rendement $\frac{2}{3}$

Le choix de ces trois codes est motivé par la volonté d'appréhender l'influence du rendement du code sur les performances de notre méthode d'identification. Les matrices génératrices et de parité des trois codes LDPC testées sont disponibles en Annexe A. Une représentation de la distribution des distances de Hamming est également donnée dans cette annexe.

Pour mesurer les effets du niveau de bruit, nous utiliserons le rapport de l'énergie par bit (utile) E_b et de celle du bruit N_0 : $E_b/N_0 = E_s/(r_c.N_0)$, avec E_s est l'énergie par symbole et $r_c = \frac{k_c}{n_c}$. Dans la suite, les probabilités de bonne identification des paramètres des codes correcteurs sont obtenues à partir de 1000 itérations de Monte-Carlo. A chaque itération, une nouvelle séquence codée est générée et est bruitée de manière aléatoire suivant le modèle du canal BBAG.

2.4.1 Identification de la longueur du code

D'après la définition de la DNC (Déf. 7), il est nécessaire de connaître la distribution du nombre de classes dans le cas d'une trame i.i.d. pour calculer les déficiences. En effet, la méthode nécessite la connaissance de $\mathbb{E}(Z^{(n)})$ qui correspond à l'espérance du nombre de classes obtenues pour une trame i.i.d. bruitée. Ces valeurs peuvent être tabulées car elles sont indépendantes du code. Elles dépendent uniquement du niveau de bruit et du nombre de symboles reçus. Ne disposant pas d'expression théorique de $\mathbb{E}(Z^{(n)})$, on l'estime par 1000 itérations de Monte Carlo en appliquant la procédure de classification au cas d'une trame i.i.d. bruitée en utilisant la moyenne empirique comme estimateur de l'espérance. Cette tabulation demande la connaissance de la variance du bruit, σ_b^2. Dans les résultats présentés ci-dessous, nous avons fait l'hypothèse que cette variance était connue. Toutefois, il existe plusieurs méthodes dans la littérature pour estimer le niveau de bruit [67, 73].

Nous noterons $\mathbf{P}_{n_c}$ la probabilité d'une estimation correcte de la longueur du code.

2.4.1.1 Impact du niveau de bruit

Dans cette sous-section, nous observons l'influence du niveau de bruit sur notre méthode d'identification pour différents rendements. Pour les codes $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$, nous comparons la probabilité d'une estimation correcte de la longueur du code ($\mathbf{P}_{n_c}$) pour 1000 mots de code bruités en fonction du niveau de bruit. Les tailles de bloc testées (n) sont comprises entre 10 et 30. Sur la Figure 2.8, cette probabilité de bonne détection, $\mathbf{P}_{n_c}$, est tracée en fonction du niveau de bruit. Le pas pour β est fixé à $\Delta\beta = 0.5$. A partir de la Figure 2.8, nous remarquons que plus le rendement du code est faible, meilleures sont les performances. Tout d'abord, pour la même longueur de code n_c égales (par exemple, $n_c = 25$ pour $\mathcal{C}_1$ et $\mathcal{C}_2$), la distance moyenne entre les classes tend à être plus grande pour le code avec moins de mots de code (ici $\mathcal{C}_1$). Il est donc globalement plus aisé de discriminer les classes. Ensuite, pour la même dimension k_c (par exemple, $k_c = 10$ pour $\mathcal{C}_1$ et $\mathcal{C}_3$), la distance moyenne est plus grande lorsque la longueur du code augmente puisque tous les mots de code non bruités sont initialement sur la surface d'une n_c-sphère de rayon $\sqrt{n_c}$. Ainsi, à dimension égale, la distance moyenne entre les mots de code est plus élevée lorsque n_c augmente.

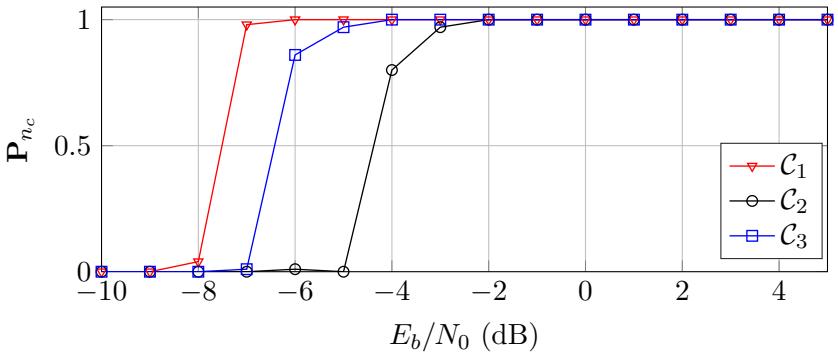


FIGURE 2.8 – Comparaison des performances pour les différents codes correcteurs : $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$

2.4.1.2 Impact de la longueur des trames reçues

Le temps d'observation a un impact significatif sur la probabilité de détection. En effet, plus il y a de données reçues, plus la détection est précise. Pour différents niveaux de bruit, la Figure 2.9 montre la probabilité de détection par rapport au nombre de mots de code reçus L_{n_c} pour les codes $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$. Pour chaque code, nous choisissons un niveau de bruit pour lequel la $\mathbf{P}_{n_c}$ est proche de 1 dans le cas de 1000 mots de code et $\Delta\beta = 0.5$. D'après la Figure 2.8, pour $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$ l'algorithme atteint $\mathbf{P}_{n_c} \approx 1$ pour $E_b/N_0 = -6$ dB, -2 dB et -4 dB, respectivement. Sur la Figure 2.9, la probabilité de bonne détection est tracée en fonction du nombre de mots de code reçus pour le niveau de bruit fixé. Cela, nous permet d'observer les gains (ou pertes) en augmentant (ou en diminuant respectivement) la longueur de la séquence reçue. Nous observons que les trois codes ont le même comportement, l'augmentation du nombre de mots de code reçus permet d'atteindre une meilleur détection.

Finalement, plus la séquence reçue est longue, meilleures sont les performances. Néanmoins, le temps de calcul augmente significativement avec la quantité de mots reçus. En effet, à chaque taille de bloc n testée, le nombre de distances calculées est $\frac{L_n \cdot (L_n - 1)}{2}$.

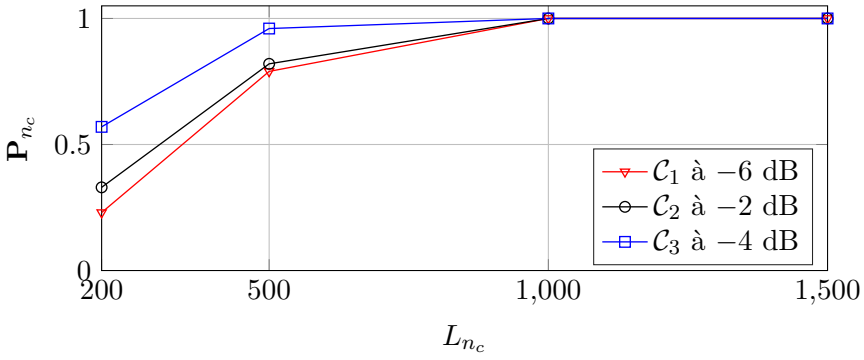


FIGURE 2.9 – Effets de la quantité de mots de code reçus pour $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$

Enfin, l'augmentation de L_{n_c} ne suffit pas à garantir de meilleures performances. En effet, la Figure 2.10 nous montre que la méthode connaît une limite de fonctionnement à -8 dB. Le fait de passer de $L_{n_c} = 1000$ à $L_{n_c} = 2000$ ne permet pas d'augmenter la probabilité de bonne identification. Lorsque le niveau de bruit augmente, les mots bruités s'éloignent de plus en plus de la position du mot de code dont ils sont issus. Quelque soit le choix du seuil, le phénomène de superposition des classes est inévitable, ce qui provoque des erreurs d'identification. Cela se produit lorsque le niveau de bruit est assez élevé et que la distance minimale du code correcteur d'erreurs ne suffit plus pour distinguer deux mots de code différents.

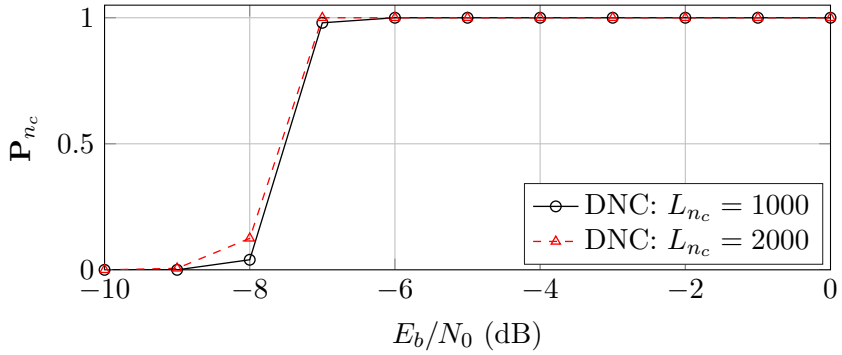


FIGURE 2.10 – Effets de la quantité de mots de code reçus pour $\mathcal{C}_1$ pour différents niveaux de bruit

2.4.1.3 Impact du pas de balayage sur le seuil de classification

La qualité de l'estimation de la longueur du code dépend également du choix du seuil β. Cependant, n'étant pas en mesure de déterminer théoriquement un seuil optimal, nous calculons $\varphi(n, \beta)$ pour plusieurs valeurs de β. Ceci nous contraint au choix d'un pas de balayage $\Delta\beta$. La Figure 2.11 représente l'impact du pas de balayage $\Delta\beta$ sur la probabilité de détection pour les

codes $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$ aux mêmes niveaux de bruit que précédemment et pour 1000 mots de code reçus. Plus le pas $\Delta\beta$ est faible, meilleure est la détection, mais l'effort de calcul demandé est alors plus grand. Le choix du pas sur le seuil est lié au rendement du code correcteur d'erreurs utilisé. Pour un n_c fixe, lorsque le rendement du code augmente, la redondance introduite diminue. Dans ce cas, la discrimination des différentes classes est plus difficile car il y a moins de lacunes dans l'espace de dimension n_c. C'est le cas pour les codes $\mathcal{C}_1$ et $\mathcal{C}_2$ qui sont de même taille ($n_c = 25$). Nous observons sur la Figure 2.11 que l'identification pour le code $\mathcal{C}_2$ est plus sensible au pas $\Delta\beta$.

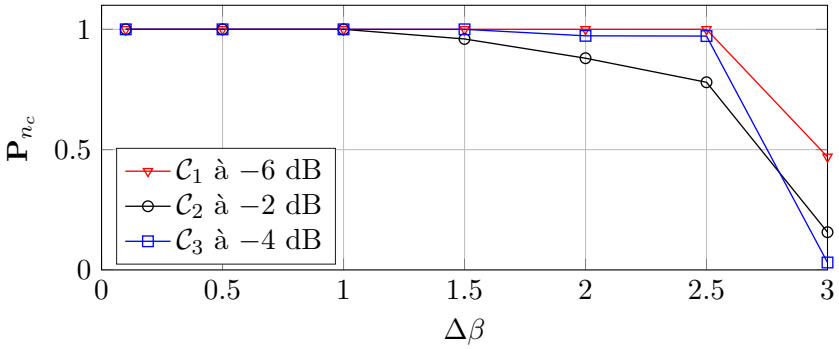


FIGURE 2.11 – Influence du pas de balayage des seuil sur la détection pour $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$

2.4.1.4 Impact de la désynchronisation de trame

Jusqu'à présent, nous avons fait l'hypothèse que la trame reçue débutait par le premier bit d'un mot de code. Nous allons étudier ici l'influence d'une non-synchronisation sur notre algorithme d'identification par déficience du nombre de classes. Cet algorithme repose sur le calcul des distances euclidiennes entre les blocs. Deux blocs sont donc susceptibles d'être séparés dans des classes différentes, même s'ils partagent des parties identiques du même mot de code.

La Figure 2.12 montre l'impact d'une séquence non synchronisée sur les performances de détection des trois codes ($\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$) pour différents niveaux de bruit (−5 dB, 0 dB et 5 dB) et pour 1000 mots de code. Nous notons Δt le pas de synchronisation. Pour les codes $\mathcal{C}_2$ et $\mathcal{C}_3$, comme prévu, notre méthode d'identification fonctionne bien autour de la région de synchronisation (quand Δt tend vers 0 ou vers n_c). Dans les autres régions de synchronisation, notre algorithme ne fonctionne pas. Pour le code $\mathcal{C}_1$, nous remarquons que pour un niveau de bruit raisonnable (0 et 5 dB), notre algorithme n'est pas sensible à la synchronisation. En effet, pour ce code, la méthode est plus performante et ceci quelle que soit la synchronisation Δt. Le code $\mathcal{C}_1$ est un code quasi-cyclique de paramètre 5. La structure quasi-cyclique de ce code a une influence particulière sur la robustesse de notre algorithme vis-à-vis de la synchronisation. Nous pouvons donc en conclure que la robustesse de la méthode à la synchronisation de trame dépend du type de code.

Dans la littérature, il existe différentes méthodes de synchronisation trame [43, 44]. Mais il serait également possible de trouver le début d'un mot de code, en exécutant l'algorithme pour différentes valeurs de Δt.

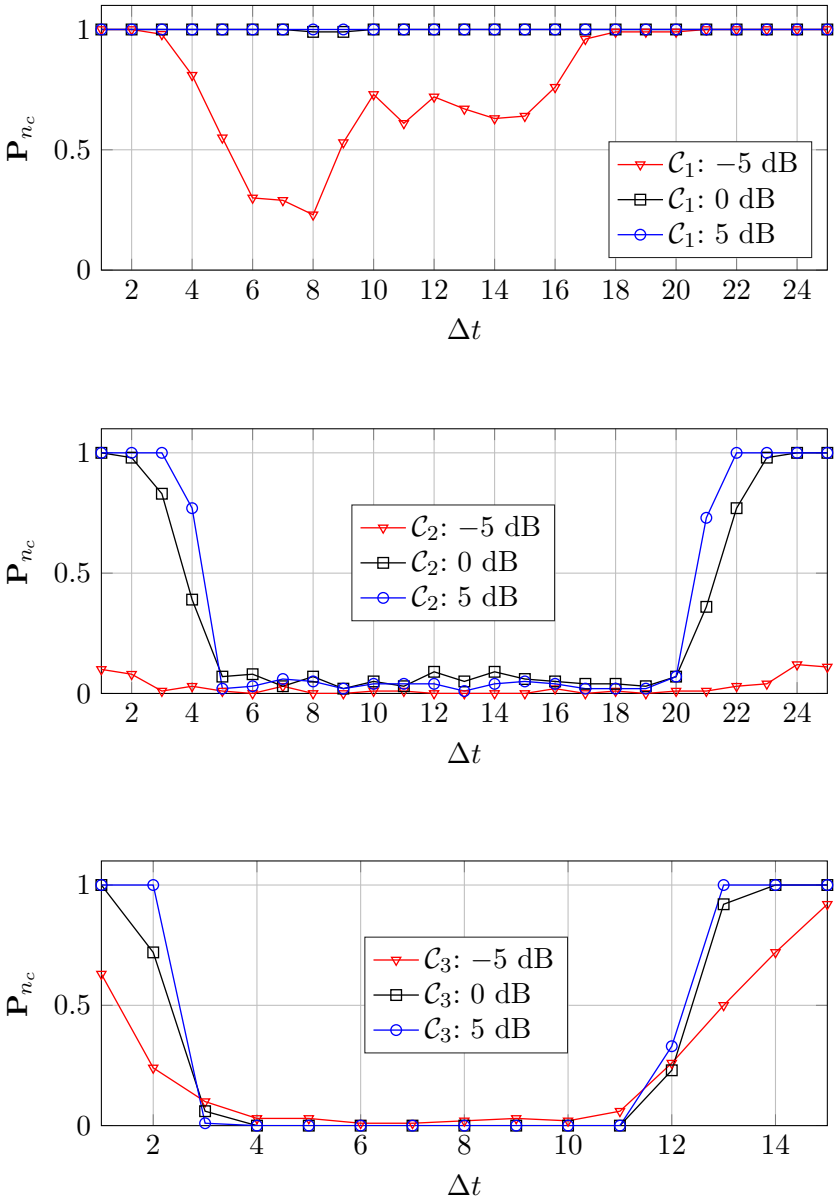


FIGURE 2.12 – Effet de la synchronisation de trame sur la détection de la longueur du code

2.4.1.5 Comparaison de notre algorithme avec l'algorithme de déficience de rang

Pour les codes C_1, C_2 et C_3, nous comparons notre méthode (DNC) avec la méthode par déficience de rang (DR) (section 1.3.1) pour $L_{n_c} = 1000$ mots de code reçus. La méthode DR est appliquée pour 5 itérations de virtualisation. A chaque itération, l'algorithme opère une permutation aléatoire sur les lignes de la matrice d'interception comme expliqué dans la

section 1.3.1. A partir de la Figure 2.8, nous observons que notre méthode basée sur les distances euclidiennes surpasse la méthode de réduction de Gauss quel que soit le code choisi. Pour une probabilité d'identification correcte proche de 0.8, le gain avec notre méthode est d'environ 4 dB pour C_1, et d'environ 3 dB pour C_2 et C_3.

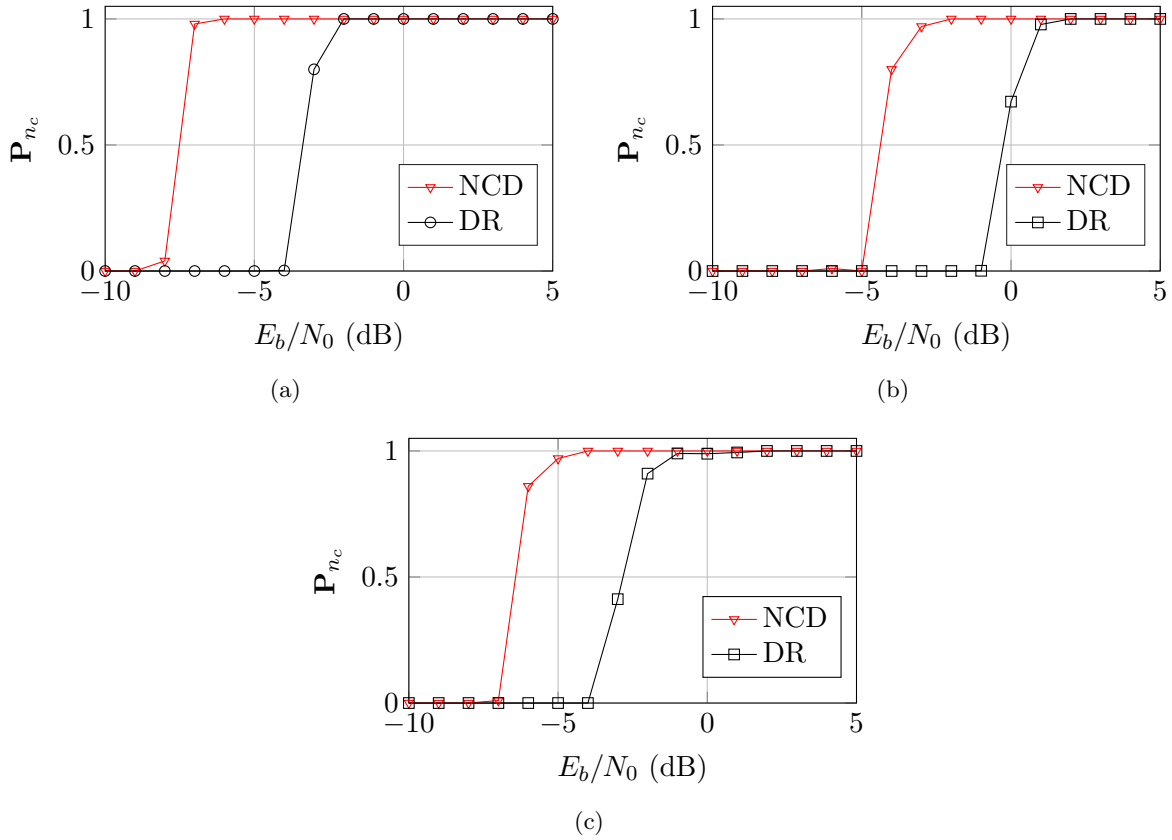


FIGURE 2.13 – Comparaison des méthodes par déficience du nombre de classe et par déficience du rang (triangles rouges et ronds noirs respectivement) pour les codes C_1 (2.13(a)), C_2 (2.13(b)) and C_3 (2.13(c))

2.4.2 Estimation de la dimension du code

Dans cette partie, nous allons analyser les performances de notre algorithme d'estimation de la dimension du code. Nous ferons l'hypothèse que la longueur du code n_c aura été correctement identifiée au préalable par la méthode de déficience du nombre de classes. Nous noterons $\mathbf{P}_{k_c}$, la probabilité de correctement estimer k_c et les analyses porteront sur les trois codes précédents : $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$. La connaissance de la variance du bruit σ_b^2 est nécessaire à l'identification de la dimension du code. Dans la suite, nous supposons que cette variance est connue et nous proposons d'étudier l'impact d'une estimation erronée de ce paramètre sur les performances de l'identification.

2.4.2.1 Impact du niveau de bruit

La Figure 2.14 permet d'observer l'impact du niveau bruit sur $\mathbf{P}_{k_c}$ pour 2000 mots de code. Nous remarquons que notre algorithme est plus efficace pour le code $\mathcal{C}_1$. Avec la même dimension $k_c = 10$, l'algorithme est moins performant pour le code $\mathcal{C}_3$ qui a un rendement plus élevé que $\mathcal{C}_1$.

Dans le cas du code $\mathcal{C}_2$, la quantité de mots de code reçus $\lfloor \frac{N}{n_c} \rfloor = 2000$ est trop faible. Quand E_b/N_0 est grand, il n'y a pas assez de collisions ou même pas de collision du tout pour estimer k_c. De plus, en raison de son rendement élevé, une légère augmentation du niveau de bruit fait entrer en collision certains mots de code même s'ils sont initialement différents. Dans ce cas, la $\mathbf{P}_{fa}$ est non négligeable. En effet, puisque $K = 2^{20}$ la $\mathbf{P}_{fa}$ a une plus grande influence sur la probabilité $\mathbf{P}_{col}$ d'après l'equation (2.18).

Dans ce qui suit, nous donnons plus de détails sur le temps d'observation et sur l'impact de $\mathbf{P}_{fa}$.

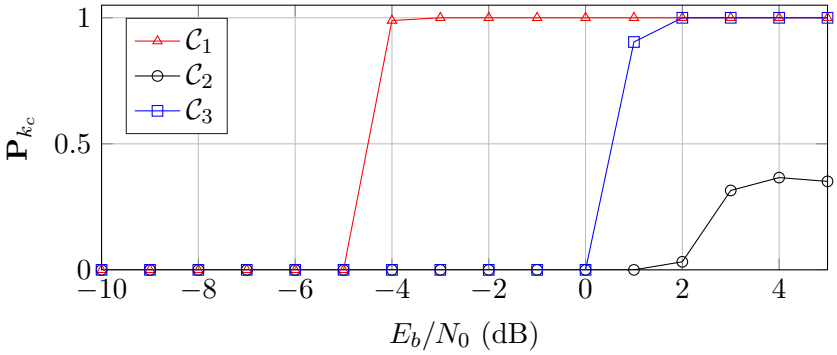


FIGURE 2.14 – Probabilité d'estimation de la dimension du code en présence de bruit pour $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$ avec 2000 mots de code

2.4.2.2 Impact de la longueur des trames reçues

En plus de dépendre du niveau de bruit, l'efficacité de l'estimateur de la dimension du code dépend de la quantité de mots de code reçus. Comme le montre la Figure 2.15, lorsque la quantité de mots de code reçus augmente la probabilité d'avoir au moins une collision converge vers 1. Nous pouvons également voir que le bruit a un impact sur la quantité de mots de code nécessaire pour estimer correctement la dimension du code. En effet, l'ajout de bruit augmente la $\mathbf{P}_{fa}$. Pour le code $\mathcal{C}_1$, 1000 mots de code sont suffisant pour estimer parfaitement k_c à 4 et 5 dB. En revanche, pour le code $\mathcal{C}_3$, si 1000 mots sont suffisants à 5 dB il en faut 1500 pour atteindre la même performance à 4 dB. Pour le code $\mathcal{C}_1$, la probabilité de bonne estimation de k_c est également tracée à -4 dB. Dans ce cas, pour atteindre une estimation parfaite, il est nécessaire d'avoir 2000 mots.

Cette différence de comportement est due au rendement respectif de chaque code. En effet, pour deux codes ayant la même dimension k_c (par exemple $\mathcal{C}_1$ et $\mathcal{C}_3$), il est plus facile d'estimer

k_c pour le code ayant le rendement le plus bas (ici, $\mathcal{C}_1$) puisque la distance moyenne augmente lorsque le rendement diminue.

Il est à noter que la performance pour $\mathcal{C}_2$ n'est pas donnée puisque nous devrions revevoir beaucoup plus que 2000 mots de code afin d'estimer k_c pour ce code (en raison de sa longueur et de son rendement).

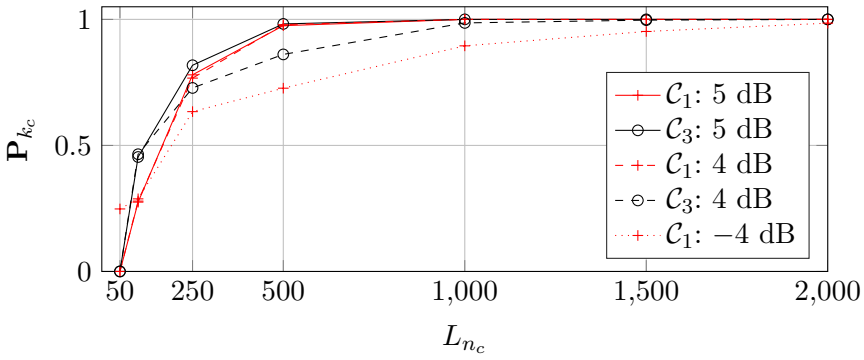


FIGURE 2.15 – Probabilité de bonne estimation de k_c pour différentes quantités de mots de code et niveau de bruit pour $\mathcal{C}_1$ et $\mathcal{C}_3$

2.4.2.3 A propos de la probabilité de fausse alarme $\mathbf{P}_{fa}$

Dans la sous-sections 2.3.1, nous avons proposé une approximation pour $\mathbf{P}_{col}$ en négligeant $\mathbf{P}_{fa}$ pour des niveaux de bruit faibles. En effet, le nombre de fausses collisions tend vers 0 lorsque la quantité de bruit diminue. Pour illustrer cela, la Figure 2.16 montre comment les fausses et vraies collisions affectent la probabilité totale de collision pour différentes quantités de bruit. Pour ce faire, nous comparons $(1-\frac{1}{K})\cdot\mathbf{P}_{fa}$ et $\frac{1}{K}\cdot\mathbf{P}_d$ pour le code $\mathcal{C}_1$ avec $L_{n_c}=\lfloor\frac{N}{n_c}\rfloor=2000$ mots de code reçus. Puisque nous avons une expression théorique pour $\mathbf{P}_d$ et une estimation de la probabilité de collision $\hat{\mathbf{P}}_{col}=\frac{2\hat{x}_{col}}{L_{n_c}\cdot(L_{n_c}-1)}$, nous accédons facilement à une valeur estimée de $\mathbf{P}_{fa}$ à partir de 2.18. Pour des niveaux de bruit peu élevées, l'impact de la probabilité de fausse collision est comparativement négligeable dans le calcul de $\mathbf{P}_{col}$ par rapport à la probabilité de vraie collision. Si nous comparons ses résultats avec ceux de la Figure 2.14, nous remarquons que tant que $(1-\frac{1}{K})\cdot\mathbf{P}_{fa}>\frac{1}{K}\cdot\mathbf{P}_d$, la probabilité de bonne détection est à 0 (de -10 à -5 dB) et qu'ensuite cette probabilité passe à 1. La proportion de fausses collisions est trop élevée entre -10 et -5 dB et empêche d'estimer k_c.

Maintenant, nous effectuons la même analyse pour le code $\mathcal{C}_2$. Nous avons vu que l'estimateur était moins efficace pour le code $\mathcal{C}_2$ avec 2000 mots de code. Nous remarquons sur la Figure 2.17 que $(1-\frac{1}{K})\cdot\mathbf{P}_{fa}\geq\frac{1}{K}\cdot\mathbf{P}_d$ pour tous les niveaux de bruit. Ce qui explique pourquoi la probabilité de bonne détection de k_c est proche de 0 sur la Figure 2.14. De plus, quelque soit le niveau de bruit, $\mathbf{P}_{col}\approx(1-\frac{1}{K})\cdot\mathbf{P}_{fa}$. D'après la Figure 2.14, la probabilité de bonne détection est différente de 0 pour un niveau de bruit compris entre 2 et 5 dB. Dans ces circonstances, la plupart des collisions survenant sont de fausses collisions, ce qui explique la mauvaise détection de k_c. Ce

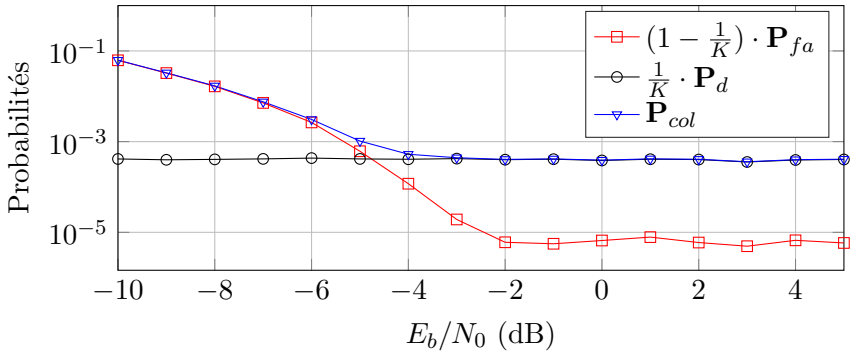


FIGURE 2.16 – Comparaison des influences respectives de $\mathbf{P}_{fa}$ et $\mathbf{P}_d$ sur la probabilité de collision pour différentes valeurs de E_b/N_0 avec $L_{n_c} = 2000$ pour $\mathcal{C}_1$

comportement est dû au rendement élevé de ce code. En effet, plus le rendement du code est élevé, plus la distance minimale est faible. Toute altération due au bruit implique alors de fausses collisions avec une probabilité élevée.

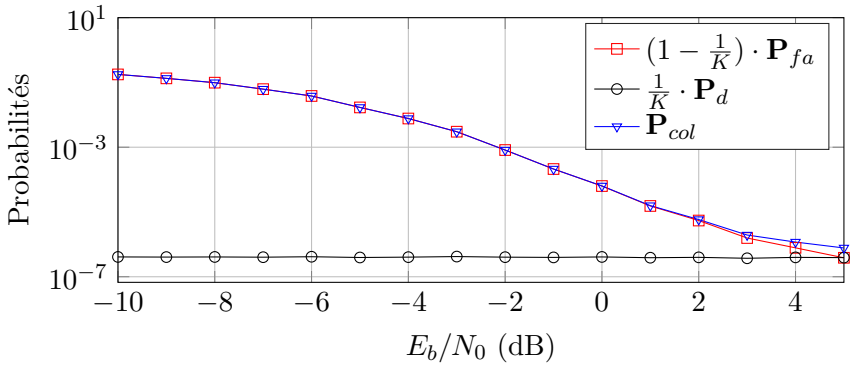


FIGURE 2.17 – Comparaison des influences respectives de $\mathbf{P}_{fa}$ et $\mathbf{P}_d$ sur la probabilité de collision pour différentes valeurs de E_b/N_0 avec $L_{n_c} = 2000$ pour $\mathcal{C}_2$

2.4.2.4 A propos de l'estimation de la variance du bruit

Jusqu'à présent nous avons fait l'hypothèse que la variance du bruit était connue or elle a un impact sur le choix du seuil β_{col} pour l'estimation de la dimension du code. Par conséquent, une valeur estimée erronée de $\hat{\sigma}_b^2$ pour la variance du bruit pourrait affecter cette estimation. Nous avons effectué des tests avec le code $\mathcal{C}_1$ pour évaluer l'impact d'une mauvaise estimation de σ_b^2 sur notre algorithme d'identification. Pour cela, nous avons artificiellement généré des erreurs $\epsilon_{\hat{\sigma}_b^2} = \pm 1\%, \pm 2\%, \pm 2\%, \pm 5\%, \pm 10\%$ ($\epsilon_{\hat{\sigma}_b^2} = \pm 1\%$ signifie que $\hat{\sigma}_b^2 = 0.99 \cdot \sigma_b^2$ ou $\hat{\sigma}_b^2 = 1.01 \cdot \sigma_b^2$). La Table 2.1 résume les résultats en donnant la probabilité de bonne estimation $\mathbf{P}_{k_c}$ à 0 dB.

D'après ce tableau, nous pouvons en conclure que des altérations mineures, de l'ordre des ±2 %, dans l'estimation de la variance du bruit n'ont pas d'effets sur les performances de notre méthode.

TABLE 2.1 – Probabilité d'estimation correcte de k_c pour une valeur de la variance du bruit erronées

$\epsilon_{\hat{\sigma}_b^2}$	0 %	±1 %	±2 %	±5 %	±7.5%	±10%
$\mathbf{P}_{k_c}$	1	1	1	0.7	0.14	0

2.5 Conclusion

La méthode présentée ici permet d'identifier la longueur et la dimension d'un code en aveugle à partir d'informations souples reçues. L'identification de la longueur du code repose sur une méthode de classification utilisant les distances euclidiennes comme critère. Cette méthode met en évidence une différence de comportement dans le nombre de classes créées pour une séquence codée d'une part et pour une séquence i.i.d. d'autre part. En présence d'un code, le nombre théorique de classes est limité par la longueur des mots d'information. Pour récupérer cette longueur, nous avons adapté un résultat du problème des anniversaires et proposé un estimateur. Le nombre de collisions attendues dépend de la dimension du code. L'estimation de la longueur et des dimensions dépend du temps d'observation et du niveau de bruit. Nous avons montré que notre méthode permettant d'identifier la taille du code offrait de très bonne performance de détection pour des codes courts. Toutefois, nous avons également mis en avant que la complexité de mise en oeuvre pour des codes longs deviendrait rédhibitoire.

Étant donné que la méthode par déficience de classes est particulièrement efficace pour des codes courts, il serait intéressant d'observer les résultats obtenus pour son application à des codes convolutifs. Toutefois, il est envisageable que des longueurs de contrainte trop grandes empêchent le bon fonctionnement de la classification. Par ailleurs, il serait possible d'étudier l'identification de la longueur d'un entrelaceur, ou l'identification des paramètres du codeur équivalent à des codes concaténés ou à des codes poinçonnés. Toutes ces opérations génèrent certainement, elles aussi, des espaces euclidiens comportant des lacunes. C'est sur celles-ci que s'appuient le critère d'identification de notre méthode. Enfin, la méthode de classification utilisée est somme toute «naïve». En effet, le choix des mots de référence est laissé au hasard, il est sans doute possible d'améliorer le processus de classification en choisissant des mots de référence plus fiables.

Comme dit précédemment, la complexité de cette méthode peut être rédhibitoire. Dans le chapitre suivant, nous souhaitons baser notre critère d'identification sur l'étude de la distribution des distances euclidiennes pour nous affranchir du processus de classification.

Chapitre 3

Identification de la longueur du code à partir des distances euclidiennes minimales

Dans le chapitre précédent, nous avons proposé une idée intuitive de l'observation de la distribution de mots dans un espace euclidien. Le critère de distance permet de mettre en évidence le caractère lacunaire de cet espace en présence d'un code correcteur d'erreurs. A travers l'exemple de la Figure 2.2, nous avons mis en évidence les particularités de la distribution des distances de Hamming en présence d'un code correcteur d'erreurs. Dans le cas de l'utilisation d'informations souples, nous nous intéressons à la distribution des distances euclidiennes qui est également affectée par la structure du code. Pour le code de Hamming $\mathcal{C}(7,4)$ et un niveau de bruit de 15 dB, nous avons représenté cette distribution des distances sur la Figure 3.1. Comme précédemment, nous comparons le cas codé avec le cas i.i.d. pour des mots de taille 7 et le même niveau de bruit. Dans le cas du code de Hamming, en bleu, nous discernons bien les 4 modes de la distribution des distances, centrés aux valeurs proche de 0.4, 3.5, 4 et 5.3, qui correspondent aux 4 distances de Hamming existant pour ce code, respectivement 0, 3, 4 et 7. Encore une fois, les deux distributions, cas codé et cas i.i.d., sont très différentes. Nous nous intéressons particulièrement à l'effet de la distance minimale d_{min}. La distance minimale de Hamming du code de Hamming $\mathcal{C}(7,4)$ est $d_{min} = 3$. En l'absence de bruit et en présence du code, nous sommes donc assurés que la distance euclidienne entre deux mots distincts est supérieure ou égale à $2\sqrt{3} \approx 3.5$. Pour un niveau de bruit faible (par exemple à 15 dB), la probabilité qu'il existe une distance entre le premier et le deuxième mode est très faible. La méthode présentée dans ce chapitre tire parti de cette particularité.

Dans ce chapitre, nous présentons donc un processus permettant d'identifier en aveugle la longueur d'un code en bloc linéaire sans connaissance a priori du codeur. Cette méthode a été décrite dans l'article [10]. Nous supposons que le signal acquis a été démodulé et synchronisé avec succès. Par conséquent, nous considérons un flux modulé en modulation de phase à 2 états (MP-2) à la sortie d'un canal à bruit blanc gaussien additif (BBAG). Ce choix de modélisation permet de générer simplement des mesures de fiabilités des bits reçus (comme dans le chapitre

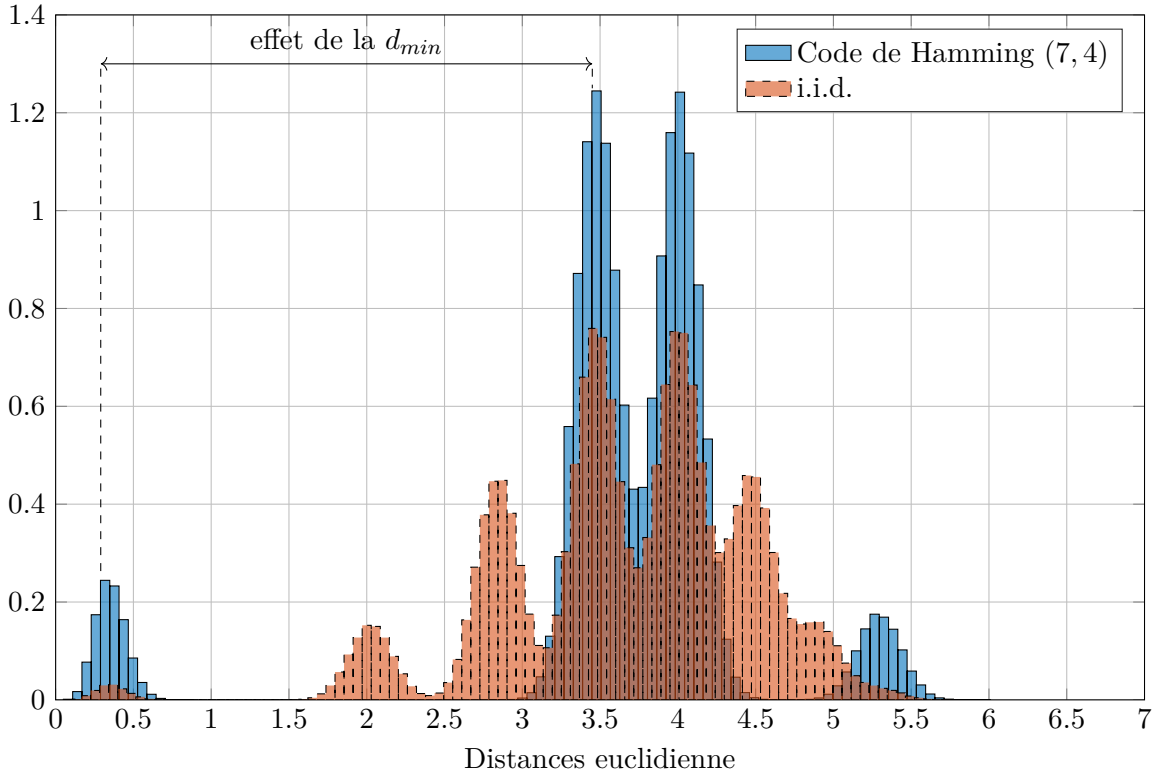


FIGURE 3.1 – Comparaison de la distribution des distances euclidiennes à 15 dB pour des mots du code de Hamming $(7, 4)$ et des mots i.i.d. de longueur 7

précédent). A partir de ce flux, nous créons des blocs qui sont comparés deux à deux grâce à la métrique de la distance euclidienne. Ces distances sont ordonnées dans une matrice dont la forme est particulière lorsque la taille du bloc correspond à la longueur du code. De là, nous parvenons à mettre en évidence cette particularité pour identifier la longueur du code. Dans les sections suivantes, nous proposons une méthode pour estimer la longueur d'un code linéaire. Tout d'abord, nous définissons ce que sont les matrices de distances et la manière dont elles sont ordonnées. Ensuite, nous exhibons un estimateur de la longueur du code dépendant de la distribution des valeurs des distances dans les matrices. Enfin, quelques tests et résultats donnent un aperçu des performances de cette méthode.

3.1 Identification de la longueur du code

3.1.1 Définition des matrices de distances ordonnées

Afin de déterminer la longueur du code, nous cherchons à détecter une différence de comportement entre une séquence i.i.d. et une séquence codée $\mathbf{y}$. A cette fin, la séquence $\mathbf{y}$ de N échantillons bruités reçus est divisée en L_n blocs contigus $\mathfrak{B}_i^{(n)}$ de longueur n comme décrit par

l'égalité 3.1, avec $L_n = \left\lfloor \frac{N}{n} \right\rfloor$ et $i \in [\![0, L_n - 1]\!]$.

$$\mathfrak{B}_i^{(n)} = [y(i \cdot n), \cdots, y(i \cdot n + n - 1)] \tag{3.1}$$

A partir des blocs créés, nous construisons une matrice de distance $\mathbf{B}^{(n)}$. Notons $\mathbf{b}_{i,j}^{(n)}$, l'élément sur la i-ième ligne et sur la j-ième colonne de cette matrice, $\mathbf{B}^{(n)}$ est défini comme suit :

$$\begin{aligned} \mathbf{B}^{(n)} &= (\mathbf{b}_{i,j}^{(n)})_{0 \leqslant i,j \leqslant L_n - 1} \\ \mathbf{b}_{i,j}^{(n)} &= d_E^2(\mathfrak{B}_i^{(n)}, \mathfrak{B}_j^{(n)}) \end{aligned} \tag{3.2}$$

Définie de cette façon, $\mathbf{B}^{(n)}$ est une matrice symétrique de diagonale nulle. Dans cette configuration, la i-ième ligne de $\mathbf{B}^{(n)}$ correspond à l'ensemble des distances entre le bloc $\mathfrak{B}_i^{(n)}$ et les autres blocs créés. Chaque ligne correspond donc à une mesure de la distribution des distances du «point de vue» d'un bloc.

De plus, il est à noter que pour un nombre fixe de symboles reçus N, la taille de la matrice diminue quand n augmente. A partir de $\mathbf{B}^{(n)}$, on obtient une matrice des distances ordonnées $\mathbf{D}^{(n)}$ par les deux étapes suivantes :

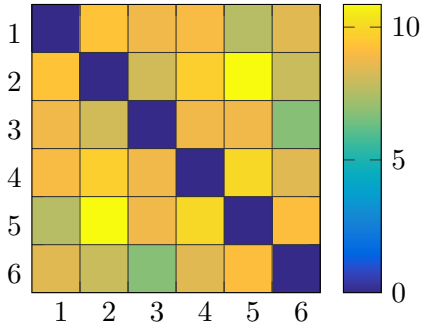
1. Pour tout i dans $[\![0, L_n - 1]\!]$, les éléments de la i-ième ligne de $\mathbf{B}^{(n)}$ sont ordonnés de manière croissante pour former la i-ième ligne de $\mathbf{D}^{(n)}$. A ce stade, la première colonne de $\mathbf{D}^{(n)}$ est nulle ($\forall i \in [\![0, L_n - 1]\!]$, $\mathbf{b}_{i,i}^{(n)} = 0$) et peut donc être retirée. Ainsi, la matrice résultante est de taille $(L_n \times L_n - 1)$.
2. Chaque colonne de cette matrice est triée par ordre croissant de sorte que $\mathbf{d}_{i,j}^{(n)} \leqslant \mathbf{d}_{i+1,j}^{(n)}$ pour tout j dans $[\![0, L_n - 2]\!]$. Enfin, $\mathbf{d}_{0,0}^{(n)} = \min_{i<j} \mathbf{d}_{i,j}^{(n)}$ et $\mathbf{d}_{L_n-1,L_n-2}^{(n)} = \max_{i<j} \mathbf{d}_{i,j}^{(n)}$. De plus, $\mathbf{d}_{0,L_n-1}^{(n)} = \max_i \min_{j \neq i} \mathbf{d}_{i,j}^{(n)}$.

Après ce processus, chaque colonne de la matrice $\mathbf{D}^{(n)}$ est une mesure de la statistique d'ordre sur les distances euclidiennes. Plus précisément, la i-ième colonne de la matrice $\mathbf{D}^{(n)}$ correspond à une mesure de la distribution de la i-ième statistique d'ordre de la distribution des distances euclidiennes entre des blocs de taille n. L'Exemple 3.1.1. détaille la création de $\mathbf{D}^{(n)}$ pour 6 mots bruités.

Exemple 3.1.1.

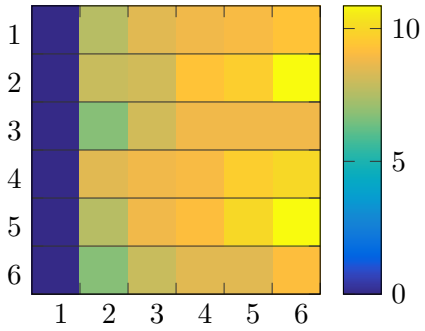
> Considérons 6 mots bruités de longueur 25 dont les positions relatives dans l'espace euclidien sont décrites par la matrice de distance suivante :

$\mathbf{B}^{(25)}$



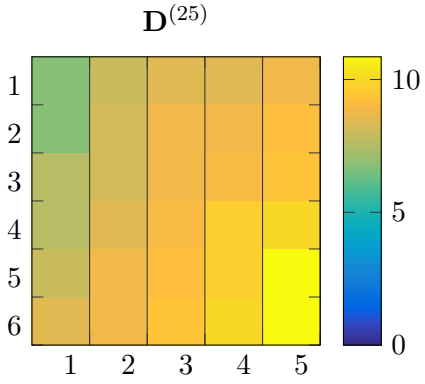
Lors de la première étape, les éléments de chaque ligne de D sont ordonnées de manière croissante pour obtenir la matrice suivante :

$\mathbf{D}^{(25)}$



La première colonne de $\mathbf{D}^{(25)}$ est retirée et lors de la deuxième étape, les éléments de chaque colonne de $\mathbf{D}^{(25)}$ sont ordonnés de manière croissante de sorte que la matrice résultante soit de la forme :

$\mathbf{D}^{(25)}$



Afin de comparer la distribution des distances euclidiennes du cas codé et du cas i.i.d., nous appliquons le même processus à une séquence i.i.d. pour créer la matrice ordonnée correspondante de taille $(L_n \times L_n - 1)$: $\mathbf{X}^{(n)}$. Lorsque nous comparons les éléments $\mathbf{D}^{(n)}$ et $\mathbf{X}^{(n)}$, leurs distributions sont assez similaires pour la plupart des valeurs de n sauf pour $n = n_c$. En effet, la

distribution des mots de code a une grande influence sur la distribution des distances dans les colonnes de $\mathbf{D}^{(n_c)}$.

Exemple 3.1.2.

Par exemple, considérons un code LDPC de longueur $n_c = 25$, de dimension $k_c = 10$ et de distance minimale $d_{min} = 6$. La Figure 3.2 permet la comparaison des ordres de grandeur des distances dans les cinq premières colonnes de $\mathbf{D}^{(n)}$ et $\mathbf{X}^{(n)}$ lorsque la taille du bloc est $n = 24$ et $n = n_c = 25$. Pour cet exemple, $N = 25000$ échantillons ont été reçus (soit $L_{n_c} = 1000$ mots de code). L'abscisse représente les indices des colonnes et l'ordonnée représente les indices des lignes. Puisque la distance minimale de Hamming a un impact majeur sur la distribution des distances les plus faibles, sur la Figure 3.2, nous nous concentrons sur les colonnes 1 à 5 de $\mathbf{D}^{(n)}$ et $\mathbf{X}^{(n)}$.

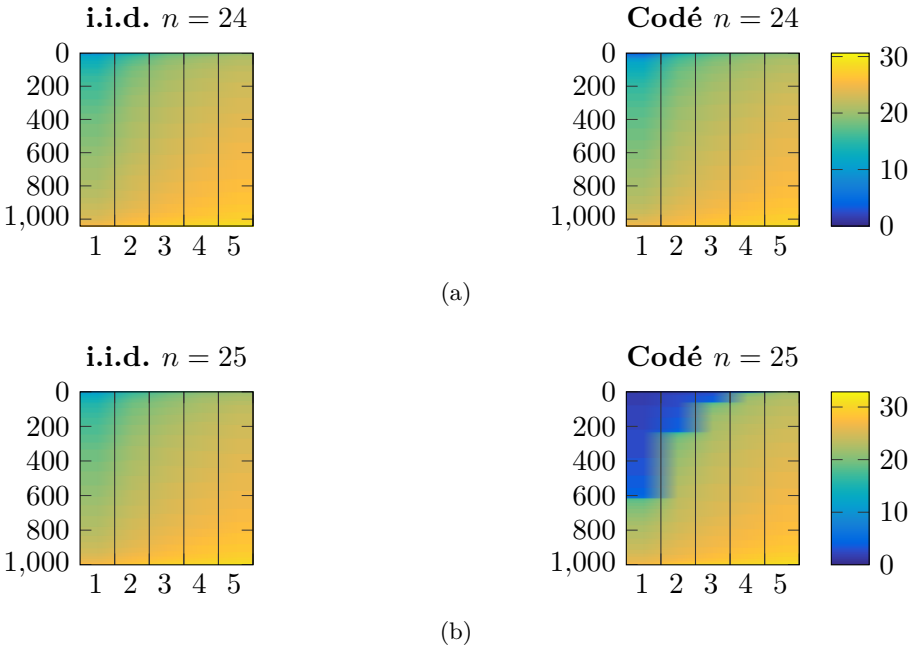


FIGURE 3.2 – Comparaison des premières colonnes de $\mathbf{D}^{(n)}$ en présence du code (à droite) et de $\mathbf{X}^{(n)}$ dans le cas de la trame i.i.d. (à gauche) pour $n = 24$ (3.2(a)) et $n = n_c = 25$ (3.2(b)) à 5 dB

Tout d'abord, lorsque $n = 24$, les distances dans $\mathbf{D}^{(n)}$ ont une distribution similaire au cas i.i.d. ($\mathbf{X}^{(n)}$). Par contre, pour $n = n_c = 25$, $\mathbf{D}^{(n_c)}$ contient une quantité significative de distances euclidiennes faibles dans ses premières colonnes contrairement aux colonnes de $\mathbf{X}^{(n_c)}$. Nous voyons bien que les distributions des distances sont différentes lorsque $n = n_c$.

Dans ce qui suit, nous nous concentrons sur la distribution des valeurs extrêmes de la distance entre deux blocs. Pour une séquence codée, la dimension et la distance minimale de Hamming du code ont des effets particuliers sur cette distribution. Nous utilisons ces particularités pour estimer la longueur du code. La méthode d'estimation présentée ici repose uni-

quement sur la première colonne des matrices $\mathbf{D}^{(n)}$ et $\mathbf{X}^{(n)}$. Cette colonne est notée $\mathbf{d}^{(n)} = [d^{(n)}(0), d^{(n)}(1), \cdots, d^{(n)}(L_n - 1)] = \mathbf{D}^{(n)}_{\cdot,1}$ et $\mathbf{x}^{(n)} = [x^{(n)}(0), x^{(n)}(1), \cdots, x^{(n)}(L_n - 1)] = \mathbf{X}^{(n)}_{\cdot,1}$.

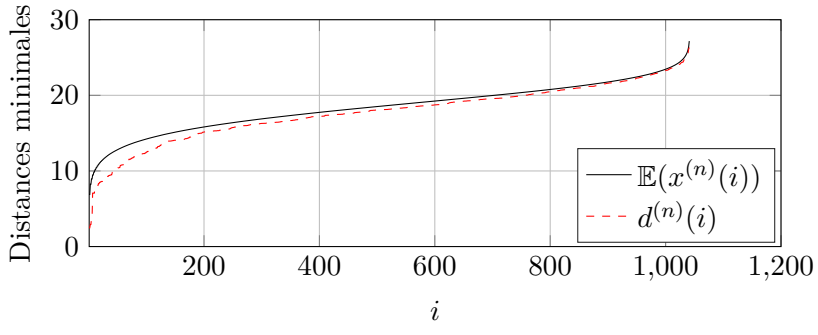
3.1.2 Estimateur de la longueur du code

Pour estimer la longueur du code, nous voulons mettre en évidence la différence de comportement entre $\mathbf{d}^{(n)}$ et $\mathbf{x}^{(n)}$ comme indiqué par la Figure 3.2. Avec notre méthode, chaque valeur du vecteur ordonné $d^{(n)}(i)$ est comparée à l'espérance d'ordre correspondant dans le cas i.i.d. : $\mathbb{E}(x^{(n)}(i))$. Pour ce faire, nous tabulons les valeurs de $\mathbb{E}(x^{(n)}(i))$: nous calculons les valeurs $x^{(n)}(i)$ pour 1000 essais de Monte-Carlo et estimons $\mathbb{E}(x^{(n)}(i))$ par une moyenne empirique. Lorsque n est différent de la longueur du code, $\mathbf{d}^{(n)}$ est distribué comme $\mathbf{x}^{(n)}$ dans le cas i.i.d.. Par contre, lorsque $n = n_c$, deux paramètres ont un impact discriminant sur la distribution de $\mathbf{d}^{(n)}$: la dimension du code k_c et sa distance de Hamming minimale d_{min}.

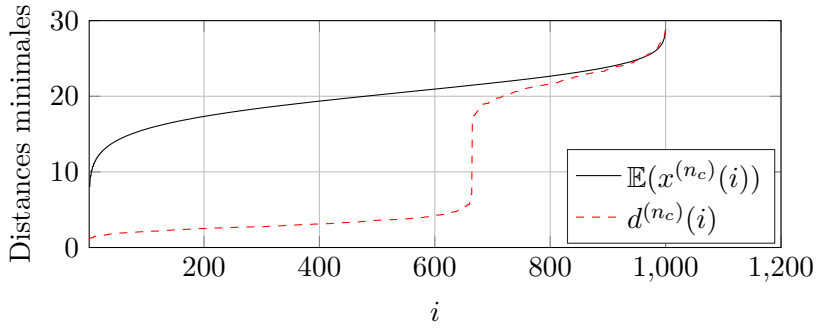
Le premier paramètre d'impact est k_c. Lorsque $n = n_c$, la probabilité que deux blocs proviennent de la même séquence de bits n_c est plus élevée dans le cas codé. En effet, le nombre total de mots de code n'est que de 2^{k_c} dans ce cas, alors qu'il y a 2^{n_c} séquences possibles de n_c bits dans le cas i.i.d. ($k_c < n_c$). Par conséquent, il est plus probable que des distances faibles apparaissent. Le deuxième paramètre d'impact est d_{min}. La distance minimale de Hamming du code, d_{min}, ne permet pas à deux mots de code différents d'avoir une distance de Hamming comprise entre 1 et $d_{min} - 1$ (pour $d_{min} > 1$). Par conséquent, il existe une plage de distances euclidiennes dont la probabilité d'occurrence est plus faible. L'étendue de cette plage est évidement liée à d_{min} mais également à la variance du bruit. En effet, plus le niveau de bruit sera élevé, moins l'effet de la valeur de d_{min} sera prononcé et détectable. L'Exemple 3.1.3. montrent l'ampleur de ces phénomènes pour le même code que l'Exemple 3.1.2..

Exemple 3.1.3.

> Reconsidérons, le code $\mathcal{C}(25, 10)$ de l'Exemple 3.1.2.. La Figure 3.3 montre les valeurs de distances ordonnées composant $\mathbf{d}^{(n)}$ et $\mathbb{E}(\mathbf{x}^{(n)})$ lorsque $n = 24$ et $n = n_c = 25$ en fonction de leur rang. Ces valeurs correspondent aux premières colonnes des matrices présentées dans la Figure 3.2.



(a)



(b)

FIGURE 3.3 – Comparaison des distributions des distances euclidiennes dans le cas i.i.d. (trait noir plein) et en présence d'un code correcteur (tirets rouges) pour $n = 24$ (3.3(a)) et $n = n_c = 25$ (3.3(b)) à 5 dB

Lorsque $n = 24$ les distributions de $\mathbb{E}(x^{(n)}(i))$ et $d^{(n)}(i)$ sont assez similaires dans la Figure 3.3(a). D'après la Figure 3.3(b), pour $n = n_c = 25$, la distribution de $\mathbf{d}^{(n_c)}$ (ligne rouge pointillée) se distingue de celle de $\mathbb{E}(\mathbf{x}^{(n_c)})$ pour des valeurs de $i < 660$. En effet, pour toutes les valeurs de $i < 660$, les valeurs de $d^{(n)}(i)$ sont particulièrement faibles (proches de 0). Dans cet exemple, les 660 premières distances sont plus susceptibles d'être des distances entre des blocs provenant de mots de code identiques, ce qui explique leurs faibles valeurs. En l'absence de bruit, ces 660 premières distances seraient égales à 0. Autour de 660-ième distance, on remarque une augmentation conséquente des valeurs que prend $d^{(n)}(i)$. Elle peut être observée à la première colonne de la Figure 3.3(b) pour le cas codé : il y a un changement de couleur (passage de bleu à jaune). Cette augmentation est dûe à d_{min}, la distance minimale du code. La distance minimale engendre des plages de distances qu'il est très peu probable que $d^{(n)}(i)$ prennent pour valeurs. Enfin, lorsque $i > 660$, la distribution de distance minimale a le même comportement que dans le cas i.i.d..

Pour quantifier les dissimilarités comportementales entre $\mathbb{E}(x^{(n)}(i))$ et $d^{(n)}(i)$, nous définissons un vecteur $\mathbf{e}_d^{(n)}$ contenant les erreurs absolues composante à composante $e_d^{(n)}(i)$:

$$\mathbf{e}_d^{(n)} = \{e_d^{(n)}(i)\}_{i\in[\![0,L_n-1]\!]} = \{|\mathbb{E}(x^{(n)}(i)) - d^{(n)}(i)|\}_{i\in[\![0,L_n-1]\!]}$$

Pour chaque taille de bloc n, nous calculons l'erreur absolue moyenne :

$$\varepsilon(\mathbf{e}_d^{(n)}) = \frac{1}{L_n} \sum_{i=0}^{L_n-1} (e_d^{(n)}(i)) = \frac{1}{L_n} \sum_{i=0}^{L_n-1} (|\mathbb{E}(x^{(n)}(i)) - d^{(n)}(i)|) \tag{3.3}$$

Comme dit précédemment, pour $n \neq n_c$, les distances minimales entre deux blocs $d^{(n)}(i)$ sont réparties comme dans le cas i.i.d. Par conséquent, l'erreur absolue moyenne $\varepsilon(\mathbf{e}_d^{(n)})$ est proche de 0. Lorsque l'erreur est maximale, $n = n_c$, ce qui donne l'estimateur suivant :

$$\hat{n}_c = \underset{n}{\operatorname{argmax}}(\varepsilon(\mathbf{e}_d^{(n)})) \tag{3.4}$$

Exemple 3.1.4.

Avec le même code LDPC que pour l'exemple précédent ($n_c = 25$, $k_c = 10$), la Figure 3.4 représente $\varepsilon(\mathbf{e}_d^{(n)})$ pour différentes valeurs de n comprises entre 10 et 30, lorsque $N = 25000$ échantillons sont reçus. Pour $n \neq n_c$, les valeurs de $\varepsilon(\mathbf{e}_d^{(n)})$ sont proches de 0. En revanche pour $n = n_c = 25$, $\varepsilon(\mathbf{e}_d^{(n)})$ est maximale. D'après l'estimateur 3.4, la valeur de $\hat{n}_c$ estimée est 25, ce qui correspond bien à la taille des blocs.

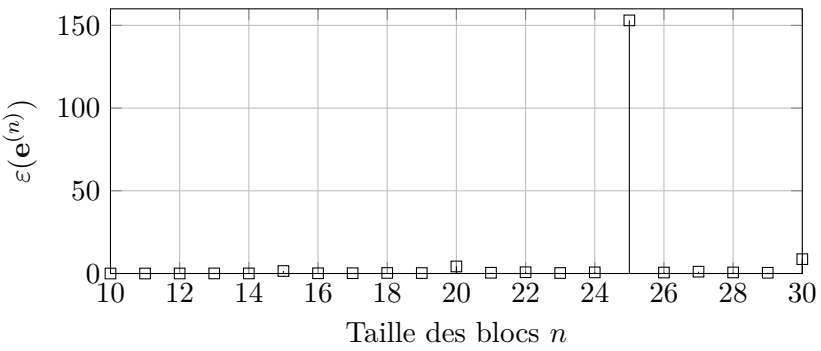


FIGURE 3.4 – Erreur moyenne entre les distances euclidiennes minimales dans le cas i.i.d. et le cas codé $\varepsilon(\mathbf{e}_d^{(n)})$ à 5 dB avec un code LDPC $\mathcal{C}(25, 10)$ pour $N = 25000$ échantillons bruités reçus

L'algorithme 2 donne une synthèse des étapes permettant l'identification de la longueur du code.

Algorithme 2 SOD : ALGORITHME D'IDENTIFICATION À PARTIR DES MATRICES DE DISTANCES

Entrées : $\mathbf{y}$, n_{min}, n_{max}

1· **Pour** $n = n_{min}$ à n_{max} **Faire**
2· Créer L_n blocs $\mathfrak{B}_i^{(n)}$ de longueur n à partir de $\mathbf{y}$ comme défini par (3.1)
3· Calculer $\mathbf{B}^{(n)}$ comme défini par (3.2)
4· Ordonner la matrice les éléments de $\mathbf{B}^{(n)}$ pour obtenir $\mathbf{D}^{(n)}$
5· $\mathbf{d}^{(n)} \leftarrow \mathbf{D}_{\cdot,1}^{(n)}$
6· Répéter les lignes 3 et 4 précédentes pour L_n blocs i.i.d. afin d'obtenir la matrice ordonnée $\mathbb{E}(\mathbf{X}^{(n)})$
7· $\mathbb{E}(\mathbf{x}^{(n)}) \leftarrow \mathbb{E}(\mathbf{X}_{\cdot,1}^{(n)})$
8· Calculer $\varepsilon(\mathbf{e}_d^{(n)})$ comme défini par (3.3)
9· **Fin Pour**
10· $\hat{n}_c = \text{argmax}_n(\varepsilon(\mathbf{e}_d^{(n)}))$

Sorties : $\hat{n}_c$

3.2 Résultats de simulation

Dans cette section, nous évaluons les performances de la méthode d'identification de n_c proposée dans ce chapitre. Pour illustrer nos résultats, nous utilisons les trois codes en bloc linéaires suivants :

- $\mathcal{C}_1$: code LDPC $\mathcal{C}(25, 10)$ de rendement $\frac{2}{5}$ et de distance minimale $d_{min} = 6$
- $\mathcal{C}_4$: code LDPC $\mathcal{C}(32, 16)$ de rendement $\frac{1}{2}$ et de distance minimale $d_{min} = 4$
- $\mathcal{C}_5$: code de Hamming $\mathcal{C}(15, 11)$ de rendement $\frac{11}{15}$ et de distance minimale $d_{min} = 3$

Les matrices génératrice et de parité de ces codes sont données dans l'Annexe B. Une représentation de la distribution des distances de Hamming est également proposée dans cette annexe. Comme dans le chapitre précédent, nous mettrons en évidence les performances de notre algorithme en le confrontant à la méthode basée sur la déficience de rang (noté DR) [83]. La méthode décrite dans ce chapitre sera notée SOD pour Statistique d'Ordre sur les Distances.

D'après l'équation 3.3, $\varepsilon(\mathbf{e}_d^{(n)})$ nécessite la connaissance de la distribution des distances minimales ordonnées dans le cas d'une trame i.i.d. En effet, la méthode demande la connaissance de $\mathbb{E}(x^{(n)}(i))$ qui correspond à l'espérance de la i-ième statistique d'ordre sur les distances minimales obtenue pour une trame i.i.d. bruitée. Ces valeurs peuvent être tabulées car elles sont indépendantes du code. Elles dépendent uniquement du niveau de bruit et du nombre de symboles reçus. Ne disposant pas d'expression théorique de $\mathbb{E}(x^{(n)}(i))$, on l'estime par itérations de Monte Carlo en appliquant la procédure d'ordonnancement des distances au cas d'une trame i.i.d. bruitée.

Dans la suite, nous noterons, $\mathbf{P}_{n_c}$, la probabilité de correctement identifier la longueur du code. Pour chacune des courbes tracées dans cette section, 1000 tirages de Monte Carlo ont été effectués. Au cours de chaque tirage, les séquences de mots d'information et de bruit sont générées aléatoirement.

3.2.1 Impact du niveau de bruit

Pour différents niveaux de bruit, la Figure 3.5 montre la probabilité d'identification $\mathbf{P}_{n_c}$ pour les codes $\mathcal{C}_1$, $\mathcal{C}_4$ et $\mathcal{C}_5$ lorsque le nombre de mots de code reçus est $L_{n_c} = 2000$ et que les valeurs de n testées s'étendent de $n_{min} = 10$ à $n_{max} = 40$. Puisque notre méthode repose sur la mesure de distances minimales, les performances sont en lien direct avec d_{min}. En effet, plus la distance minimale est grande, plus il est aisé d'identifier le code. Comme indiqué dans la sous-section 3.1.2, l'amplitude du saut de distances euclidiennes (représentée dans la Figure 3.3(b)) dépend de d_{min} et du niveau de bruit. Si d_{min} est faible, alors le saut de distance est faible également. La présence de bruit accentue ce comportement et quand le niveau de bruit est trop élevé, les distances minimales dans le cas codé ont tendance à être distribuées comme dans le cas i.i.d. Par conséquent, lorsque d_{min} est petit, les codes sont plus difficiles à identifier avec la méthode SOD.

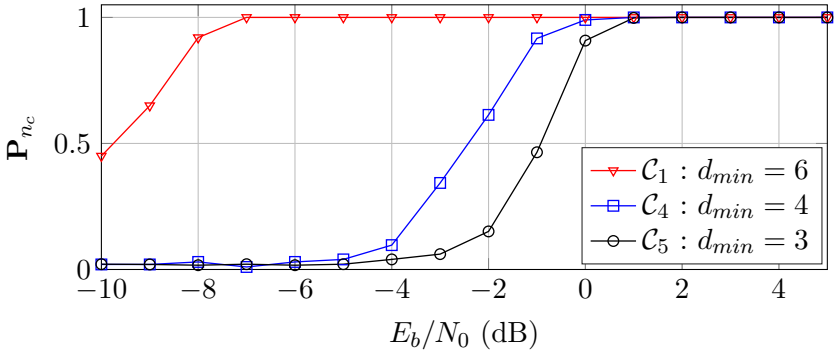


FIGURE 3.5 – Comparaison de la probabilité de correctement identifier la longueur du code pour $\mathcal{C}_1$, $\mathcal{C}_4$ et $\mathcal{C}_5$ en fonction du niveau de bruit à partir de $L_{n_c} = 2000$ mots de code reçus

De plus, la probabilité d'avoir deux blocs provenant du même mot de code dépend de la dimension du code k_c. Les codes $\mathcal{C}_1$ et $\mathcal{C}_5$ ont une dimension proche, $k_c = 10$ et $k_c = 11$ respectivement. Pour ces deux codes, L_{nc} est du même ordre de grandeur que 2^{k_c} comparativement au cas du code $\mathcal{C}_4$. Cependant, la distance minimale de Hamming est $d_{min} = 6$ et $d_{min} = 3$ pour $\mathcal{C}_1$ et $\mathcal{C}_5$ respectivement. Par conséquent, la plage de distances minimales générée par d_{min} permet de détecter $\mathcal{C}_1$ plus efficacement que C_5 à des niveaux de bruit plus élevés. Pour $\mathcal{C}_4$, L_{n_c} est beaucoup plus petit que 2^{k_c} et notre algorithme fonctionne moins efficacement qu'avec $\mathcal{C}_1$. Cependant, $\mathcal{C}_4$ donne de meilleurs résultats que $\mathcal{C}_5$ malgré le fait que $L_{n_c} \ll 2^{16}$ et $L_{n_c} \approx 2^{11}$. Pour $\mathcal{C}_4$, la méthode SOD est plus résiliente à la présence de bruit car sa d_{min} permet d'obtenir un plus grand nombre de distances minimales faibles.

3.2.2 Comparaison avec la méthode de déficience de rang

Nous comparons ici les performances de notre algorithme d'identification de n_c, noté SOD, avec la méthode par déficience de rang, noté DR, pour le code $\mathcal{C}_1$. Nous analysons ces résultats pour différents niveaux de bruit et pour $L_{n_c} = 500$ et $L_{n_c} = 1000$ mots de code reçus. La

méthode DR est calculée pour cinq itérations de virtualisation. A chaque itération, l'algorithme opère une permutation aléatoire sur les lignes de la matrice d'interception comme expliqué dans la section 1.3.1. La plage de taille de blocs n testée s'étend de $n_{min} = 10$ à $n_{max} = 30$.

A partir de la Figure 3.6, nous observons que notre méthode basée sur les statistiques des distances minimales surpasse la méthode d'élimination de gauss quel que soit la durée d'observation. Elle exploite efficacement deux caractéristiques de la distribution des mots de code : la distance minimale de Hamming et la dimension du code. Pour $L_{n_c} = 1000$, la probabilité de bonne détection est proche de 1 dès -6 dB pour notre algorithme d'identification contre -2 dB pour l'algorithme DR, soit un gain de 4 dB. Pour $L_{n_c} = 500$, cette même probabilité est atteinte vers -5 dB pour notre algorithme contre -2 dB pour l'algorithme DR, soit un gain de 3 dB. Nous remarquons également que le nombre de mots reçus a une influence sur notre algorithme d'identification.

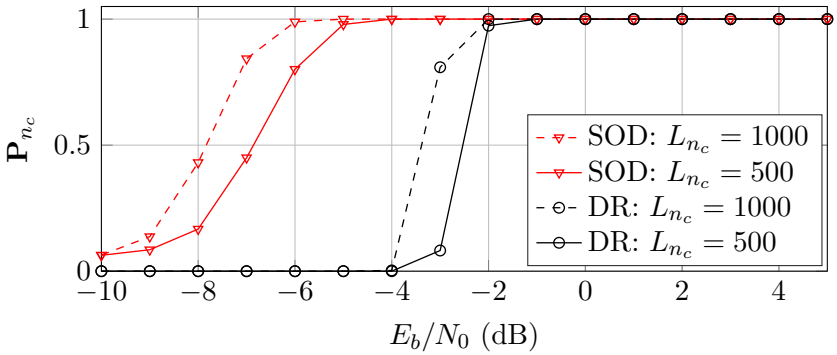


FIGURE 3.6 – Comparaison des méthodes SOD et DR (triangles rouges et cercles noirs respectivement) pour le code $\mathcal{C}_1$ à partir de $L_{n_c} = 500$ et $L_{n_c} = 1000$ mots de code reçus

3.2.3 Impact de la longueur des trames reçues

Comme le montre la Figure 3.6, la durée d'observation a une influence sur l'efficacité des méthodes comparées. La Figure 3.7 représente la probabilité d'une identification correcte par rapport à la quantité de mots de code reçus L_{n_c} pour le code $\mathcal{C}_1$ et pour différents niveaux de bruit. La tendance générale est la suivante : plus le nombre de mots reçus est grand, meilleures sont les performances de notre algorithme. La Figure 3.7 permet de comparer les deux méthodes SOD et DR pour un niveau de bruit de -3 dB. La quantité minimale de mots de code nécessaire pour estimer la longueur du code est considérablement plus faible avec la méthode des statistiques. Environ 250 mots de code sont suffisants avec la méthode SOD alors que l'approche DR nécessite plus de 1250 mots de code. Nous pouvons également remarquer que même à un niveau de bruit plus élevé (-5 et -7 dB), notre méthode surpasse les performances de la méthode DR à -3 dB et ceci pour toutes les valeurs de L_n.

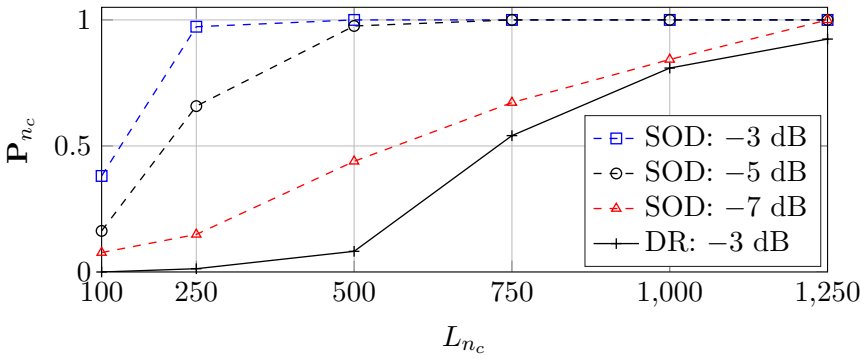


FIGURE 3.7 – Influence du nombre de mots de code reçus L_{n_c} sur la probabilité de correctement identifier la longueur du code $\mathcal{C}_1$

3.2.4 Comparaison des deux méthodes basées sur les informations souples

Nous avons montré que la méthode par déficience du nombre de classes (DNC) proposée dans le chapitre précédent et que la méthode par statistique d'ordre sur les distances (SOD) surpassent la méthode par déficience de rang. Dans cette section, nous nous intéressons à la comparaison des deux méthodes basées sur les informations souples.

Les performances des deux méthodes d'identification de n_c pour le code $\mathcal{C}_1$ sont représentées sur la Figure 3.8 pour différentes valeurs de L_{n_c}. Nous constatons que pour $L_{n_c} = 2000$ mots de code reçus, la méthode basée sur les statistiques d'ordres surpasse la méthode par déficience du nombre de classe. Pour $L_{n_c} = 500$, nous pouvons faire le même constat mais le gain est moins important pour des niveaux de bruit plus élevés. Pour $L_{n_c} = 1000$, nous remarquons que pour des niveaux de bruit élevés, compris entre -10 et -8 dB, la méthode SOD offre de meilleures performances que la méthode DNC. En revanche, la méthode DNC permet d'obtenir une probabilité de détection de 1 dès -7 dB contre -6 dB pour la méthode SOD. Pour la méthode DNC, l'ajout de mots de code augmente la probabilité que deux mots appartiennent à une même classe mais lorsque le niveau de bruit est trop élevé, les agglomérats formés par les mots bruités sont plus difficilement discriminés quelque soit le seuil β, ce qui explique que pour $L_{n_c} = 1000$ et des niveaux de bruit élevés la méthode DNC ne permette pas d'identifier correctement n_c, contrairement à la méthode SOD.

Sur cette même figure, nous remarquons que pour la méthode DNC, le gain entre la courbe pour $L_{n_c} = 1000$ et $L_{n_c} = 2000$ est quasiment nul. En revanche pour la méthode SOD, l'augmentation du nombre de mots reçus permet l'augmentation de la probabilité d'avoir deux mots de code initialement identiques à l'émission. Par conséquent, la probabilité de mesurer des distances faibles augmente. Comme le critère d'identification s'appuie sur les distances minimales, il en résulte que la probabilité de correctement identifier n_c tend à augmenter aussi.

La Figure 3.9 permet d'observer l'effet de l'augmentation de L_{n_c} sur la distribution des distances minimale. Nous observons que pour $L_{n_c} = 2000$, la différence de comportement avec le cas i.i.d. correspondant est plus grande (*c.f.* aire entre les deux courbes en bleu) que pour

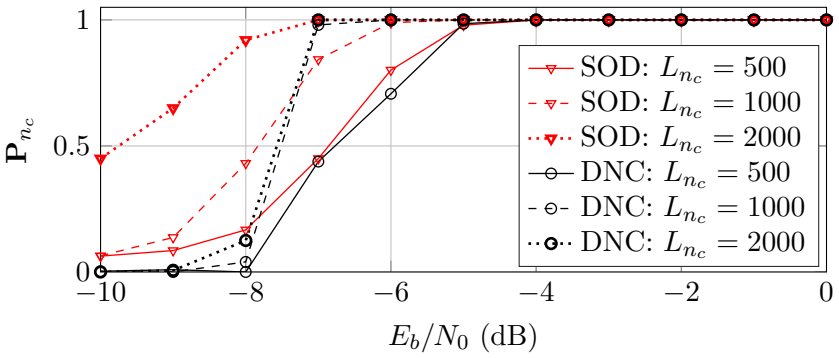


FIGURE 3.8 – Comparaison de la probabilité de correctement identifier la longueur du code pour $\mathcal{C}_1$ avec la méthode DNC et la méthode SOD en fonction du niveau de bruit à partir de $L_{n_c} = 1000$ mots de code reçus

$L_{n_c} = 1000$ (*c.f.* aire entre les deux courbes en rouge). Le nombre de distances minimales entre deux mots de code initialement identiques est d'environ 1700 pour $L_{n_c} = 2000$ alors qu'il n'est que d'environ 600 pour $L_{n_c} = 1000$.

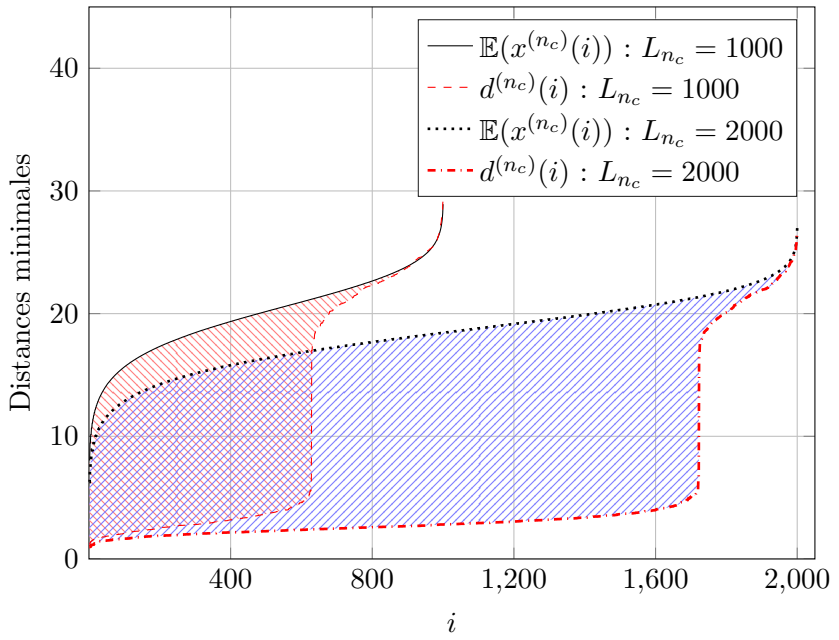


FIGURE 3.9 – Comparaison de la distribution des distances minimales pour $\mathcal{C}_1$ pour $L_{n_c} = 1000$ et $L_{n_c} = 2000$ mots de code reçus à 5 dB

Pour finir, nous évaluons l'impact d'une mauvaise synchronisation sur notre méthode. La Figure 3.10 donne un aperçu de l'impact de différentes valeurs de désynchronisation sur la probabilité de bonne identification de la longueur du code à 0 et à −5 dB pour les méthodes SOD et DNC. Nous remarquons que la synchronisation de la trame a les mêmes effets, quelque soit la méthode. En effet, à l'instar de la méthode DNC, la structure quasi-cyclique du code $\mathcal{C}_1$ permet d'obtenir de très bonnes performances d'identification à 0 dB quelque soit la valeur de

Δt avec la méthode SOD. A -5 dB, les résultats sont sensiblement les mêmes pour les deux méthodes.

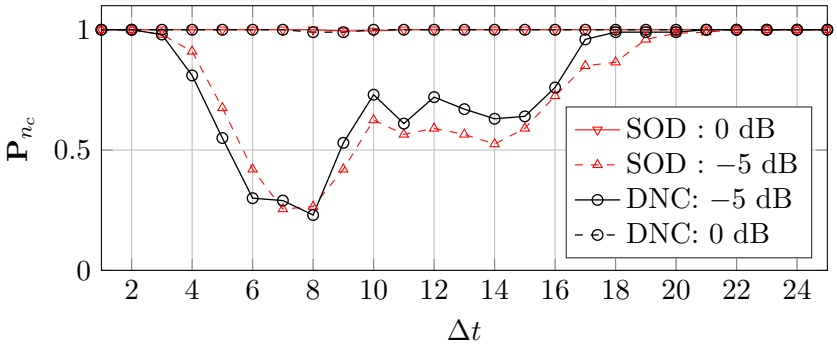


FIGURE 3.10 – Comparaison de la probabilité de correctement identifier la longueur du code pour $\mathcal{C}_1$ avec la méthode DNC et la méthode SOD en fonction de la valeur de désynchronisation Δt à partir de $L_{n_c} = 1000$ mots de code reçus

3.3 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode s'appuyant sur les statistiques d'ordre pour identifier la longueur d'un code en bloc linéaire binaire. Il s'agit d'un algorithme basé sur l'utilisation des informations souples. A partir d'une séquence reçue, il calcule les distances euclidiennes minimales entre des blocs de symboles bruités. La répartition des distances minimales ordonnées dépend du codeur. Lorsque le nombre d'échantillons reçus est suffisant, cette distribution diffère radicalement de la distribution obtenue à partir d'une séquence i.i.d. Nous utilisons cette différence pour estimer la longueur du code. Cette méthode est particulièrement efficace pour les codes avec une grande distance de Hamming minimale.

Nous avons montré que cette méthode est plus performante que la méthode par déficience de rang basée sur des colonnes presque dépendantes dans [83] avec une quantité suffisamment importante d'échantillons reçus. Il s'avère également que la méthode SOD est plus robuste à la présence du bruit que la méthode par déficience du nombre de classes présentée dans le chapitre précédent. En effet, l'augmentation de la quantité de mots reçus permet d'améliorer les performances de la méthode car elle s'appuie sur l'observation des distances minimale qui sont alors en plus grand nombre. Par conséquent, cette méthode est plus adaptée à l'identification de codes plus longs que la méthode DNC. En effet, si leur distance minimale de Hamming est assez grande et que la quantité de mots de code reçus est suffisante, l'identification de la longueur du code est peut être possible. Il serait donc intéressant d'établir des bornes inférieures sur la quantité nécessaire de mots reçus en fonction du niveau de bruit et de la distance minimale du code à identifier. De même, la méthode SOD demande de tester différentes tailles de blocs : l'établissement d'un critère d'arrêt permettrait de limiter le nombre de ces tests. Enfin, l'utilisation de cette méthode sur d'autres familles de codes correcteurs d'erreurs est une voie à explorer. Il serait par exemple intéressant d'observer les effets de la longueur de contrainte des

codes convolutifs. Ensuite, l'étude de l'identification des paramètres de codeurs équivalents à des codes concaténés et des codes poinçonnés ou d'un entrelaceur pourrait mener à l'identification de turbo-codes.

Chapitre 4

Reconstruction par observation des distances euclidiennes

Dans les chapitres précédents nous avons présenté deux nouvelles méthodes basées sur le calcul de distances euclidiennes permettant d'identifier la longueur d'un code. La connaissance de cette longueur, bien que nécessaire, ne suffit pas pour décoder le signal reçu et donc retrouver le message émis. En effet, il est nécessaire d'identifier des relations de parité du code afin de construire une base du code dual pour mettre en oeuvre un décodeur et retrouver le message d'origine. Dans ce chapitre, nous nous concentrons sur la reconstruction du code qui consiste à identifier un ensemble de relations de parité formant une base du code dual. Le problème est formalisé comme suit : à partir de L_{n_c} mots de code bruités reçus d'un code $\mathcal{C}$, on souhaite identifier des mots $\mathbf{h}$ appartenant au code dual $\mathcal{C}^T$. Ces mots $\mathbf{h}$ correspondent à des relations de parité du code $\mathcal{C}$.

Certaines méthodes s'appuyant sur les informations fermes donnent des solutions à ce problème comme celles présentées dans la section 1.3. Ces méthodes s'appuient sur des manipulations algébriques [16, 22, 97] appliquées à une matrice d'interception $\mathbf{M}$. Cette matrice $\mathbf{M}$ est formée par les mots reçus de taille n_c. Pour limiter la complexité algorithmique, certaines de ces méthodes proposent de réduire ce problème à un sous-problème [16, 97]. Cela consiste à appliquer la recherche de bits liés par une relation de parité à une sous-matrice de $\mathbf{M}$, donc sur un ensemble de mots plus courts. Dans ce chapitre, nous nous intéressons à la résolution de ce sous-problème.

Comme soulevé lors du développement des méthodes DNC et SOD, la distribution des distances euclidiennes dans un espace généré par un code a une distribution particulière. Pour identifier des relations de parité du code, nous allons donc observer les distances euclidiennes dans l'espace généré par un code de parité. La Figure 4.1 montre un exemple de la distribution des distances pour les éléments du code de parité $\mathcal{C}(4, 3)$, ainsi que pour des mots i.i.d. de longueur 4. L'ensemble des mots d'un code de parité sont de poids de Hamming paire. Cette particularité fait que la probabilité d'obtenir des distances euclidiennes entre 0.5 et 2.5 est très faible en présence de bruit. Nous comparons ces distances euclidiennes avec celles obtenues à partir de mots i.i.d. de taille 4 et nous observons qu'elles n'ont pas le même comportement.

Partant de ce principe, à partir de mots de code bruités, la distribution des mots formés par les symboles impliqués dans une relation de parité devrait avoir un comportement particulier. En effet, cette distribution devrait être similaire à la distribution des mots du code de parité $\mathcal{C}(4,3)$ et donc différer de celle du cas i.i.d.

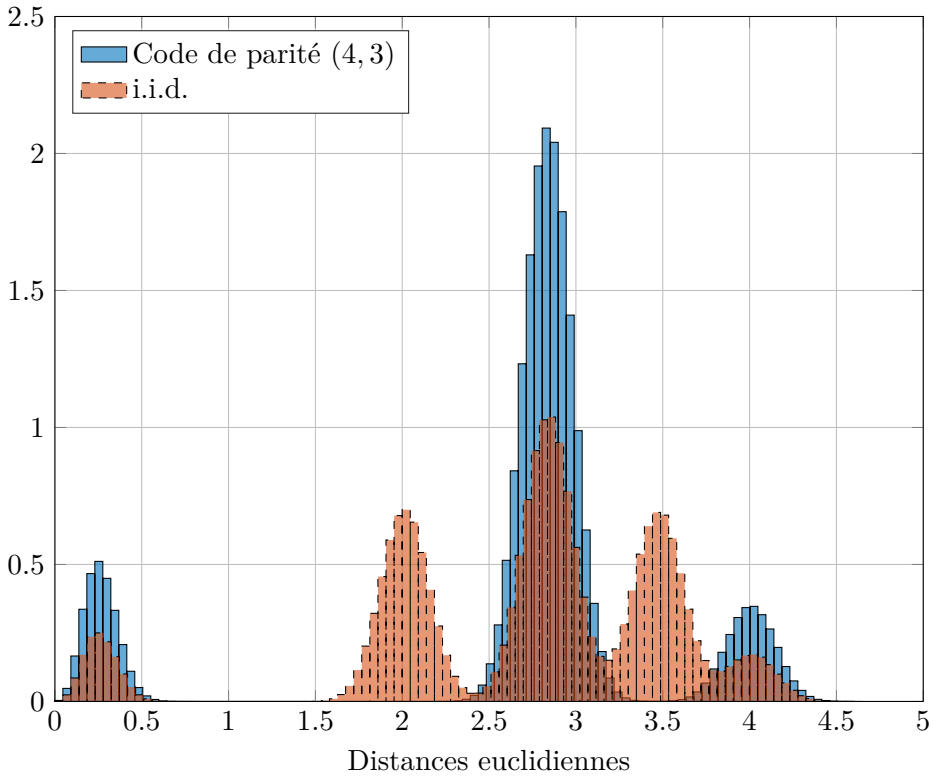


FIGURE 4.1 – Distribution des distances euclidiennes pour des mots du code de parité $\mathcal{C}(4,3)$

En partant de cette idée, dans la suite de ce chapitre, nous proposons une méthode pour identifier des relations de parité d'un code en bloc linéaire quelconque à partir de mots de code bruités reçus. Cette méthode repose sur l'observation de la distribution des distances entre les mots formés à partir de symboles impliqués dans une relation de parité. Tout d'abord, nous exposons les principes fondant la méthode. Ensuite, nous exhibons un critère permettant de discriminer les différentes relations. Enfin, quelques tests et résultats donnent un aperçu des performances de cette méthode que nous comparons à une méthode de la littérature fréquemment utilisée lors de la résolution du sous-problème présenté précédemment, à savoir le paradoxe des anniversaires [22]. Pour finir, une extension de cette méthode au cas des codes convolutifs est proposée.

4.1 Reconstruction basée sur l'observation de la distribution des distances euclidiennes

Comme pour les méthodes DNC et SOD, la méthode présentée ici traite des symboles issus d'une MP-2 bruités par un canal BBAG. Nous ferons l'hypothèse que la longueur du code est connue ou parfaitement identifiée. A partir de la séquence reçue $\mathbf{y}$ de longueur N, la matrice $\mathbf{M}_{soft}$, de taille $(L_{n_c} \times n_c)$ avec $L_{n_c} = \lfloor \frac{N}{n_c} \rfloor$, est formée. Chacune des L_{n_c} lignes de cette matrice correspond à un mot de code bruité issu d'un code $\mathcal{C}(n_c, k_c)$.

4.1.1 Principes de la méthode de reconstruction

Cette méthode consiste à tester des relations $\mathbf{h}$ de poids w comme suit : dans chaque mot de code reçu, on ne conserve que les bits impliqués dans la relation testée, pour former des mots plus courts de longueur w. A partir des nouveaux mots formés, nous calculons toutes les distances euclidiennes deux à deux pour observer leur distribution. Deux cas de figure sont observés :

- Lorsque la relation testée appartient effectivement au code dual de $\mathcal{C}(n_c, k_c)$, la distribution des distances deux à deux des mots formés correspond à celle des distances pour un code de parité de longueur w.
- A l'inverse, lorsque la relation testée n'appartient pas au code dual de $\mathcal{C}(n_c, k_c)$, la distribution des distances correspond à celle des distances pour des mots i.i.d. de longueur w.

La détection des relations de parité $\mathbf{h}$ appartenant effectivement au code dual se fait donc par la comparaison de la distribution observée des distances à celles d'une trame i.i.d. et d'une trame issue d'un code de parité simple. En effet, lorsque $\mathbf{h}$ est effectivement une relation de parité, le poids des mots formés par les bits impliqués dans cette relation est nécessairement paire, comme dans le cas d'un code de parité.

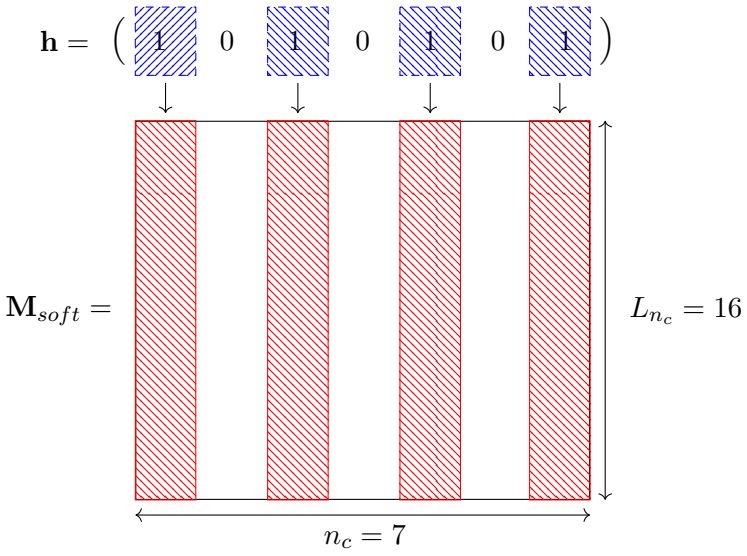
Pour une relation testée $\mathbf{h}$ de poids w, nous gardons dans la matrice $\mathbf{M}_{soft}$ les w colonnes correspondant aux bits impliqués par la relation testée. Ensuite, nous calculons les distances euclidiennes entre les L_{n_c} mots de taille w et nous les ordonnons. Nous noterons $\mathbf{x}^{(\mathbf{h})} = (x^{(\mathbf{h})}(0), x^{(\mathbf{h})}(1), \cdots, x^{(\mathbf{h})}(d-1))$ les d distances mesurées à partir de la matrice $\mathbf{M}_{soft}$ pour la relation testée $\mathbf{h}$. Ces distances sont ordonnées de telle sorte que : $x^{(\mathbf{h})}(0) \leqslant x^{(\mathbf{h})}(1) \leqslant \cdots \leqslant x^{(\mathbf{h})}(d-1)$. Afin de comparer ces distances à celles obtenues avec un code de parité, nous réalisons le codage de L_{n_c} mots d'information avec le code de parité $\mathcal{C}(w, w-1)$. Puis nous calculons l'ensemble des distances euclidiennes entre les mots du code de parité deux à deux et nous ordonnons ces distances. Nous noterons $\mathbf{X}^{(par)} = (X^{(par)}(0), X^{(par)}(1), \cdots, X^{(par)}(d-1))$ un échantillon ordonné de variables aléatoires suivant la distribution des distances euclidiennes des mots bruités du code de parité de longueur w. Enfin, nous notons $\mathbf{X}^{(iid)} = (X^{(iid)}(0), X^{(iid)}(1), \cdots, X^{(iid)}(d-1))$ un échantillon de variables aléatoires suivant la distribution des distances euclidiennes deux à deux de mots i.i.d. bruités de longueur w. Comme pour les distances observées, ces

échantillons sont ordonnés de sorte que : $X^{(par)}(0) \leqslant X^{(par)}(1) \leqslant \cdots \leqslant X^{(par)}(d-1)$ et $X^{(iid)}(0) \leqslant X^{(iid)}(1) \leqslant \cdots \leqslant X^{(iid)}(d-1)$. L'objectif est donc de quantifier l'adéquation des distances mesurées $\mathbf{x}^{(\mathbf{h})}$ aux distributions de $\mathbf{X}^{(par)}$ et $\mathbf{X}^{(iid)}$. L'Exemple 4.1.1. donne un aperçu de ce que peuvent être ces différentes distributions.

Exemple 4.1.1.

A partir de 16 mots du code de Hamming $\mathcal{C}_7(7,4)$, nous créons la matrice d'interception $\mathbf{M}_{soft}$. Cette matrice est donc de taille (16×7). Nous cherchons des relations de poids $w = 4$. Pour chaque relation testée $\mathbf{h}$, seules les colonnes indiquées par la position des 1 dans $\mathbf{h}$ sont conservées.

Prenons l'exemple d'une relation de parité du code $\mathbf{h} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$. Dans la matrice $\mathbf{M}_{soft}$ seules les colonnes hachurées en rouge sont conservées pour former les 16 mots de 4 symboles dans le schéma suivant :



Ensuite, les distances euclidiennes entre les mots de 4 symboles sont calculées et ordonnées, deux à deux. Nous calculons également ces distances pour le cas d'un code de parité $\mathcal{C}(4,3)$ et dans le cas de mots i.i.d. de taille 4. Pour un niveau de bruit de 5 dB, la figure suivante permet de comparer la distribution des distances calculées $\mathbf{x}^{(\mathbf{h})}$ à $\mathbb{E}(\mathbf{X}^{(iid)})$ et à $\mathbb{E}(\mathbf{X}^{(par)})$ lorsque $\mathbf{h}$ est une relation de parité du code $\mathbf{h} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$.

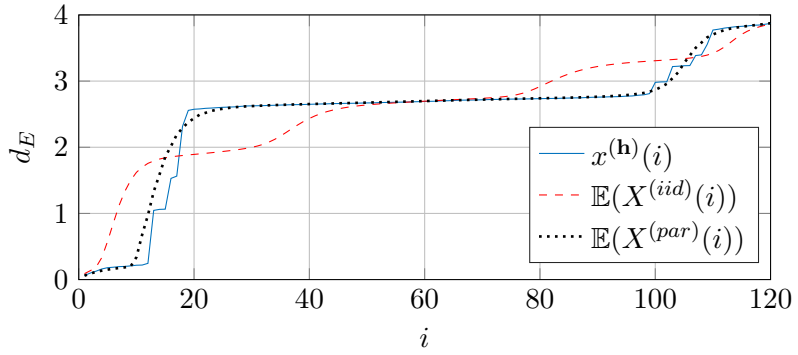


FIGURE 4.2 – Comparaison des distributions des distances euclidiennes en présence d'une vraie relation de parité pour le code $\mathcal{C}_7(7,4)$

Nous pouvons vérifier que la distribution pour $\mathbf{x}^{(\mathbf{h})}$ est très similaire à $\mathbb{E}(\mathbf{X}^{(par)})$ et diffère de $\mathbb{E}(\mathbf{X}^{(iid)})$ car $\mathbf{h}$ est une relation de parité du code de Hamming $(7,4)$.

Prenons maintenant le cas ou $\mathbf{h}$ n'est pas une relation de parité du code tel que $\mathbf{h} = \begin{pmatrix}1 & 0 & 1 & 1 & 0 & 0 & 1\end{pmatrix}$. La figure suivante illustre les différentes distribution des distances avec cette relation $\mathbf{h}$:

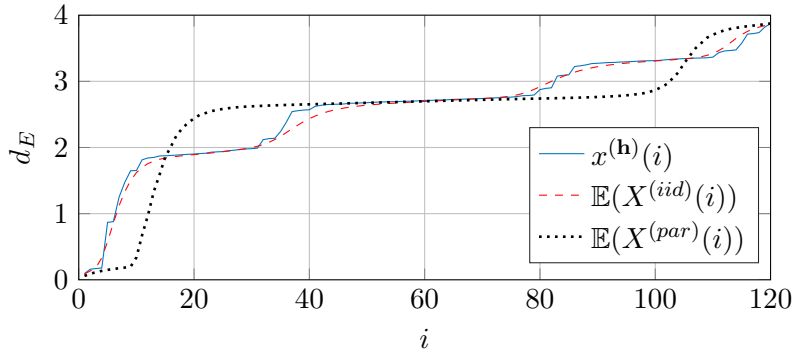


FIGURE 4.3 – Comparaison des distributions des distances euclidiennes en présence d'une fausse relation de parité pour le code $\mathcal{C}_7(7,4)$

Nous vérifions que la distribution pour $\mathbf{x}^{(\mathbf{h})}$ est très similaire à $\mathbb{E}(\mathbf{X}^{(iid)})$ et diffère de $\mathbb{E}(\mathbf{X}^{(par)})$ car $\mathbf{h}$ n'est pas une relation de parité du code de Hamming $\mathcal{C}_7(7,4)$.

Le nombre de distances calculées deux à deux, soit la valeur de d, dépend du poids des relations recherchées. En effet, les distances calculées sont celles entre des mots de taille w. Il y a donc 2^w mots différents sur w bits. En pratique, on constate que l'utilisation de 2^w mots suffit à l'obtention d'une mesure représentative de la distribution des distances dans le cas du code de parité ou de la trame i.i.d.. Ainsi en fixant à 2^w le nombre de mots traités, on obtient $d = \frac{2^w \cdot (2^w - 1)}{2}$ distances calculées. Il est à noter que la quantité de mots nécessaires pour identifier des relations de parité peut donc être très faible. En effet, ce sera toujours un multiple de 2^w et, pour limiter la complexité, w doit rester faible. Il n'y a pas de dépendance directe à la longueur

du code n_c, ce qui permet de potentiellement identifier des relations de grande taille n_c et de poids w faible à partir de 2^w mots de code bruités.

4.1.2 Utilisation des informations souples

Pour profiter au mieux de l'information contenue dans les données reçues, les mots de longueur w créés à partir de la matrice $\mathbf{M}_{soft}$ sont ordonnés par fiabilité. La fiabilité de chaque mot de poids w est déterminée par celle de son bit le moins fiable et la fiabilité d'un bit correspond à la valeur absolue de son information souple.

Définition 10. *Soit deux mots de w bits, $\mathbf{m_1}$ et $\mathbf{m_2}$:*

$\mathbf{m_1}$ *est plus fiable que* $\mathbf{m_2}$ *si et seulement si* $\min_{k\in[\![0,w-1]\!]}(\{|m_1(k)|\}) < \min_{k\in[\![0,w-1]\!]}(\{|m_2(k)|\})$.

Comme vu précédemment, les mots reçus sont traités par blocs de 2^w. En réalisant ce tri par fiabilité, il est donc possible de diminuer artificiellement le niveau de bruit sur les mots utilisés, ce qui permet de diminuer la probabilité de fausse détection. En effet, il est possible de n'utiliser qu'une portion réduite des N symboles reçus puisque 2^w mots suffisent. Finalement, les mots créés sont ordonnés et regroupés en $\lfloor \frac{L_{nc}}{2^w} \rfloor$ blocs contigus mais distincts de 2^w mots. Parmi ces $\lfloor \frac{L_{nc}}{2^w} \rfloor$ blocs, nous notons N_B, le nombre de blocs contigus retenus pour l'identification ($N_B \in [\![1, \lfloor \frac{L_{nc}}{2^w} \rfloor]\!]$). Ce nombre de bloc N_B est déterminé par l'utilisateur. Pour chacun des blocs, les $d = \frac{2^w \cdot (2^w-1)}{2}$ distances euclidiennes sont calculées et ordonnées. On note $\mathbf{x}_j^{(\mathbf{h})}$, le vecteur de distances ordonnées issu du j-ième bloc de mots ordonnés : $\mathbf{x}_j^{(\mathbf{h})} = (x_j^{(\mathbf{h})}(0), x_j^{(\mathbf{h})}(1), \cdots, x_j^{(\mathbf{h})}(d-1))$, avec $j \in [\![1, N_B]\!]$. Dans un soucis de clarté, $x_j^{(\mathbf{h})}(1)$ est, par exemple, la deuxième valeur la plus faible du vecteur de distance issu du j-ième bloc lors du test de la relation $\mathbf{h}$. A partir de tous ces blocs sont calculées les distances ordonnées moyennes $\hat{\mathbf{x}}^{(\mathbf{h})}$ pour la relation testée $\mathbf{h}$:

$$\begin{aligned}
\hat{\mathbf{x}}^{(\mathbf{h})} &= \frac{1}{N_B}\sum_{j=1}^{N_B}(x_j^{(\mathbf{h})}(0), x_j^{(\mathbf{h})}(1), \cdots, x_j^{(\mathbf{h})}(d-1))) \\
&= \left(\frac{1}{N_B}\sum_{j=1}^{N_B} x_j^{(\mathbf{h})}(0), \frac{1}{N_B}\sum_{j=1}^{N_B} x_j^{(\mathbf{h})}(1), \cdots, \frac{1}{N_B}\sum_{j=1}^{N_B} x_j^{(\mathbf{h})}(d-1)\right) \\
&= (\hat{x}^{(\mathbf{h})}(0), \hat{x}^{(\mathbf{h})}(1), \cdots, \hat{x}^{(\mathbf{h})}(d-1)) \qquad (4.1)
\end{aligned}$$

La Figure 4.4 résume la manière dont est obtenue le vecteur de distances moyennes ordonnées. La matrice à partir de laquelle sont extraits les blocs correspond à la matrice obtenue lorsqu'on se restreint à certaines colonnes de $\mathbf{M}_{soft}$ au regard de $\mathbf{h}$.

Les variables aléatoires distribuées comme les distances issues du processus de fiabilisation pour une trame i.i.d. (et pour des mots issus d'un code de parité respectivement) sont notée $\mathbf{X}^{(iid)}$ (respectivement $\mathbf{X}^{(par)}$). Ainsi, lorsque $\mathbf{h}$ est une relation de parité, $\hat{\mathbf{x}}^{(\mathbf{h})}$ suit la même loi de probabilité que $\mathbf{X}^{(par)}$. A l'inverse, lorsque $\mathbf{h}$ n'est pas une relation de parité, $\hat{\mathbf{x}}^{(\mathbf{h})}$ suit la même loi de probabilité que $\mathbf{X}^{(iid)}$. Deux calculs de distance quadratique permettent de quantifier

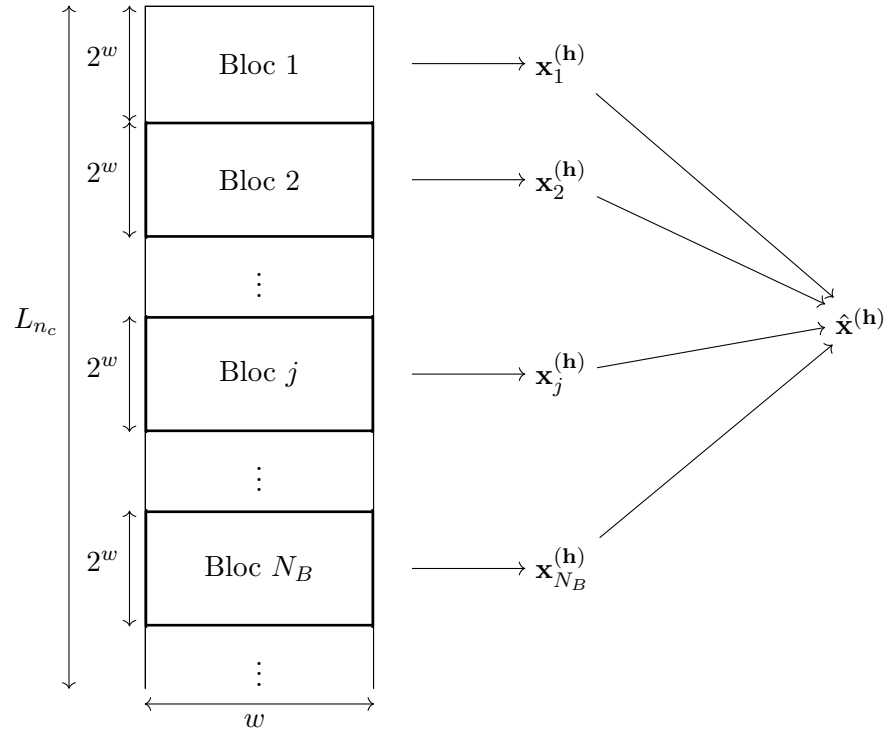


FIGURE 4.4 – Processus de fiabilisation : génération d'un vecteur de distances moyennes à partir de blocs de 2^w mots

l'adéquation des distances mesurées $\hat{\mathbf{x}}^{(\mathbf{h})}$ à l'une ou l'autre des distributions :

$$\mathcal{A}_{\mathbf{h}}^{(par)} = \sum_{i=0}^{d-1} (\hat{x}^{(\mathbf{h})}(i) - \mathbb{E}(X^{(par)}(i)))^2 \tag{4.2}$$

et

$$\mathcal{A}_{\mathbf{h}}^{(iid)} = \sum_{i=0}^{d-1} (\hat{x}^{(\mathbf{h})}(i) - \mathbb{E}(X^{(iid)}(i)))^2 \tag{4.3}$$

Exemple 4.1.2.

Pour le code de Hamming $\mathcal{C}_7(7,4)$ utilisé précédemment, nous calculons $\mathcal{A}_{\mathbf{h}}^{(par)}$ et $\mathcal{A}_{\mathbf{h}}^{(iid)}$ pour évaluer l'adéquation des distributions avec le cas du code de parité et le cas i.i.d., respectivement. Pour calculer $\mathcal{A}_{\mathbf{h}}^{(par)}$ et $\mathcal{A}_{\mathbf{h}}^{(par)}$, nous testons 18 relations différentes de poids $w = 4$. La figure suivante donne un aperçu de la distribution du niveau d'adéquation à la présence d'un code de parité pour ces 18 relations :

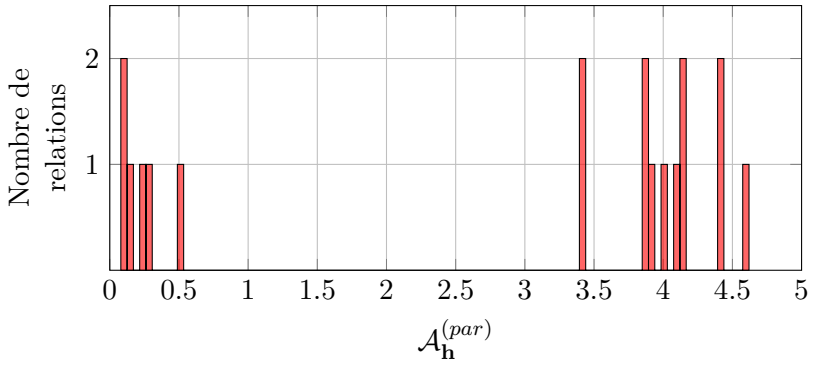


FIGURE 4.5 – Distribution de $\mathcal{A}_{\mathbf{h}}^{(par)}$ pour le code $\mathcal{C}_7(7,4)$

Cette figure représente l'histogramme de $\mathcal{A}_{\mathbf{h}}^{(par)}$. Nous remarquons que deux groupes distincts se forment. Lorsque le niveau d'adéquation entre $\hat{\mathbf{x}}^{(\mathbf{h})}$ et $\mathbf{X}^{(par)}$ est élevé, la valeur de $\mathcal{A}_{\mathbf{h}}^{(par)}$ est faible, ici elle est comprise entre 0 et 1. Il y a six relations qui correspondent à ce cas de figure. Lorsque le niveau d'adéquation entre les deux distributions est faible, la valeur de $\mathcal{A}_{\mathbf{h}}^{(par)}$ augmente, ici elle est comprise entre 3 et 5. Il y a douze relations qui correspondant à ce cas de figure.

De même, la figure suivante donne un aperçu de la distribution du niveau d'adéquation au cas i.i.d. pour ces mêmes relations :

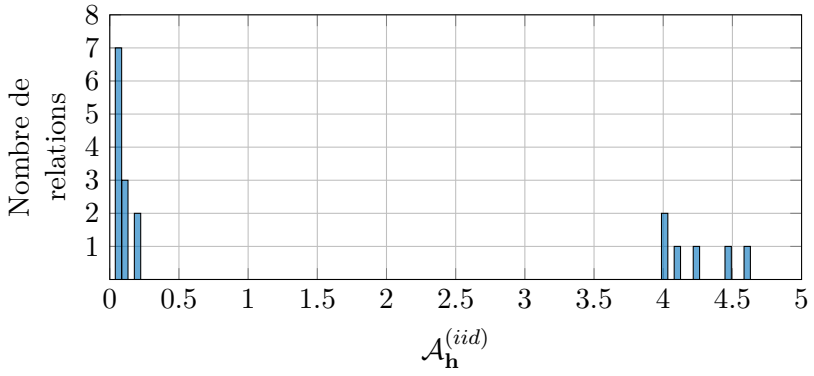


FIGURE 4.6 – Distribution de $\mathcal{A}_{\mathbf{h}}^{(iid)}$ pour le code $\mathcal{C}_7(7,4)$

Cette figure représente l'histogramme de $\mathcal{A}_{\mathbf{h}}^{(iid)}$. De même que pour $\mathcal{A}_{\mathbf{h}}^{(par)}$, deux groupes distincts se forment. Lorsque le niveau d'adéquation entre $\hat{\mathbf{x}}^{(\mathbf{h})}$ et $\mathbf{X}^{(iid)}$ est élevé, la valeur de $\mathcal{A}_{\mathbf{h}}^{(iid)}$ est faible, ici elle est comprise entre 0 et 0.5. Il y a douze relations qui correspondant à ce cas de figure. Lorsque le niveau d'adéquation entre les deux distributions est faible, la valeur de $\mathcal{A}_{\mathbf{h}}^{(iid)}$ augmente, ici elle est comprise entre 3.5 et 5. Il y a six relations qui correspondant à ce cas de figure.

Dans les deux figures précédentes, les groupes de 6 et 12 relations discriminés contiennent respectivement les mêmes relations testées. Le groupe de 6 contient les véritables relations de parité du code de Hamming $\mathcal{C}_7(7,4)$. C'est pour ce groupe que $\mathcal{A}_{\mathbf{h}}^{(par)}$ (respectivement $\mathcal{A}_{\mathbf{h}}^{(iid)}$) est faible (respectivement grand).

4.1.3 Critère de détection d'une relation de parité

Finalement, notre critère pour décider si une relation $\mathbf{h}$ est effectivement une relation de parité repose sur la valeur que prend le rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$. Plus sa valeur est élevée, plus il est probable que la relation testée appartienne effectivement au code dual. En effet, cela correspond au cas où $\mathcal{A}_{\mathbf{h}}^{(iid)}$ et $\mathcal{A}_{\mathbf{h}}^{(par)}$ ont respectivement une valeur très grande et très faible conjointement.

Pour la plupart des relations testées, le critère a une valeur proche de 0 : ces relations n'appartiennent pas au code dual. En effet, lorsque $\mathbf{h}$ n'appartient pas au code dual, $\mathcal{A}_{\mathbf{h}}^{(par)}$ est prépondérant sur $\mathcal{A}_{\mathbf{h}}^{(iid)}$ et $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ tend vers 0. A l'inverse pour une minorité de relations testées, $\mathcal{A}_{\mathbf{h}}^{(iid)}$ est prépondérant sur $\mathcal{A}_{\mathbf{h}}^{(par)}$. Ces cas de figure correspondent aux relations $\mathbf{h}$ telles que $\mathbf{h} \in \mathcal{C}^T$. L'identification de ces relations passe donc par le choix d'un seuil sur la valeur minimale du critère de décision $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$. Nous noterons S_a ce seuil et nous déciderons que :

$$\begin{cases} \text{Si } \frac{\mathcal{A}_{\mathbf{h}}^{(iid)}}{\mathcal{A}_{\mathbf{h}}^{(par)}} \geq S_a \text{ alors } \mathbf{h} \in \mathcal{C}^T \\ \text{Si } \frac{\mathcal{A}_{\mathbf{h}}^{(iid)}}{\mathcal{A}_{\mathbf{h}}^{(par)}} < S_a \text{ alors } \mathbf{h} \notin \mathcal{C}^T \end{cases}$$

N'ayant pas d'expression analytique pour ce seuil, le choix du seuil sera laissé à l'utilisateur à partir de l'observation de l'histogramme du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$.

Exemple 4.1.3.

Reprenons le code de Hamming de l'Exemple 4.1.2.. Pour les mêmes relations testées, l'histogramme du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ prend la forme suivante :

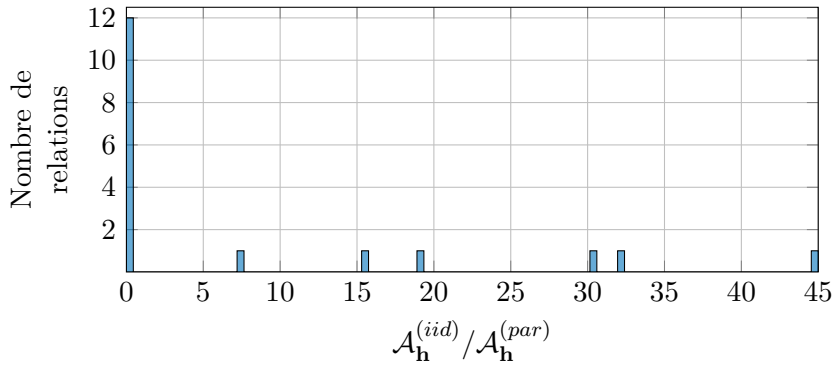


FIGURE 4.7 – Distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour le code $\mathcal{C}_7(7,4)$

> Nous pouvons vérifier que le critère $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ est quasiment nul pour les 12 relations testées qui n'appartiennent pas au dual du code de Hamming. Pour les 6 relations testées appartenant au dual du code de Hamming les valeurs du critère s'étendent de 7 à 45. Ici, choisir un seuil autour de 5 permet d'exclure les 12 relations erronées et d'identifier 6 relations de parité de poids 4 du code.

L'application de cette méthode repose donc sur le calcul du critère $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour tous les mots $\mathbf{h}$ de poids w afin de déterminer ceux pour lesquels le comportement est différent.

4.2 Méthode basée sur le paradoxe des anniversaires

4.2.1 Principe de la méthode

Nous nous intéressons maintenant à une méthode utilisée dans la littérature permettant de réduire la complexité de la recherche de relation de parité [22]. A partir de $\mathbf{y}$, nous créons la matrice d'interception binaire $\mathbf{M}$ dont les lignes correspondent aux L_{n_c} mots de code de longueur n_c reçus. Pour cela une décision est prise à partir des symboles de la séquence $\mathbf{y}$. La méthode présentée dans cette section utilise le paradoxe des anniversaires pour éviter une recherche exhaustive des relations de parité et ainsi réduire la complexité temporelle. La recherche exhaustive de relations de taille n_c et de poids w reviendrait à tester $\binom{n_c}{w}$ combinaisons. L'organisation des recherches autour du paradoxe des anniversaires permet de ne tester que $\binom{\lfloor n_c/2 \rfloor}{\lfloor w/2 \rfloor}.\binom{\lceil n_c/2 \rceil}{\lceil w/2 \rceil}$. A cette fin, la matrice d'interception $\mathbf{M}$ est scindée en deux sous-matrices d'interception $[\mathbf{M}_1\ \mathbf{M}_2]$ ($\mathbf{M}_1$ contient la première moitié des colonnes de $\mathbf{M}$ et $\mathbf{M}_2$ la seconde) comme illustré par la Figure 4.8. En l'absence de bruit, deux demi-relations $\mathbf{h}_1$ et $\mathbf{h}_2$ forment une relation de parité si :

$$\mathbf{h}_1 \cdot \mathbf{M}_1^T = \mathbf{h}_2 \cdot \mathbf{M}_2^T \tag{4.4}$$

Lorsque l'égalité (4.4) est vraie, on dit qu'il y a *collision*. L'idée du paradoxe repose sur le fait que la probabilité d'une collision est élevée. En procédant ainsi le nombre moyen de relations à tester avant d'en trouver une appartenant effectivement au code dual est peu élevé. Toutefois, cette compartimentation ne permet pas de trouver toutes les relations de parité car elle impose une contrainte sur la répartition des poids dans les relations testées. Une solution est d'itérer sur la méthode en appliquant une permutation aléatoire sur les colonnes de la matrice d'interception à chaque itération. De plus, l'égalité (4.4) n'est valable qu'en l'absence de bruit.

4.2.2 Algorithme de reconstruction par collision dans le cas bruité

Une amélioration de cette méthode consiste à relâcher la contrainte sur l'égalité (4.4). En effet, en présence de bruit, même lorsque $\mathbf{h} = [\mathbf{h}_1\ \mathbf{h}_2]$ appartient au code dual, l'égalité (4.4) n'est pas nécessairement vérifiée. Lorsque $\mathbf{h} \cdot [\mathbf{M}_1\ \mathbf{M}_2]^T \neq \mathbf{0}$ certains mots de la matrice $\mathbf{M}$ ne respectent pas la relation et empêchent la détection. Il convient donc de poser un seuil sur la proportion de mots de $[\mathbf{M}_1\ \mathbf{M}_2]$ pour lesquels l'égalité (4.4) peut ne pas être vérifiée. Notons p_s

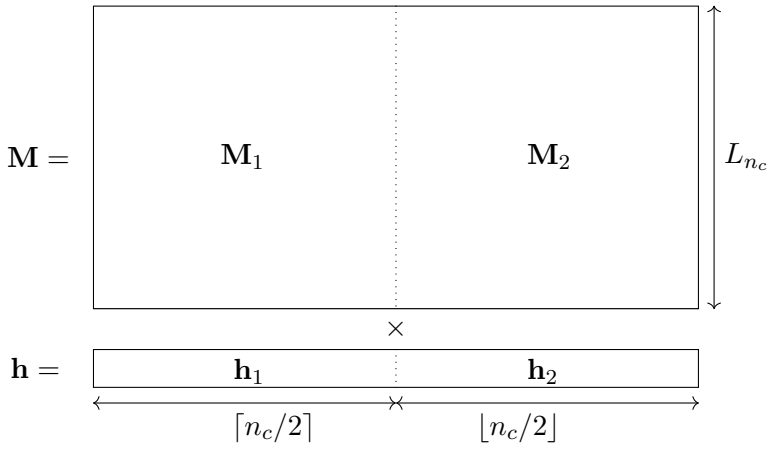


FIGURE 4.8 – Forme de la matrice pour le paradoxe des anniversaires

une telle proportion. Une relation testée sera considérée comme appartenant au code dual si le poids de $\mathbf{s} = \mathbf{h}_1 \cdot \mathbf{M}_1^T + \mathbf{h}_2 \cdot \mathbf{M}_2^T = \mathbf{h} \cdots \mathbf{M}^T$ est au plus de $L_{n_c} \cdot p_s$. Nous définissons alors $\mathcal{L}_w$ l'ensemble des relations $\mathbf{h}$ de poids w respectant cette condition :

$$\mathcal{L}_w = \{\mathbf{h} \in \mathrm{GF}(2)^{n_c}; w(\mathbf{h}_1 \cdot \mathbf{M}_1^T + \mathbf{h}_2 \cdot \mathbf{M}_2^T =) \leqslant L_{n_c} \cdot p_s, w(\mathbf{h}) = w\} \tag{4.5}$$

Dans [22], les auteurs posent un seuil S sur le poids de $\mathbf{s} = [\mathbf{h}_1\ \mathbf{h}_2] \cdot \mathbf{M}^T$. Si le poids de $\mathbf{s}$ est inférieur à S, alors on décide que $\mathbf{h} = [\mathbf{h}_1\ \mathbf{h}_2]$ est une relation de parité. Le choix de ce seuil est fait suivant la formule :

$$S = \frac{L_{n_c}}{2} - \sqrt{L_{n_c} \cdot \frac{n_c \cdot \log(2)}{2}} \tag{4.6}$$

par conséquent : $p_s = S/L_{n_c}$.

4.3 Réduction de la complexité de notre méthode

Le principe de collision ne s'adapte pas à notre méthode basée sur la distribution des distances euclidiennes. Toutefois, il est possible d'appliquer la division de la matrice d'interception pour réduire le nombre de relations testées lors d'une itération. En effet, notre méthode nous contraint à tester toutes les relations de taille n_c et de poids w, soit les $\binom{n_c}{w}$ combinaisons, pour obtenir un aperçu de la distribution des valeurs de notre critère et décider quelles sont celles qui sont effectivement des relations de parité. Il est donc possible de diviser la matrice d'interception en deux et de tester les relations $\mathbf{h}_1$ de poids $\lfloor w/2 \rfloor$ et $\mathbf{h}_2$ de poids $\lceil w/2 \rceil$ sur la matrice $\mathbf{M}_{soft}$. Dans ce cas, il faudra donc tester $\binom{\lfloor n_c/2 \rfloor}{\lfloor w/2 \rfloor} \cdot \binom{\lceil n_c/2 \rceil}{\lceil w/2 \rceil}$ relations $\mathbf{h}$. Par contre, toutes les relations de poids w ne sont pas testées. A l'instar de la méthode par collision, il est possible de trouver de nouvelles relations de poids w en réalisant une permutation aléatoire sur les colonnes de la matrice $\mathbf{M}_{soft}$.

Dans les Exemples 4.1.2. et 4.1.3. du code de Hamming $\mathcal{C}_7(7,4)$, nous avions appliqué ce principe. En effet, nous n'avions pas testé les $\binom{7}{4} = 35$ relations de poids $w = 4$ et de longueur 7 possible. Nous n'avions testé que $\binom{3}{2} \cdot \binom{4}{2} = 18$ relations de poids 4. Pour obtenir ces 18 relations, nous avons réalisé toutes les combinaisons des relations de poids 2 sur des mots de taille 3 (pour obtenir $\mathbf{h}_1$) et sur des mots de taille 4 (pour $\mathbf{h}_2$). Nous avons ensuite concaténé ces relations afin d'obtenir les 18 relations $\mathbf{h} = [\mathbf{h}_1\ \mathbf{h}_2]$ de poids $2 + 2 = 4$. Ensuite, pour chacune de ces 18 relations, nous avons calculé le critère $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ afin de décider de l'appartenance ou non de la relation $\mathbf{h}$ au code dual. Nous pouvons voir que le fait de ne tester qu'une partie des relations de poids 4 permet tout de même de détecter un ensemble de relation de parité tout en réduisant la complexité de notre méthode.

Dans la suite, nous appliquerons donc ce principe. Notre algorithme d'identification des relations de parité est résumé par l'algorithme 3.

Algorithme 3 Algorithme de reconstruction aveugle du code dual

Entrées : $\mathbf{M}_{soft}$, n_c, w, $\mathbb{E}(\mathbf{X}^{(par)})$, $\mathbb{E}(\mathbf{X}^{(iid)})$, N_B

Pour $\mathbf{h}_1 \in \mathrm{GF}(2)^{\lfloor w/2 \rfloor}$ **Faire**
  Pour $\mathbf{h}_2 \in \mathrm{GF}(2)^{\lceil w/2 \rceil}$ **Faire**
    $\mathbf{h} = [\mathbf{h}_1\ \mathbf{h}_2]$
    Ordonnancement des mots de $\mathbf{M}_{soft}$ de longueur w en suivant la Définition 10
    Pour $j \in [\![1, N_B]\!]$ **Faire**
      Calcul des distances deux à deux et ordonnancement : $\mathbf{x}_j^{(\mathbf{h})} = (x_j^{(\mathbf{h})}(0), x_j^{(\mathbf{h})}(1), \cdots, x_j^{(\mathbf{h})}(d-1))$
    Fin Pour
    Calcul des espérances empiriques des distances : $\hat{\mathbf{x}}^{(\mathbf{h})} = (\hat{x}^{(\mathbf{h})}(0), \hat{x}^{(\mathbf{h})}(1), \cdots, \hat{x}^{(\mathbf{h})}(d-1))$
    Calcul de l'adéquation à la distribution issue du code parité : $\mathcal{A}_{\mathbf{h}}^{(par)} = \sum_{i=0}^{d-1} (\hat{x}^{(\mathbf{h})}(i) - \mathbb{E}(X^{(par)}(i)))^2$
    Calcul de l'adéquation à la distribution issue d'une trame i.i.d. : $\mathcal{A}_{\mathbf{h}}^{(iid)} = \sum_{i=0}^{d-1} (\hat{x}^{(\mathbf{h})}(i) - \mathbb{E}(X^{(iid)}(i)))^2$
    Calcul du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$
  Fin Pour
Fin Pour
$\mathcal{L}_w = \{\mathbf{h}; \mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)} \geqslant S_a\}$

Sorties : Liste des relations identifiées comme appartenant au code dual $\mathcal{L}_w$

4.4 Étude comparative des résultats

Dans cette section, nous évaluons les performances de notre méthode de reconstruction. Pour cela nous étudierons les performances des deux codes suivantes :

- $\mathcal{C}_6$: code LDPC $\mathcal{C}_6(20, 10)$ de rendement $\frac{1}{2}$
- $\mathcal{C}_7$: code de Hamming $\mathcal{C}_7(7, 4)$ de rendement $\frac{4}{7}$

Les matrices génératrice et de parité de ces codes sont données dans l'Annexe B. Une représentation de la distribution des distances de Hamming est également proposée dans cette annexe.

Afin de quantifier l'intérêt de l'utilisation des informations souples pour la reconstruction d'un code correcteur d'erreurs nous proposons des résultats comparant les deux méthodes présentées ci-dessus pour le code $\mathcal{C}_6$. Dans chacune des méthodes un paramètre de seuil est à régler :

1- La proportion p_s pour la méthode par collision :
On fixe sa valeur comme expliqué dans la section 4.2.2.

2- Le seuil S_a sur le rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour la méthode par distribution des distances :
Ne disposant pas d'expression analytique pour ce seuil, le choix de sa valeur est laissé à la discrétion de l'utilisateur de la méthode sur la base de l'observation d'un histogramme du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$, tel que celui de l'Exemple 4.1.3.. En effet, dans cet exemple, pour le code de Hamming $\mathcal{C}_7(7,4)$ un seuil adéquat pourrait être par exemple placé autour de 5. Ce seuil permettrait de détecter les 6 relations de parité mises en exergue.

Enfin, nous ne connaissons pas la distribution des variables aléatoires $\mathbf{X}^{(par)}$ et $\mathbf{X}^{(iid)}$. Une solution consiste à tabuler des estimations des espérances $\mathbb{E}(X_i^{(par)})$ et $\mathbb{E}(X_i^{(iid)})$ pour tout i dans $[\![0, d-1]\!]$. Ces espérances dépendent de la quantité de bruit, du nombre de blocs et du nombre de mots par blocs (donc par extension du poids des relations recherchées : w). Pour effectuer la tabulation, il faut donc estimer la variance du bruit blanc additif gaussien σ_b^2 résultant du passage des bits par le canal de transmission. En effet, pour calculer $\mathcal{A}_{\mathbf{h}}^{(par)}$ et $\mathcal{A}_{\mathbf{h}}^{(iid)}$, il est nécessaire d'estimer $\mathbb{E}(X_i^{(par)})$ et $\mathbb{E}(X_i^{(iid)})$ pour les mêmes conditions de bruit que la trame reçue. Comme nous l'avons vu dans les chapitres précédents, il existe dans la littérature des méthodes permettant d'identifier la variance du bruit.

4.4.1 Observation des histogrammes du critère de détection pour le code $\mathcal{C}_6$

Comme nous pouvons le voir dans l'Annexe B, la matrice de parité du code $\mathcal{C}_6$ est composée de 10 relations de parité de poids 4. Par conséquent, nous nous sommes limités à la recherche des relations de parité indépendantes de poids 4 pour ce code. La Figure 4.9 représente un histogramme de la distribution empirique de notre critère $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 5 dB pour le code $\mathcal{C}_6(20,10)$ et des relations de poids $w=4$. Le nombre de relations $\mathbf{h}$ testées est de $\binom{10}{2}\cdot\binom{10}{2} = 2025$. Pour la plupart des relations, 2016 exactement, le critère a une valeur comprise entre 0 et 1 : ces relations n'appartiennent pas au code dual. Cependant, 9 relations voient leur valeur de critère s'écarter significativement du mode de cette distribution empirique. Ces 9 relations sont visibles sur la Figure 4.10 qui donne un agrandissement de la zone encadrée en noir dans la Figure 4.9. En effet, pour des valeurs du critère comprises entre 3 et 27, nous pouvons vérifier que nous avons 9 valeurs du critère pour lesquelles nous identifions à chaque fois une relation. Ces valeurs correspondent aux 9 relations $\mathbf{h}$ qui sont des relations de parité du code, soit $\mathbf{h} \in \mathcal{C}^T$. La sélection de ces relations passe donc par le choix d'un seuil sur la valeur minimale du critère de décision $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$. Par exemple, le choix d'un seuil à 2 permettra de détecter ces 9 relations de parité.

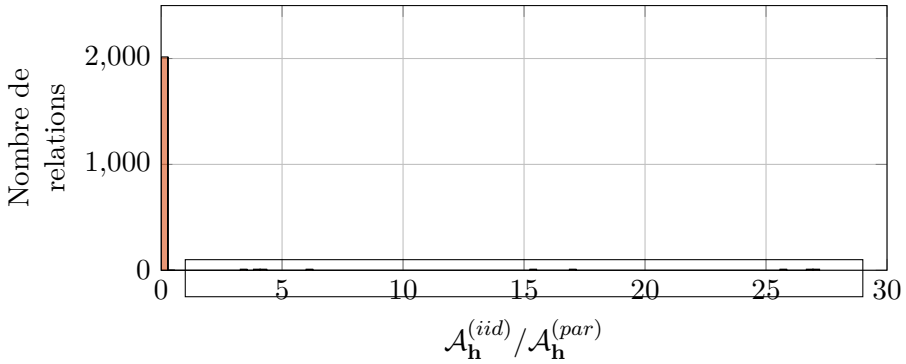


FIGURE 4.9 – Distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 5 dB pour le code $\mathcal{C}_6(20, 10)$

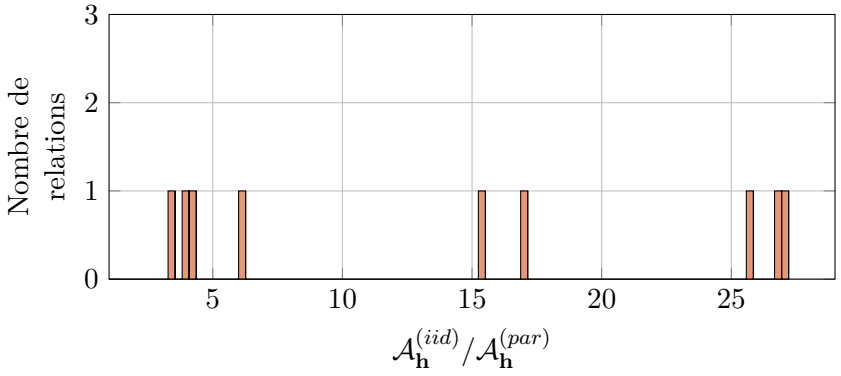


FIGURE 4.10 – Agrandissement sur la distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 5 dB pour des valeurs entre 1 et 28 pour le code $\mathcal{C}_6(20, 10)$

La Figure 4.11 représente cette fois la distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 0 dB. Comme dans le cas d'un niveau de bruit à 5 dB, nous remarquons que pour la plupart des relations le critère à une valeur comprise entre 0 et 1. Il sera donc aisé d'écarter ses relations qui ne seront donc pas considérée comme des relations de parité du code. Comme précédemment, la Figure 4.12 permet d'observer un agrandissement de la zone encadrée en noir dans la Figure 4.11. Nous pouvons voir que pour certaines relations le critère prend des valeurs différentes. Il y a 1 relation autour de la valeur 2 et 5 relations entre les valeurs 4 et 8. Un seuil autour de la valeur 2 permettrait d'identifier ces 6 relations comme étant des relations de parité du code. Toutefois, la relation qui a été détectée autour de 2 (encadrée en rouge), qui correspond à une valeur faible du critère, n'est pas une relation de parité de notre code. Faire le choix d'un seuil trop faible peut donc mener à de mauvaises détections. Dans cet exemple, faire le choix d'un seuil supérieur à 4 permet de détecter 5 relations de parité du code.

4.4.2 Impact du niveau de bruit

Comme nous venons de le voir, le choix du seuil dans notre méthode a un impact très important sur les performances de notre algorithme. En effet, si le seuil est trop bas on risque

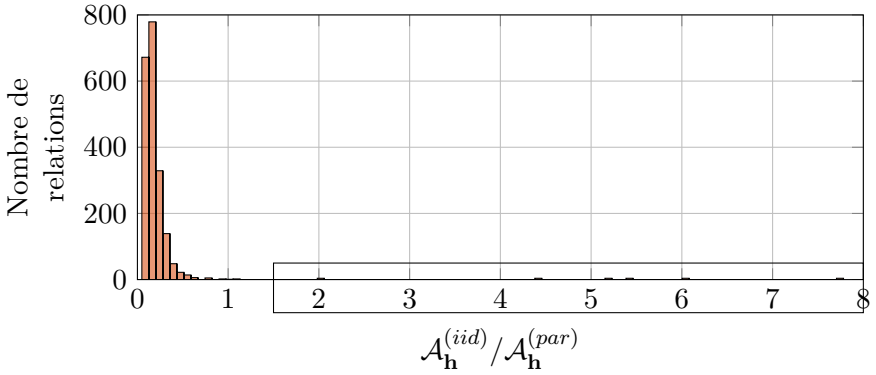


FIGURE 4.11 – Distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 0 dB pour le code $\mathcal{C}_6(20, 10)$

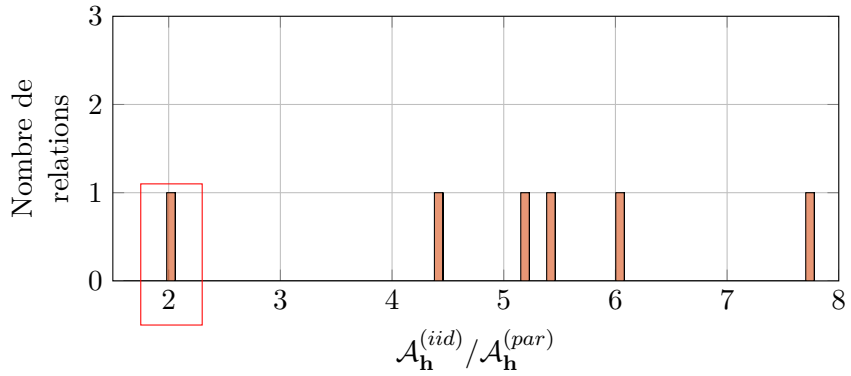


FIGURE 4.12 – Agrandissement sur la distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 0 dB pour des valeurs entre 1.5 et 8 pour le code $\mathcal{C}_6(20, 10)$

d'augmenter la probabilité de mauvaise détection. Cette probabilité correspond au fait de sélectionner des relations comme étant des relations de parité alors qu'elles ne le sont pas. Et au contraire, si le seuil est trop élevé, on risque d'augmenter la probabilité de non détection des relations de parité.

Analyse du code LDPC $\mathcal{C}_6(20, 10)$

N'ayant pas de formule théorique pour le choix du seuil S_a nous l'avons déterminé empiriquement. Les simulations ont été effectuées pour 200 mots reçus. Pour différentes valeurs de seuil et pour 1000 itérations de Monte-Carlo, nous avons compté le nombre de fois où des relations erronées étaient identifiées pour trouver le seuil minimal nécessaire. Ainsi, pour chaque valeur de seuil et à chacune de ces 1000 itérations, une nouvelle séquence de symboles bruités reçus est générée. Le nombre de blocs utilisés est fixé à $N_B = 10$ et 10 permutations sur les colonnes de la matrice d'interception sont réalisées afin d'identifier plus de relations de parité. La Figure 4.13 donne un aperçu de la probabilité de mauvaise détection, notée $\mathbf{P}_{md}$, en fonction de la valeur seuil sur le rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour différents niveaux de bruits. $\mathbf{P}_{md}$ correspond à la probabilité qu'au moins une relation identifiée est fausse. Pour des niveaux de bruit allant de -0.5

à 2 dB, ces résultats permettent de déterminer le seuil minimum pour lequel la probabilité de mauvaise détection est nulle. La table 4.1 recense les seuils optimaux pour différentes quantités de bruit au regard des résultats de cette simulation.

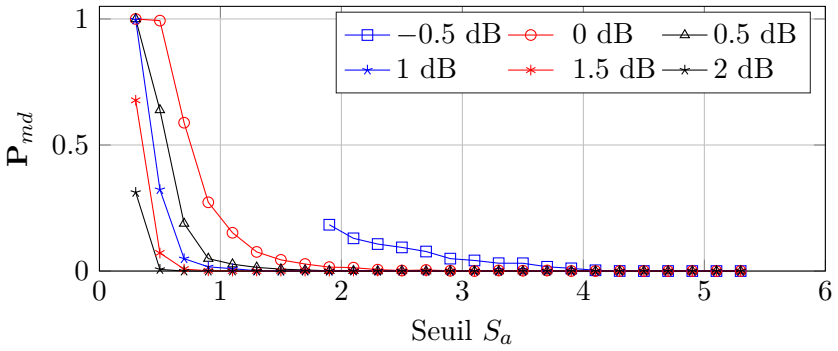


FIGURE 4.13 – Probabilité de mauvaise détection en fonction du seuil sur la valeur de $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour le code $\mathcal{C}_6(20, 10)$

TABLE 4.1 – Valeur du seuil optimal pour différents E_b/N_0 pour le code $\mathcal{C}_6(20, 10)$

E_b/N_0 (dB)	−0.5	0	0.5	1	1.5	2
Seuil S_a	4.5	3.9	2.7	2.7	1.1	0.7

En utilisant les valeurs du seuil optimal S_a (Tableau 4.1) pour notre méthode de reconstruction, nous pouvons comparer les performances des deux méthodes de reconstruction en terme de robustesse au bruit. Pour comparer les performances de ces méthodes, nous allons compter le nombre de vraies relations de parité indépendantes identifiées, nous le noterons N_h. En effet, pour mettre en oeuvre un décodeur il est nécessaire de construire une base du code dual. Sur la Figure 4.14, nous avons représenté N_h en fonction du niveau de bruit pour la méthode de reconstruction à partir des distances et celle par collisions. D'après cette figure, la méthode de reconstruction à partir des distances euclidiennes permet d'augmenter la robustesse aux bruits puisque pour un même niveau de bruit, nous sommes capables de trouver plus de relations de parité qu'avec la méthode par collision. Toutefois, ces résultats sont à prendre avec précaution puisque pour notre méthode nous n'avons pas obtenu d'expression pour le seuil S_a et nous avons donc utilisé la valeur du seuil optimal. De plus notre méthode est plus complexe à mettre en oeuvre que la méthode par collisions. En revanche, il est très certainement possible d'améliorer les performances de notre méthode en étudiant de plus près l'impact du nombre de blocs nécessaires et donc le nombre de données utiles afin de tirer profit au maximum de la fiabilité des mots.

Analyse du code de Hamming $\mathcal{C}_7(7, 4)$

Nous pouvons également analyser les performances de notre méthode sur le code de Hamming $\mathcal{C}_7(7, 4)$ utilisé dans les Exemples 4.1.2. et 4.1.3.. Nous cherchons à mesurer le nombre de relations

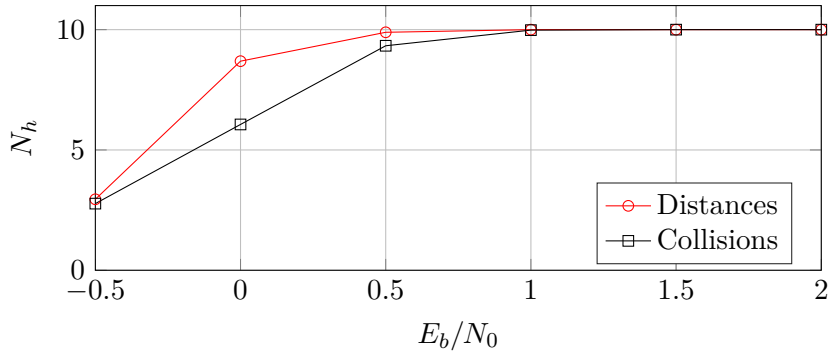


FIGURE 4.14 – Quantité N_h de relations identifiées en fonction du niveau de bruit pour le code $\mathcal{C}_6(20, 10)$

indépendantes identifiées pour des niveaux de bruit compris entre -0.5 et 3 dB. Comme pour le code précédent, les seuils optimaux ont été déterminés empiriquement : 1000 itérations de Monte-Carlo ont été effectuées pour chaque valeur de seuil testé. Les simulations ont été effectuées pour 200 mots reçus. Le nombre de blocs utilisés dans ces tests est de $N_B = 10$, le nombre d'itérations (en terme de permutations aléatoires appliquées sur les colonnes de la matrice d'interception) est de 10. La Figure 4.15 donne l'allure de la probabilité de mauvaise détection $\mathbf{P}_{md}$ en fonction de la valeur du seuil pour différents niveaux de bruit. La table 4.2 recense les seuils optimaux pour différentes quantités de bruit au regard des résultats de cette simulation.

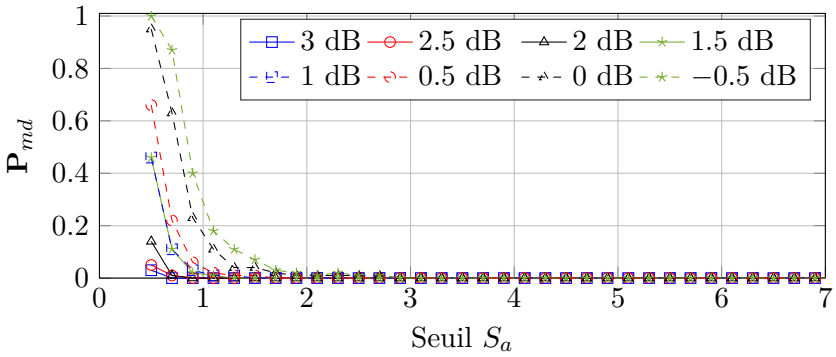


FIGURE 4.15 – Probabilité de mauvaise détection en fonction du seuil sur la valeur de $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour le code $\mathcal{C}_7(7, 4)$

TABLE 4.2 – Valeur du seuil optimal pour différents E_b/N_0 pour le code $\mathcal{C}_7(7, 4)$

E_b/N_0 (dB)	−0.5	0	0.5	1	1.5	2	2.5	3
Seuil S_a	5.9	5.3	3.1	2.9	2.7	1.3	1.1	1.1

En utilisant les valeurs du seuil optimal S_a (Tableau 4.2), nous obtenons les performances décrites par la Figure 4.16 pour le code de Hamming $\mathcal{C}_7$. Sur cette figure, le nombre N_h de relations indépendantes de poids 4 identifiées est représenté pour des niveaux de bruit variant entre -0.5 et 3 dB. La position des 1 dans la matrice de parité de ce code est donnée dans l'annexe B.

La matrice de parité est composée de 3 relations de parité de poids 4. Pour chaque valeur du niveau de bruit, 1000 essais de Monte-Carlo ont été effectués. Pour ce code, il existe uniquement 3 relations indépendante de poids 4. Nous pouvons voir qu'à partir de 2 dB, ces 3 relations de parité indépendantes de poids 4 ont été identifié par notre algorithme de reconstruction.

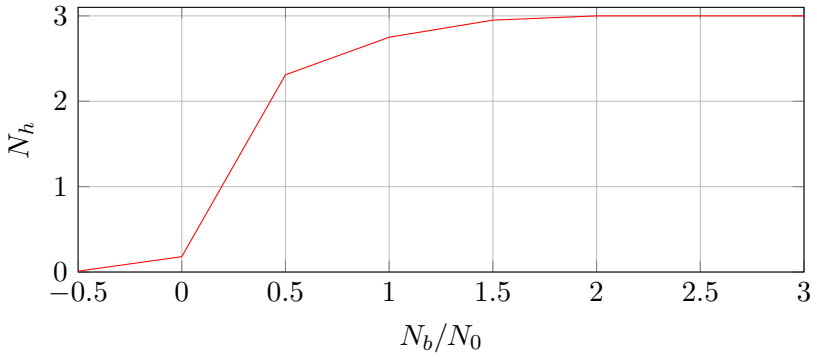


FIGURE 4.16 – Quantité N_h de relations identifiées en fonction du niveau de bruit pour le code $\mathcal{C}_7(7,4)$

4.5 Extension de la méthode à la reconstruction des codes convolutifs

Le principe de cette méthode peut être étendu à la reconstruction des codes convolutifs. Comme nous l'avons vu dans le chapitre 1, un code convolutif diffère d'un code en bloc par la mémoire introduite par le code convolutif. En effet, dans ce cas les mots de code de taille n_c ne dépendent pas uniquement d'un mot d'information de k_c bits mais aussi des $(K_c - 1)$ mots d'information précédents. La matrice de parité $\mathbf{H}(D)$, sous forme polynomiale systématique, est composée de $((n_c - k_c) \times n_c)$ polynômes générateurs de dégrée strictement inférieur à K_c. Elle est de la forme suivante :

$$\mathbf{H}(D) = \begin{pmatrix} h_{1,1}(D) & \cdots & h_{1,k_c}(D) & h_0(D) & & \\ \vdots & \cdots & \vdots & & \ddots & \\ h_{n_c-k_c,1}(D) & \cdots & h_{n_c-k_c,k_c}(D) & & & h_0(D) \end{pmatrix} \tag{4.7}$$

avec :

$$\forall i \in [\![1,\, n_c - k_c]\!],\ \forall j \in [\![1,\, k_c]\!],\ h_{i,j}(D) = \sum_{l=0}^{L} h_{i,j}(l) \cdot D^t \tag{4.8}$$

et :

$$h_0(D) = \sum_{l=0}^{L} h_0(l) \cdot D^t \tag{4.9}$$

$h_{i,j}(l)$ est le l-ième coefficient du polynôme $h_{i,j}(D)$, $h_0(l)$ est le l-ième coefficient du polynôme $h_0(D)$ et L est un entier inférieur ou égal à K_c.

Comme pour la matrice génératrice d'un tel code, il est possible d'écrire la matrice de parité d'un code convolutif sous la forme d'une matrice à coefficients binaires. Notons $\mathbf{H}$ cette matrice de taille infinie :

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_0 & & & & & \\ \mathbf{H}_1 & \mathbf{H}_0 & & & & \\ \vdots & \ddots & \ddots & & & \\ \mathbf{H}_{K_c-1} & \cdots & \mathbf{H}_1 & \mathbf{H}_0 & & \\ & \mathbf{H}_{K_c-1} & \cdots & \mathbf{H}_1 & \mathbf{H}_0 & \\ & & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \tag{4.10}$$

où $\forall l \in [\![0,\, K_c - 1]\!]$ $\mathbf{H}_l$ est une matrice de taille $((n_c - k_c) \times n_c)$ de la forme :

$$\mathbf{H}_l = \begin{pmatrix} h_{1,1}(l) & \cdots & h_{1,k_c}(l) & h_0(l) & & \\ \vdots & \ldots & \vdots & & \ddots & \\ h_{n_c-k_c,1}(l) & \cdots & h_{n_c-k_c,k_c}(l) & & & h_0(l) \end{pmatrix} \tag{4.11}$$

Cette matrice $\mathbf{H}$ est bien une matrice de parité du code si :

$$\mathbf{G} \cdot \mathbf{H}^T = 0$$

avec $\mathbf{G}$ la matrice génératrice d'un code convolutif décrite par l'équation 1.11. La structure de cette matrice de parité est particulière à cause de la mémoire introduite par le code.

4.5.1 Réorganisation de la matrice d'interception

Pour la reconstruction des relations de parité d'un code convolutif, la construction de la matrice d'interception diffère de celle avec les codes en bloc. En effet, il faut dans ce cas tenir compte de la mémoire introduite par le code. Pour un code convolutif, les relations de parité ne sont pas de taille n_c mais de taille :

$$n_a = n_c \cdot K_c$$

En regardant la structure de la matrice $\mathbf{H}$, nous pouvons voir que sa structure est basée sur la répétition du bloc de sous-matrices $\begin{pmatrix} \mathbf{H}_{K_c-1} & \cdots & \mathbf{H}_1 & \mathbf{H}_0 \end{pmatrix}$. Ce bloc est décalé de n_c bits à chaque ligne. C'est ce décalage qui traduit l'effet de la mémoire introduit par le code. Ce bloc correspond au $(n_c - k_c)$ relations de parité de taille $n_a = n_c \cdot K_c$.

Pour prendre en compte l'effet de mémoire, chaque ligne de la matrice d'interception doit correspondre à la concaténation de K_c mots de code générés consécutivement par le code convolutif. Ainsi, comme présenté dans [62], la matrice d'interception $\mathbf{B}$ obtenue est de la forme suivante :

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_0 & \mathbf{B}_1 & \cdots & \mathbf{B}_{K_c-1} \end{pmatrix} \tag{4.12}$$

où les matrices $\mathbf{B}_l$ sont de taille $((L_{n_c} - K_c + 1) \times n_c)$:

$$\mathbf{B}_l = \begin{pmatrix} c^{(1)}(l) & \cdots & c^{(n_c)}(l) \\ c^{(1)}(l+1) & \cdots & c^{(n_c)}(l+1) \\ \vdots & \cdots & \vdots \\ c^{(1)}(L_{n_c} - K_c + 1) & \cdots & c^{(n_c)}(L_{n_c} - K_c + 1) \end{pmatrix}, \quad \forall l = 0, \cdots, K_c - 1 \tag{4.13}$$

L'Exemple 4.5.1. permet d'illustrer l'organisation des bits reçus dans la matrice $\mathbf{B}$.

Exemple 4.5.1.

Considérons un code convolutif de longueur $n_c = 2$, de dimension $k_c = 1$ et de longueur de contrainte $K_c = 3$. Une séquence issue de ce code est donc de la forme :

$$\mathbf{y} = \begin{pmatrix} c^{(1)}(0)\, c^{(2)}(0) & c^{(1)}(1)\, c^{(2)}(1) & c^{(1)}(2)\, c^{(2)}(2) & c^{(1)}(3)\, c^{(2)}(3) & c^{(1)}(4)\, c^{(2)}(4) & c^{(1)}(5)\, c^{(2)}(5) \end{pmatrix}$$

Les mots de code sont réorganisés comme suit :

$$\mathbf{B} = \begin{pmatrix} c^{(1)}(0)\, c^{(2)}(0) & c^{(1)}(1)\, c^{(2)}(1) & c^{(1)}(2)\, c^{(2)}(2) \\ c^{(1)}(1)\, c^{(2)}(1) & c^{(1)}(2)\, c^{(2)}(2) & c^{(1)}(3)\, c^{(2)}(3) \\ c^{(1)}(2)\, c^{(2)}(2) & c^{(1)}(3)\, c^{(2)}(3) & c^{(1)}(4)\, c^{(2)}(4) \\ c^{(1)}(3)\, c^{(2)}(3) & c^{(1)}(4)\, c^{(2)}(4) & c^{(1)}(5)\, c^{(2)}(5) \end{pmatrix}$$

4.5.2 Observation de la distribution du critère de détection

Après avoir réorganisé les symboles reçus dans la nouvelle matrice d'interception, les relations $\mathbf{h}$ testées sont appliquées en masque comme illustré dans l'Exemple 4.1.1.. Seules les colonnes concernées sont conservées pour former des mots plus courts. De même que pour les codes en bloc, les distances euclidiennes deux à deux sont calculées. La série ordonnée de distances obtenue est comparée aux distributions obtenues dans le cas d'une trame i.i.d. et dans le cas d'une séquence issue d'un code de parité. Tout ceci permet de calculer une valeur du critère $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour chaque relation testée $\mathbf{h}$.

Code convolutif $\mathcal{C}(2, 1, 3)$

Nous avons réalisé des tests sur un code convolutif $\mathcal{C}(n_c = 2, k_c = 1, K_c = 3)$ dont les matrices génératrice et de parité sont :

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

et

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

Sous forme binaire, la relation de parité de taille $n_a = n_c \cdot K_c = 6$ obtenue à partir des équations 4.10 et 4.11 est :

$$\mathbf{h}_{\mathcal{C}(2,1,3)} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Cette relation est une relation de parité de poids 5. Pour ce code, la Figure 4.17 est un histogramme des valeurs du critère obtenues à partir d'une trame de symboles bruités à 5 dB. Le nombre de mots reçus est $L_{n_c} = 500$ et les relations testées sont de poids $w = 5$ et de longueur 6, elles sont au nombre de $\binom{6}{5} = 6$. La matrice d'interception $\mathbf{B}$ est alors de taille (498×6). La réorganisation proposée permet d'identifier une relation de parité de poids 5 : $\mathbf{h}_{\mathcal{C}(2,1,3)} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$. Cette relation est discriminée car elle génère une donnée aberrante du critère qui vaut environ 15. Pour toutes les autres relations testées, 5 exactement, le critère a une valeur comprise entre 0 et 1 : ces relations n'appartiennent pas au code dual. La relation identifiée est la seule vraie relation de poids 5 et de longueur 6. Dans cette configuration, le choix d'un seuil S_a supérieur à 1 permettra de détecter cette relation de parité.

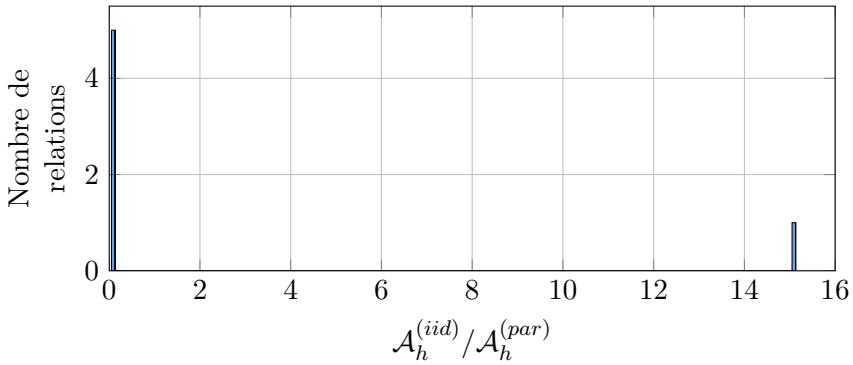


FIGURE 4.17 – Distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 5 dB pour $L_{n_c} = 500$ mots reçus avec le code $\mathcal{C}(2,1,3)$

Pour le même code convolutif, lorsque que le niveau de bruit augmente ($E_b/N_0 = 1$ dB) la même relation est toujours identifiable pour 500 mots de code reçus comme le montre la Figure 4.18. Cette fois, la valeur du critère obtenue pour $\mathbf{h}_{\mathcal{C}(2,1,3)}$ est plus faible (entre 2.2 et 2.3), mais le choix d'un seuil adéquat permet toujours de discriminer cette relation. De même, lorsque le nombre de mots reçus est plus faible, la détection de cette relation est encore possible. La Figure 4.19 est un histogramme du critère lorsque le nombre de mots reçus est de $L_{n_c} = 200$ à 5 dB. Encore une fois, la valeur de $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ est plus faible que dans le cas décrit par la Figure 4.17 (entre 4.5 et 5), mais reste suffisamment discriminante si le seuil est correctement choisi.

Code convolutif $\mathcal{C}(2,1,7)$

Enfin, nous avons aussi testé cette méthode pour un code convolutif avec une mémoire plus grande le $\mathcal{C}(2,1,7)$. La matrice génératrice $G(D)$ et la matrice de parité $H(D)$ sont telles que :

$$G(D) = \begin{bmatrix} 1 + D^2 + D^3 + D^5 + D^6 & 1 + D + D^2 + D^3 + D^6 \end{bmatrix}$$

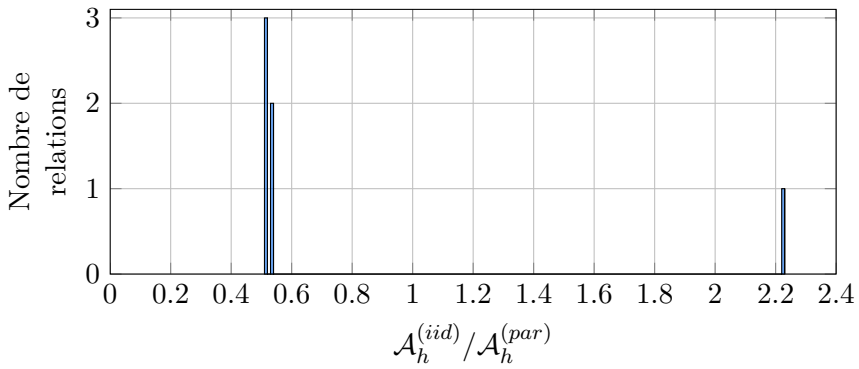


FIGURE 4.18 – Distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 1 dB pour $L_{n_c} = 500$ mots reçus avec le code $\mathcal{C}(2,1,3)$

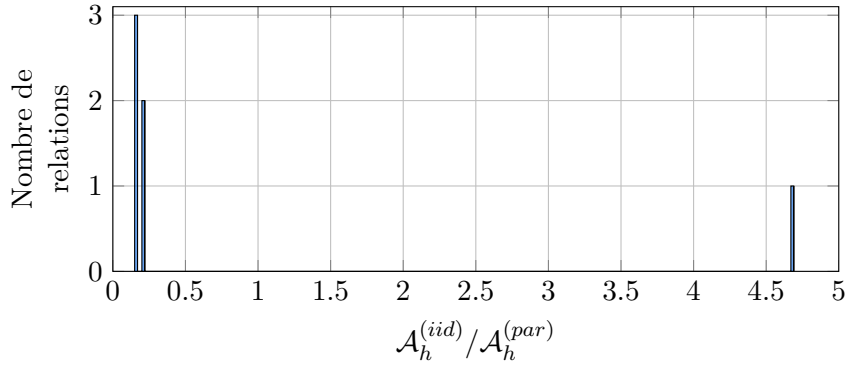


FIGURE 4.19 – Distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 5 dB pour $L_{n_c} = 200$ mots reçus avec le code $\mathcal{C}(2,1,3)$

et

$$H(D) = \begin{bmatrix} 1 + D + D^2 + D^3 + D^6 & 1 + D^2 + D^3 + D^5 + D^6 \end{bmatrix}$$

Sous forme binaire, la relation de parité de taille $n_a = n_c \cdot K_c = 14$ obtenue à partir des équation 4.10 et 4.11 est :

$$\mathbf{h}_{\mathcal{C}(2,1,7)} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Cette relation est une relation de parité de taille 14 et de poids 10.

Le nombre de mots reçu est de 500 et le niveau de bruit est de 5 dB. Les relations testées sont de poids $w = 10$ et de longueur 14, elles sont au nombre de $\binom{14}{10} = 1001$. La matrice d'interception $\mathbf{B}$ est alors de taille (494×14). Comme précédemment, il n'y a qu'une seule relation de poids 10 à identifier : $\mathbf{h}_{\mathcal{C}(2,1,7)}$. La Figure 4.20 donne l'histogramme du critère. Comme précédemment la détection de la relation est possible en choisissant le seuil S_a adéquat puisque pour 1000 des relations testées la valeur du critère est proche de 0.65 et que pour une relation le critère vaut 1.5. L'occurrence de cette relation est visible sur la Figure 4.21 qui est un agrandissement de la zone encadrée en noir dans la Figure 4.20. Le critère $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ prend des valeurs entre

0.5 et 0.8 pour les 1000 relations testées qui n'appartiennent pas au code dual et le critère vaut environ 1.5 quand $\mathbf{h}_{\mathcal{C}(2,1,7)}$ est utilisée comme masque.

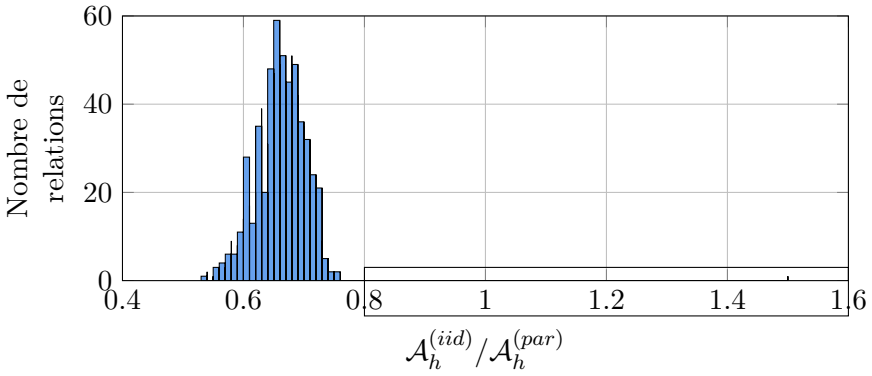


FIGURE 4.20 – Distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ à 5 dB pour $L_{n_c} = 500$ mots reçus avec le code $\mathcal{C}(2, 1, 7)$

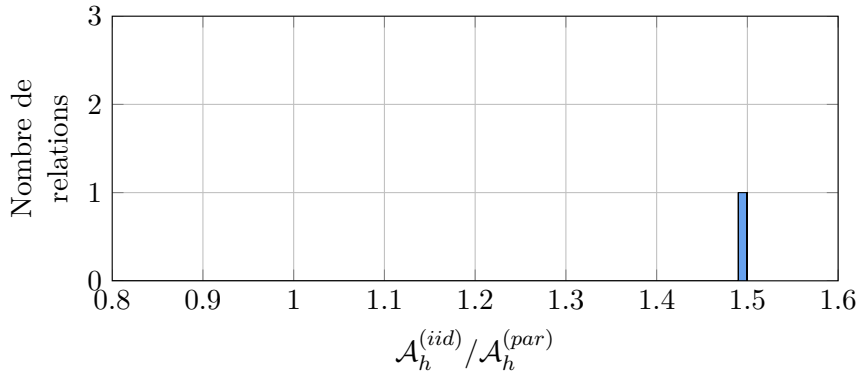


FIGURE 4.21 – Agrandissement sur la distribution du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$ pour des valeurs entre 0.8 et 1.6 à 5 dB pour $L_{n_c} = 500$ mots reçus avec le code $\mathcal{C}(2, 1, 7)$

La procédure complète d'identification n'a pas été appliquée pour les codes convolutifs. En effet, il n'y a pas eu d'ordonnancement par fiabilité des lignes de la matrice d'interception et l'ensemble des mots de code a été utilisé dans cette expérience. De plus, il n'y a pas eu de test de performances pour les codes convolutifs. Mais, ces premiers résultats nous permettent de montrer que notre méthode de reconstruction peut être adaptée au cas des codes convolutifs.

4.6 Conclusion

Dans ce chapitre, nous avons proposé une méthode de reconstruction d'un code linéaire. La méthode repose sur l'observation de la distribution des distances euclidiennes entre les mots formés par les symboles prenant part à une relation de parité. Lorsque ces symboles sont liés par une relation de parité, la distribution des distances est similaire à celle obtenue pour les mots d'un code de parité. Cette particularité permet de discriminer l'existence d'une relation de parité.

Nous avons comparé cette méthode avec une méthode par collision. Nous avons montré que notre méthode pouvait permettre d'obtenir des résultats sensiblement meilleurs que celle par collision. En revanche nous avons mis en avant que les tests et analyses réalisés pour notre méthode n'étaient pas suffisant pour conclure sur ses performances réelles. En effet, pour notre méthode, il est nécessaire de mettre en oeuvre une expression pour déterminer le seuil sur la valeur du rapport $\mathcal{A}_{\mathbf{h}}^{(iid)}/\mathcal{A}_{\mathbf{h}}^{(par)}$. L'établissement de ce seuil pourrait passer par la formalisation de ce problème comme un test d'hypothèse pour lequel il faudrait choisir une probabilité de fausse alarme par exemple. La complexité algorithmique de cette méthode nous pousse à penser que cette méthode est particulièrement adaptée aux codes courts et à la résolution du sous-problème évoqué en introduction de ce chapitre. En effet, la recherche exhaustive de relations de parité demande de tester un nombre de combinaisons qui augmente grandement avec la longueur du code n_c. Par ailleurs, il est nécessaire de discuter de l'influence du nombre de blocs utilisés N_B sur l'efficacité de la méthode. Lorsque N_B est grand, l'évaluation de la distribution moyenne des distances ordonnées est plus fiable statistiquement car un grand nombre d'observations est utilisée. Toutefois, si N_B est trop grand, nous perdons l'avantage de la diminution artificielle du niveau de bruit. Il faudrait donc trouver un compromis et évaluer une valeur adéquat de N_B en fonction du niveau de bruit.

Enfin, le critère employé dans cette méthode n'est pas contraint par la famille de code en bloc binaire. Les codes LDPC sont toutefois des candidats privilégiés à la recherche de relations de parité de poids faibles. Les codes convolutifs étant des codes relativement courts, ils sont aussi des bons candidats pour cette méthode de reconstruction. Les premiers tests effectués pour des codes convolutifs sont d'ailleurs plutôt positifs. En effet, pour ce type de code, nous avons montré qu'en réorganisant la matrice d'interception afin de tenir compte de la longueur contrainte du code, il était possible d'appliquer notre méthode de reconstruction. La connaissance de la longueur de contrainte du code est également nécessaire pour reconstruire les relations de parité d'un code convolutif. Il serait donc intéressant de développer une nouvelle méthode, basée sur les distances euclidiennes, afin d'identifier l'ensemble des paramètres d'un code convolutif n_c, k_c et K_c.

Conclusion et perspectives

La multiplication des objets communicants et des normes associées est un grand défi pour les communications numériques. L'objectif est de maintenir une qualité de service malgré l'encombrement du spectre et les incompatibilités créées entre les différentes normes et objets connectés. Des solutions sont trouvées dans la radio cognitive et dans l'AMC par la création de récepteurs intelligents capable de s'adapter à un changement des conditions environnementales et à des paramètres d'émission changeants. Ces récepteurs doivent alors identifier ces paramètres en semi-aveugle. Cette problématique apparaît aussi dans le cadre d'applications de surveillance et de sécurité. En effet, dans ce contexte une interception de signaux peut demander l'identification en aveugle des paramètres d'émission sans connaissances a priori. L'identification de ces paramètres se fait alors dans un contexte non-coopératif. Dans ce manuscrit, nous nous sommes concentrés sur l'identification en aveugle des paramètres de l'étape de codage de canal. Notre étude a porté sur la reconstruction des codes correcteurs d'erreurs binaires linéaires. Dans la littérature, la plupart des méthodes permettant d'identifier ces codes s'appuient sur l'information binaire reçue, soit des informations fermes. L'approche proposée ici se voulait originale en basant toutes les méthodes présentées sur l'utilisation d'informations souples octroyant une mesure de fiabilité sur les bits reçus. Nous avons donc développé deux méthodes d'identification de paramètres tels que la longueur du code et une méthode de reconstruction pour identifier des relations de parité du code. Toutes ces méthodes utilisent les distances euclidiennes comme métrique discriminante.

Dans le chapitre 2, nous avons présenté des méthodes permettant d'identifier la longueur et la dimension d'un code en bloc linéaire. Ces méthodes utilisent des informations souples. Ce chapitre introduit le concept de la déficience du nombre de classes. En effet, la méthode d'identification de la longueur du code repose sur la disposition particulière des mots de code bruités dans l'espace euclidien. A cause de la redondance introduite par le code, cet espace est lacunaire comparé à un espace peuplé de mots i.i.d.. C'est ce phénomène que nous proposons de détecter pour mettre en exergue la présence d'un code et identifier sa longueur. Pour cela, la séquence bruitée reçue est divisée en blocs contigus de tailles identiques. Ces blocs sont ensuite regroupés en classes. Chaque classe est formée sur un critère de distance. Les blocs appartenant à une même classe sont proches au sens de la distance euclidienne. Lorsque la taille des blocs atteint la longueur du code, le nombre de classes obtenues est plus faible que celui attendu pour des blocs i.i.d. de cette longueur. La méthode a été comparée à une méthode de la littérature basée sur la déficience de rang qu'elle surpasse en terme de robustesse au bruit. Toutefois,

la complexité de cette méthode la rend peu viable pour des codes longs. Par conséquent elle semble tout indiquée pour la reconnaissance de codes utilisés dans les communications de l'IoT qui doivent être des codes relativement courts. Dans ce chapitre, une méthode d'identification de la dimension d'un code a également été proposée. Cette méthode repose sur la notion de collision et le paradoxe des anniversaires : la probabilité de recevoir deux fois le même mots de code est relativement élevée si la séquence dont nous disposons est assez longue. En présence de bruit nous définissons que deux mots entrent en collisions si leur distance est inférieure à un seuil donné. Compter le nombre de collision nous permet ensuite d'identifier la dimension du code. Ces travaux ont donné lieu à deux publications [8, 9].

La méthode par déficience du nombre de classe a introduit les phénomènes et les particularités de la distribution des distances que nous souhaitions observer. Bien que cette méthode offre de bonnes performances, nous avons vu que sa complexité rendait son utilisation rédhibitoire pour des codes longs. Nous avons donc proposé une autre méthode d'identification de la longueur d'un code en bloc linéaire. Cette méthode s'appuie sur une autre particularité de la distribution des distances imposée par un code : la distance minimale. Comme précédemment, la séquence bruitée reçue est divisée en blocs contigus de tailles identiques. Les distances deux à deux entre ces blocs sont calculées pour ne conserver que la distance minimale pour chaque bloc. Les distances minimales ainsi obtenues ont une distribution qui se dissocie nettement de celle obtenue pour des blocs i.i.d.. Cette distinction permet d'identifier la longueur du code. Cette méthode est plus performante que la méthode par déficience de rang. Elle surpasse aussi la méthode par déficience du nombre de classes que nous avons proposée dans le chapitre 2 en terme de robustesse au bruit lorsque la longueur de la séquence reçue augmente. Cette méthode basée sur l'observation des distances minimales est plus adaptée que la méthode par déficience de classe à l'identification des codes plus longs. Toutefois, cette méthode appelle à trouver un critère d'arrêt pour limiter le nombre de taille de bloc à tester. Ces travaux ont donné lieu à une publication [10].

L'étape ultime permettant le décodage de la séquence reçue est la reconstruction à proprement parler du code correcteur d'erreurs. Cette étape consiste à identifier des éléments du code dual : les relations de parité. A partir d'un ensemble de relations de parité formant une base du code dual, il est possible de mettre en oeuvre un décodeur afin de décoder la trame reçue pour retrouver le message original. Par soucis de complexité, les relations de parité identifiées par notre algorithmes sont des relations de poids faible. Nous avons donc proposé une méthode de reconstruction s'appuyant sur la distribution des mots formés par les symboles bruités impliqués dans une relation de parité. Lorsque des bits sont impliqués dans une relation de parité, ils forment un mot de code de parité. Comme vu dans les chapitre 2 et 3, en présence d'un code les blocs de symboles issus de mots de code ont une distribution particulière dans l'espace euclidien. Il en va de même avec les mots d'un code de parité. Cette particularité nous permet de détecter la présence d'une relation de parité. En effet, lorsqu'une relation de parité est testée, seuls les symboles impliqués dans celle-ci sont conservés dans l'ensemble des mots reçus. En comparant

la distribution des distances euclidiennes pour les nouveaux mots formés à celles obtenues pour les mots d'un code de parité et des mots i.i.d., il est possible de déterminer si la relation testée appartient au code dual ou pas. Nous avons comparé cette méthode avec une méthode par collision qui est aussi adapté à la recherche de relations de parité de poids faible. Nous avons montré que notre méthode semblait offrir de meilleures performances que celle par collision. En revanche, nous avons mis en avant que l'étude de certains paramètres restait à faire, comme la quantité de symboles reçus, ou le nombre de blocs de mots de code traités parmi ceux reçus. Cette étude devrait sans doute permettre d'améliorer les performances de notre méthode. De plus, cette méthode ne permet pas de discriminer les relations testées individuellement. Il serait certainement plus commode de faire un test d'adéquation à l'une et l'autre des distributions de distances euclidiennes pour chaque relation testée en posant une seuil sur une probabilité de fausse alarme. Enfin, la méthode paraît tout à fait adaptée à la reconstruction de code LDPC, qui sont par nature des codes ayant des relations de parité de poids faible. Les codes convolutifs étant des codes relativement court, l'utilisation de cette méthode de reconstruction est tout indiquée. Les premiers tests que nous avons pu faire sur cette famille de code sont encourageant.

L'ensemble des méthodes présentées dans ce manuscrit sont nées avec l'idée que l'utilisation de valeurs fiabilisées pour l'identification de codes correcteurs d'erreurs est paradoxalement peu fréquente alors qu'elle est souvent préconisée lors des étapes de décodage correcteurs d'erreurs. Nous avons donc pris le parti d'utiliser des symboles binaires bruités et la distance euclidienne pour métrique pour effectuer ces travaux d'identification et de reconstruction. Dans ce contexte, il a été nécessaire de voir quel était l'incidence de la présence d'un code sur ce type de données. Les résultats présentés se concentrent sur les codes en blocs linéaires, cependant, l'adaptation de ces méthodes aux codes poinçonnés et aux codes convolutifs semble être une piste viable à explorer. Leur extension au cas des turbo-code et plus généralement aux codes concaténés est également une voie à développer. Étant donné que les méthodes d'identification de la longueur d'un code et la méthode de reconstruction s'appuient sur l'observation de distances, il serait intéressant de faire ces deux étapes de manière simultanée. Ceci nous mènerait naturellement à l'estimation du rendement du code. Enfin, toutes ces méthodes utilisent la comparaison à des distributions de distances dans le cas d'une séquence i.i.d.. Il serait intéressant d'obtenir des expressions décrivant ces distributions pour réduire la complexité de nos algorithmes et possiblement définir des critères d'arrêt.

Annexe A

Calcul de la probabilité de détection $\mathbf{P}_d$

Dans cette annexe, le détail du calcul de la probabilité $\mathbf{P}_d$ que deux blocs entrent en collision est présenté.

Soit $Y(k) = S(k)+B(k)$ la variable aléatoire portant le k-ième symbole bruité de la séquence reçue $\mathbf{y}$ avec $S(k) \sim \mathcal{U}(\{-1, +1\}$ et $B(k) \sim \mathcal{N}(0, \sigma_b^2)$. Si $X(p) = (Y(i \cdot n + p) - Y(j \cdot n + p))^2$, alors $X(0), X(1), X(2), \cdots, X(n-1)$ est une séquence de variables aléatoires i.i.d.. Ces variables sont mutuellement indépendantes et chacune d'entre elles suit la même loi de probabilité. Notons μ_X et σ_X^2 la moyenne et la variance de cette loi. A partir d'une approximation issue du théorème central limite [7], pour n_c grand, nous pouvons considérer que $\sum_{p=0}^{n-1} X(p) = X(0)+X(1)+\cdots+X(n-1)$ est une variable aléatoire normale de moyenne $n \cdot \mu_X$ et de variance $n \cdot \sigma_X^2$. Par conséquent, $X_{d^2} = \sum_{p=0}^{n-1}(Y(i \cdot n + p) - Y(j \cdot n + p))^2 = \sum_{p=0}^{n-1} X(p)$ est distribué comme une variable aléatoire normale de moyenne $\mu_{d^2} = n \cdot \mu_X$ et de variance $\sigma_{d^2}^2 = n \cdot \sigma_X^2$.

Dans la section 2.3.1 nous nous intéressons à la probabilité que deux blocs i et j, avec $i \neq j$, entrent en collision dans le cas où ils sont deux versions bruités du même mot de code de taille $n = n_c$:

$$\begin{aligned} \mathbf{P}_d &= \mathbb{P}(d_E(\mathfrak{B}_i^{(n_c)}, \mathfrak{B}_j^{(n_c)}) \leq \beta_{col} | \, \mathbf{s}_i = \mathbf{s}_j) \\ &= \mathbb{P}(X_{d^2} \leq \beta_{col}^2 | \, \mathbf{s}_i = \mathbf{s}_j) \\ &= \mathbb{P}(\sum_{p=0}^{n_c-1} X(p) \leq \beta_{col}^2 | \, \mathbf{s}_i = \mathbf{s}_j) \end{aligned}$$

D'après la condition $\mathbf{s}_i = \mathbf{s}_j$, la variable aléatoire $X(p)$ est telle que $X(p) = (B(i \cdot n_c + p) - B(j \cdot n_c + p))^2$. De plus, $(B(i \cdot n_c + p) - B(j \cdot n_c + p))$ suit une loi normale de moyenne nulle et de variance $2.\sigma_b^2$. Dans ces circonstances, nous pouvons conclure que $\frac{X_{d^2}}{2\sigma_b^2}$ suit une loi du khi-2 centrée et réduite : $\frac{X_{d^2}}{2\sigma_b^2} \sim \chi^2(n_c)$. Par conséquent, la fonction de répartition correspondante est la suivante :

$$\mathbf{P}_d = \frac{\gamma\left(\frac{n_c}{2}, \frac{\beta_{col}^2}{4\sigma_b^2}\right)}{\Gamma\left(\frac{n_c}{2}\right)}$$

où $\Gamma(\cdot)$ et $\gamma(\cdot, \cdot)$, correspondent respectivement à la fonction gamma :

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$$

et à la fonction gamma incomplète :

$$\gamma(a, x) = \int_0^{x} t^{a-1} e^{-t} dt$$

Annexe B

Matrices génératrices et matrices de parité des codes utilisés

Dans cette annexe, une représentation graphique des matrices génératrices et de parité des codes utilisés dans les chapitres 2, 3 et 4 est donnée. Chaque carré noir donne les positions des 1 et les carrés blancs celles des 0. Pour chacun de ces codes, la distribution des distances de Hamming en l'absence de bruit est donnée, ainsi que celle pour des mots i.i.d. de même longueur.

B.1 Code LDPC : $\mathcal{C}_1(25, 10)$

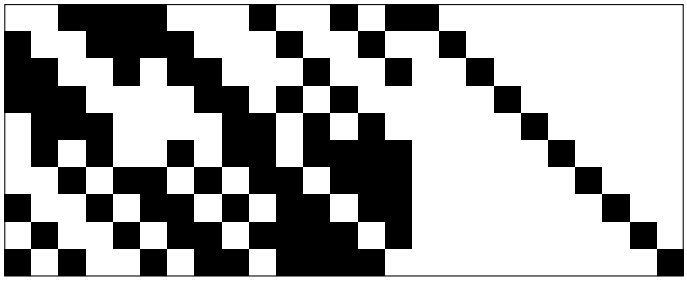


FIGURE B.1 – Matrice génératrice du code $\mathcal{C}_1$: $n_c = 25$ et $k_c = 10$

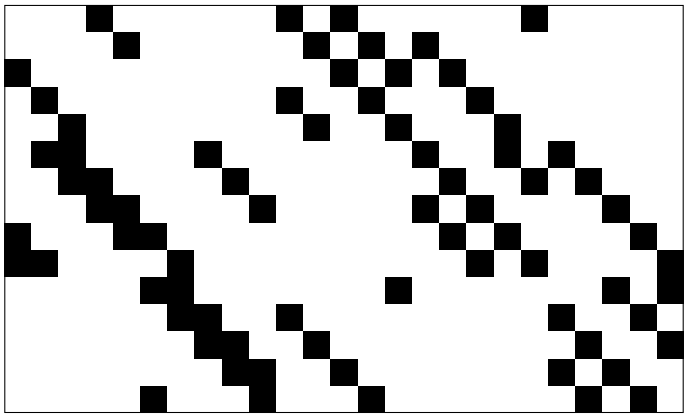


FIGURE B.2 – Matrice de parité du code $\mathcal{C}_1$: $n_c = 25$ et $k_c = 10$

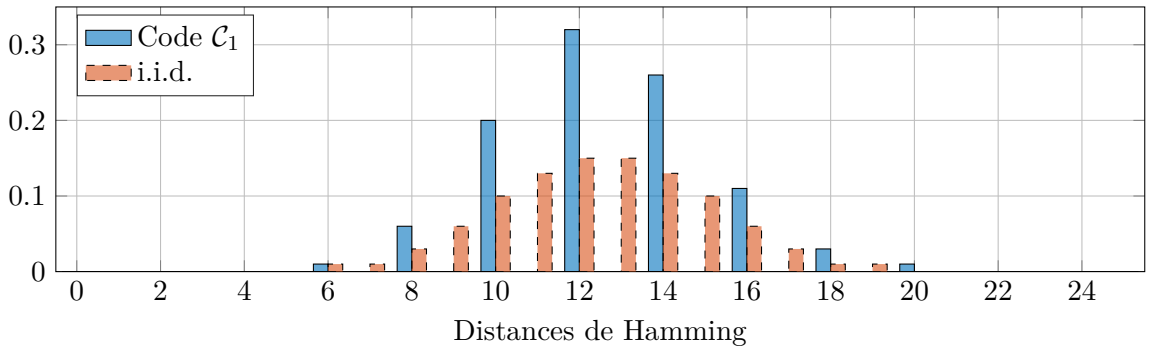


FIGURE B.3 – Distribution des distances en l'absence de bruit pour le code LDPC $\mathcal{C}_1(25, 10)$ et pour des mots i.i.d. de longueur 25

B.2 Code LDPC : $\mathcal{C}_2(25, 20)$

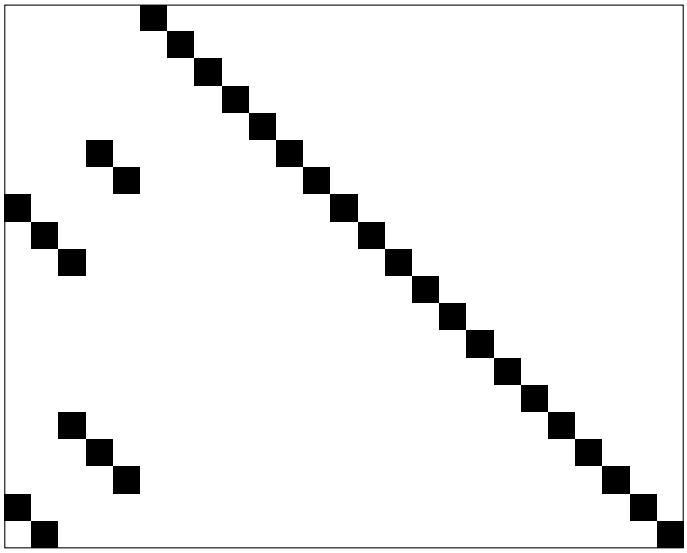


FIGURE B.4 – Matrice génératrice du code $\mathcal{C}_2$: $n_c = 25$ et $k_c = 20$

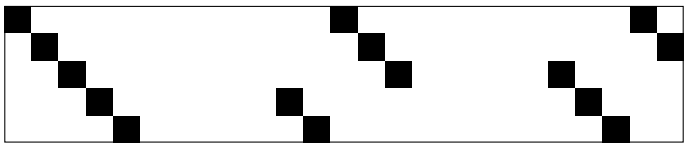


FIGURE B.5 – Matrice de parité du code $\mathcal{C}_2$: $n_c = 25$ et $k_c = 20$

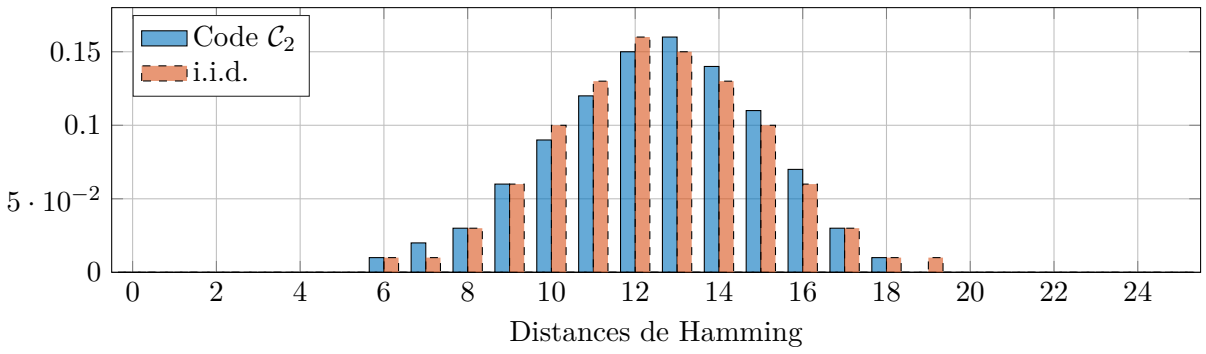


FIGURE B.6 – Distribution des distances en l'absence de bruit pour le code LDPC $\mathcal{C}_2(25, 20)$ et pour des mots i.i.d. de longueur 25

B.3 Code LDPC : $\mathcal{C}_3(15, 10)$

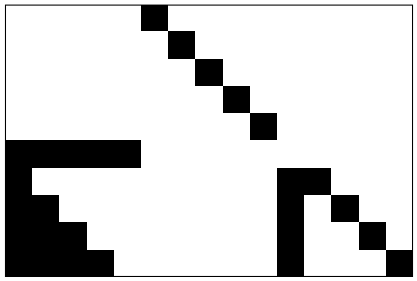


FIGURE B.7 – Matrice génératrice du code $\mathcal{C}_3$: $n_c = 15$ et $k_c = 10$

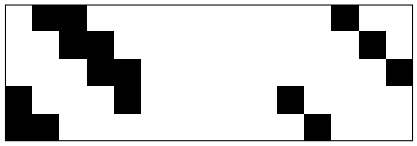


FIGURE B.8 – Matrice de parité du code $\mathcal{C}_3$: $n_c = 15$ et $k_c = 10$

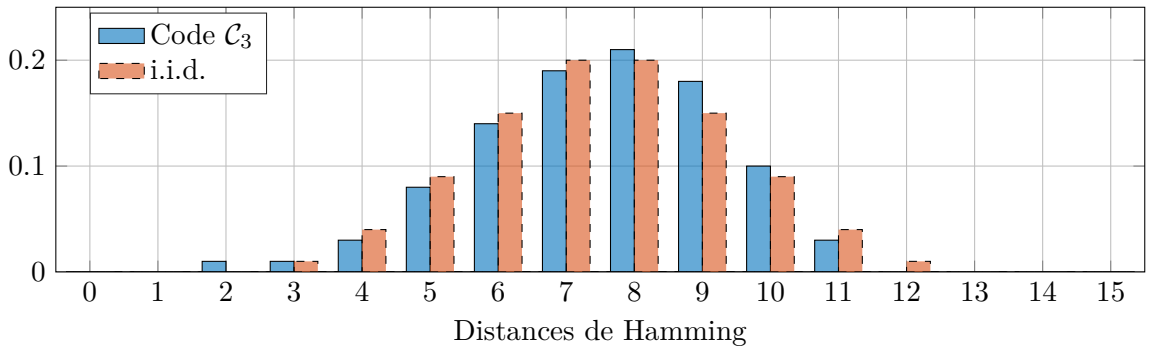


FIGURE B.9 – Distribution des distances en l'absence de bruit pour le code LDPC $\mathcal{C}_2(15, 10)$ et pour des mots i.i.d. de longueur 15

B.4 Code LDPC : $\mathcal{C}_4(32, 16)$

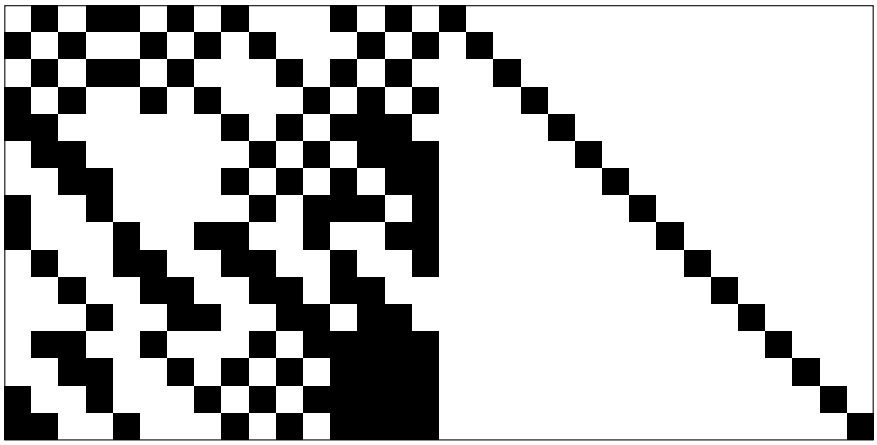


FIGURE B.10 – Matrice génératrice du code $\mathcal{C}_4$: $n_c = 32$ et $k_c = 16$

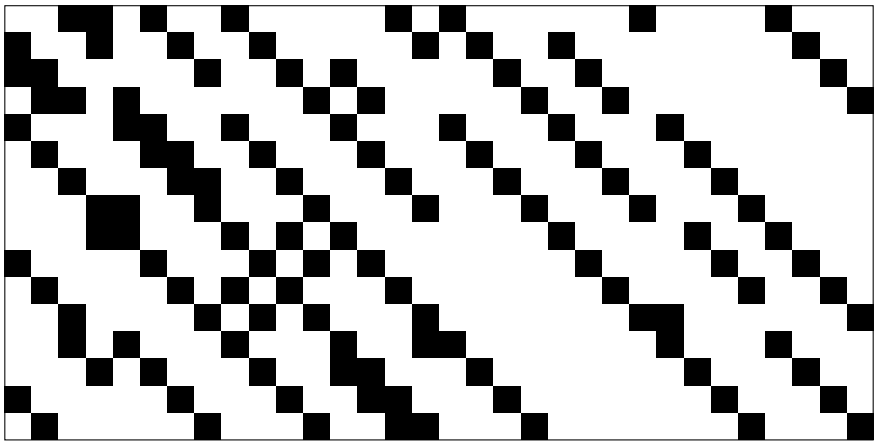


FIGURE B.11 – Matrice de parité du code $\mathcal{C}_4$: $n_c = 32$ et $k_c = 16$

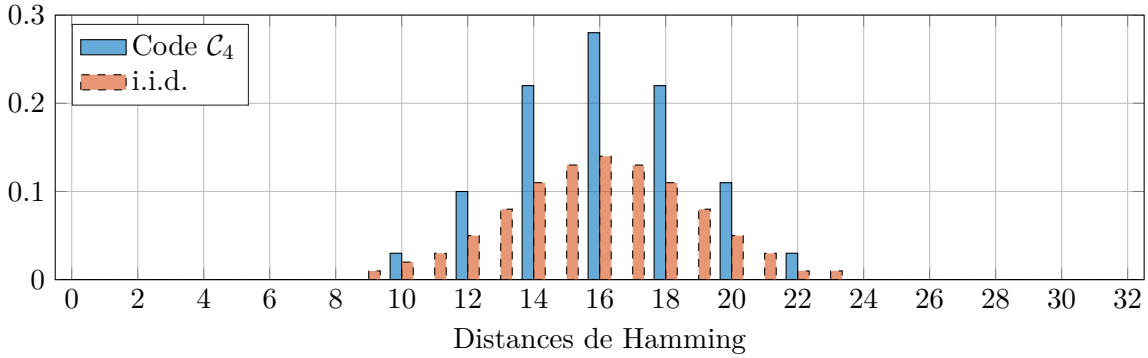


FIGURE B.12 – Distribution des distances en l'absence de bruit pour le code LDPC $\mathcal{C}_4(32, 16)$ et pour des mots i.i.d. de longueur 32

B.5 Code de Hamming : $\mathcal{C}_5(15, 11)$

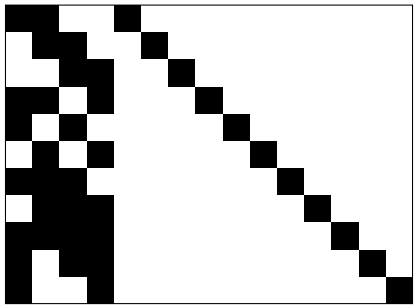


FIGURE B.13 – Matrice génératrice du code $\mathcal{C}_5$: $n_c = 15$ et $k_c = 11$

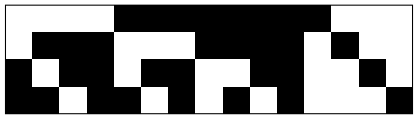


FIGURE B.14 – Matrice de parité du code $\mathcal{C}_5$: $n_c = 15$ et $k_c = 11$

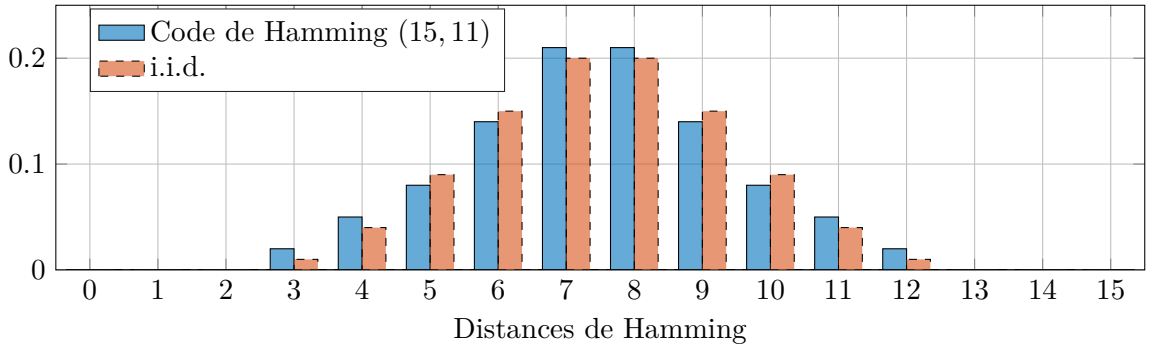


FIGURE B.15 – Distribution des distances en l'absence de bruit pour le code de Hamming $\mathcal{C}_5(15, 11)$ et pour des mots i.i.d. de longueur 15

B.6 Code LDPC : $\mathcal{C}_6(20, 10)$

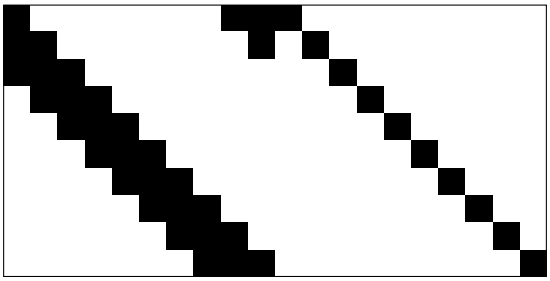


FIGURE B.16 – Matrice génératrice du code $\mathcal{C}_6$: $n_c = 20$ et $k_c = 10$

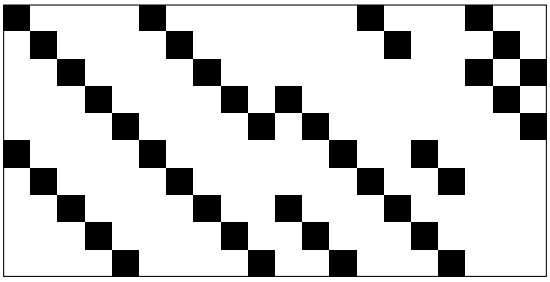


FIGURE B.17 – Matrice de parité du code $\mathcal{C}_6$: $n_c = 20$ et $k_c = 10$

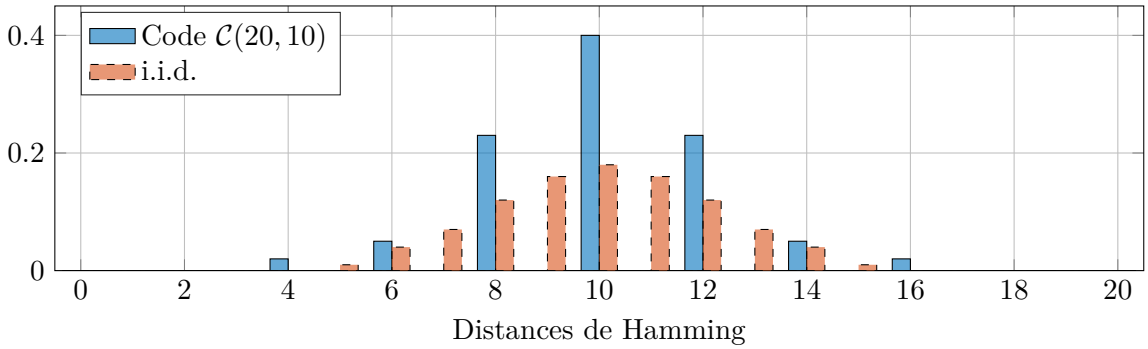


FIGURE B.18 – Distribution des distances en l'absence de bruit pour le code LDPC $\mathcal{C}_6(20, 10)$ et pour des mots i.i.d. de longueur 20

B.7 Code de Hamming : $\mathcal{C}_7(7,4)$

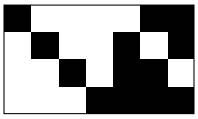


FIGURE B.19 – Matrice génératrice du code $\mathcal{C}_7$: $n_c = 7$ et $k_c = 4$

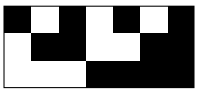


FIGURE B.20 – Matrice de parité du code $\mathcal{C}_7$: $n_c = 7$ et $k_c = 4$

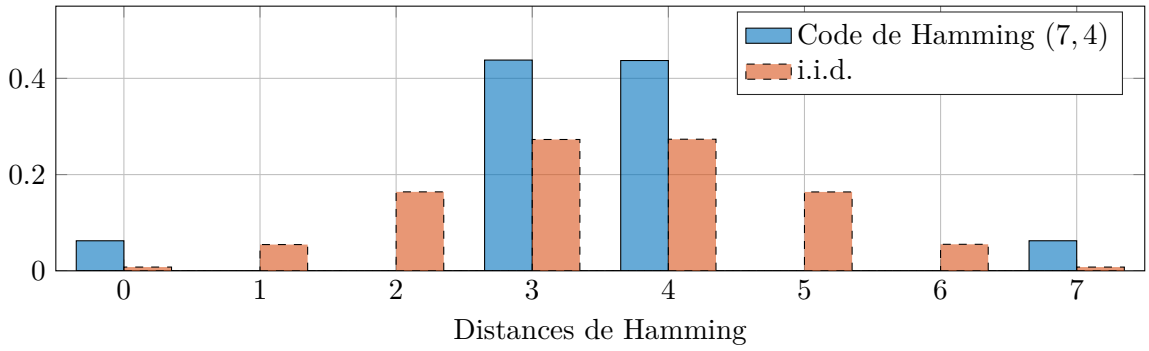


FIGURE B.21 – Distribution des distances en l'absence de bruit pour le code de Hamming $\mathcal{C}_7(7,4)$ et pour des mots i.i.d. de longueur 7

Bibliographie

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. *IEEE Transactions on Information Theory*, 20(2) :284–287, March 1974.

[2] J. Barbier. Reconstruction of Turbo-code encoders. In *Proceedings of SPIE*, volume 5819, pages 463–473, 2005.

[3] J. Barbier and J. Letessier. Forward Error Correcting Codes Characterization Based on Rank Properties. In *International Conference on Wireless Communications & Signal Processing (WCSP)*, November 2009.

[4] J. Barbier, G. Sicot, and S. Houcke. Algebraich Approach for the Reconstruction of Linear and Convolutional Error Correcting Codes. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 16, pages 66 –71, November 2006.

[5] M. Bellard and J.-P. Tillich. Detecting and reconstructing an unknown convolutional code by counting collisions. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2967–2971, Honolulu, Hawaii, USAB, June 2014.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding : Turbo-codes. In *IEEE International Conference on Communications (ICC)*, volume 2, pages 1064–1070, Geneva, Switzerland, May 1993.

[7] D. P. Bertsekas and J. N. Tsitsiklis. *Introduction to Probability*, chapter 7. MIT, 2000.

[8] A. Bonvard, S. Houcke, R. Gautier, and M. Marazin. Classification Based on Euclidean Distance Distribution for Blind Identification of Error Correcting Codes in Noncooperative Contexts. *IEEE Transactions on Signal Processing*, 66(10) :2572–2583, May 2018.

[9] A. Bonvard, S. Houcke, M. Marazin, and R. Gautier. Identification des paramètres d'un code correcteur par déficience du nombre de classes. In *GRETSI*, 2015.

[10] A. Bonvard, S. Houcke, M. Marazin, and R. Gautier. Order Statistics on Minimal Euclidean Distance for Blind Linear Block Code Identification. In *IEEE International Conference on Communications (ICC)*, pages 1–5, May 2018.

[11] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1) :68 – 79, 1960.

[12] P. L. Boyd and R. Clark Robertson. Recovery of unknown constraint length and generator polynomials for linear convolutional encoders. In *IEEE 21st Century Military Communications Conference*, volume 2, pages 947–951, Los Angeles, CA, USA, October 2000.

[13] P. L. Boyd and R. Clark Robertson. Recovery of unknown constraint length and generator polynomials for linear convolutional encoders in noise. In *IEEE 21st Century Military Communications Conference*, volume 2, pages 952–956, Los Angeles, CA, USA, October 2000.

[14] G. Burel and R. Gautier. Blind Estimation of Encoder and Interleaver Characteristics in a Non Cooperative Context. In *International Conference on Communication, Internet and Information Technology*. IASTED, November 2003.

[15] J. Cain, G. Clark, and J. Geist. Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding. *IEEE Transactions on Information Theory*, 25(1) :97–100, January 1979.

[16] K. Carrier and J.-P. Tillich. Identifying an unknown Code by Partial Gaussian Elimination. *Workshop on Coding and Cryptography*, 2017.

[17] C. Chabot. Recognition of a code in a noisy environment. In *IEEE International Symposium on Information Theory*, pages 2210–2215, 2007.

[18] W. Chen and G. Wu. Blind Recognition of $(n-1)/n$ Rate Punctured Convolutional Encoders in a Noisy Environment. *Journal of Communications*, 10(4) :260–267, April 2015.

[19] M. Cluzeau. Block code reconstruction using iterative decoding techniques. In *IEEE International Symposium on Information Theory*, 2006.

[20] M. Cluzeau. *Reconnaissance d'un schéma de codage*. PhD thesis, École Polytechnique, Novembre 2006.

[21] M. Cluzeau and M. Finiasz. Reconstruction of punctured convolutional codes. In *IEEE Information Theory Workshop*, pages 75–79, October 2009.

[22] M. Cluzeau and M. Finiasz. Recovering a Code's Length and Synchronization from a Noisy Intercepted Bitstream. In *IEEE International Symposium on Information Theory*, pages 2737–2741, 2009.

[23] M. Cluzeau, M. Finiasz, and J.-P. Tillich. Methods for the reconstruction of parallel turbo codes. In *IEEE International Symposium on Information Theory*, pages 2008–2012, June 2010.

[24] M. Cluzeau and J.-P. Tillich. On the Code Reverse Engineering Problem. In *IEEE International Symposium on Information Theory*, 2008.

[25] M. Côte. *Reconnaissance de codes correcteurs d'erreurs*. PhD thesis, École Polytechnique, Mars 2010.

[26] M. Côte and N. Sendrier. Reconstruction of convolutional codes from noisy observation. In *IEEE International Symposium on Information Theory*, pages 546–550, June 2009.

[27] M. Côte and N. Sendrier. Reconstruction of a turbo-code interleaver from noisy observation. In *IEEE International Symposium on Information Theory*, pages 2003–2007, June 2010.

[28] Y. G. Debessu, H. Wu, H. Jiang, and S. Y. Chang. Blind encoder parameter estimation for turbo codes. In *IEEE Global Communications Conference (GLOBECOM)*, pages 4233–4237, Décembre 2012.

[29] Y. G. Debessu and H. C. Wu. Novel blind encoder parameter estimation for Turbo Code. *IEEE Communications Letters*, 2012.

[30] Y. G. Debessu, H. C. Wu, and H. Jiang. Novel Blind Encoder Parameter Estimation for Turbo Codes. *Communication Letters*, 16(12) :1917–1920, Dec. 2012.

[31] J. Dingel and J. Hagenauer. Parameter Estimation of a Convolutional Encoder from Noisy Observations. In *IEEE International Symposium on Information Theory*, pages 1776–1780, June 2007.

[32] I. Dumer. On minimum distance decoding of linear codes. *5th Joint Soviet-Swedisch International Workshop on Information Theory*, 1991.

[33] E. Filiol. Reconstruction of Convolutional Encoders over GF(q). In *Cryptography and Codong : 6th IMA International Conference*, pages 101–109, July 1997.

[34] E. Filiol. *Décodage, Détection et Reconnaissance des Codes Linéaires Binaires.* PhD thesis, Université de Limoges, Octobre 1998.

[35] G. D. Forney. *Concatenated codes.* PhD thesis, Cambridge, Massachusetts : MIT, 1965.

[36] R. G. Gallager. *Low Density Parity Check Codes.* PhD thesis, Massachusetts Institute of Technology, 1963.

[37] L. Gan, D. Li, Z. Liu, and L. Li. A low complexity algorithm of blind estimation of convolutional interleaver parameters. *Science China Information Sciences*, 56(4) :1–9, April 2013.

[38] R. Gautier, G. Burel, M. Marazin, and C. Ñsiala Nzéza. Blind identification of block interleaver length and encoder parameters. *MTA Review*, XXI(1), March 2011.

[39] R. Gautier, M. Marazin, and G. Burel. Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs punctured. In *International Conference on Communication COMM*, pages 345–348, June 2008.

[40] R. Gautier, M. Marazin, and G. Burel. Dual code method for blind identification of convolutional encoder for cognitive radio receiver design. In *IEEE Globecom Workshops*, pages 1–6, Honolulu,USA, November 2009.

[41] A. J. Han Vinck, P. Dolezal, and Y.-G. Kim. Convolutional encoder state estimation. *IEEE Transactions on Information Theory*, 44(4) :1604–1608, July 1998.

[42] A. Hocquenghem. On a class of error correcting binary group codes. *Chiffres (Paris)*, 2 :147–156, September 1959.

[43] R. Imad, S. Houcke, and C. Douillard. Blind frame synchronization on Gaussian channel. In *15th European Signal Processing Conference*, pages 555–559, September 2007.

[44] R. Imad, G. Sicot, and S. Houcke. Blind Frame Synchronization for Error Correcting Codes having a Sparse Parity Chack Matrix. *IEEE Transaction on Communications*, 49(6) :1574–1577, 2009.

[45] J. Jeong, D. Yoon, J. Lee, and S. Choi. Blind reconstruction of a helical scan interleaver. In *IEEE 8th International Conference on Information, Communications Signal Processing*, pages 1–4, Singapour, December 2011.

[46] Y. Jia, L. Li, Y. Li, and L. Gan. Blind Estimation of Convolutional Interleaver Parameters. In *IEEE 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–4, Shanghai,China, September 2012.

[47] Y.-Q. Jia, L.-P. Li, Y.-Z. Li, and L. Gan. Blind Estimation of Communication Emitter Feature Parameters. In *IEEE 12th International Conference on Computer and Information Technology (CIT)*, pages 281–285, 10 2012.

[48] D. Jo, S. Kwon, and D. Shin. Blind Reconstruction of BCH Codes Based on Consecutive Roots of Generator Polynomials. *IEEE Communications Letters*, 22(5) :894–897, May 2018.

[49] H. Lee, C.-S. Park, J.-H. Lee, and Y. Song. Reconstruction of BCH codes using probability compensation. In *IEEE 18th Asia-pacific Conference Communications (APCC)*, pages 591–594, Jeju Island, October 2012.

[50] W. Lei, H. Yihua, H. Shiqi, and Q. Lin. The method of estimating the length of linear cyclic code based on the distribution of code weight. In *IEEE 2th International Conference on Information Science and Engineering*, pages 2459–2462, December 2010.

[51] J.-S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, pages 1354–1359, 1988.

[52] C. Li, T. Zhang, and Y. Liu. Blind Recognition of RS Codes Based on Galois Field Columns Gaussian Elimination. *IEEE International Congress on Image and Signal Processing*, 2014.

[53] D. Li and L. Gan. A Method of Parameters Estimation of SCCC Turbo Code. In *2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing*, pages 5–9, Décembre 2013.

[54] W. Li, J. Lei, L. Wen, and B. Chen. An Improved Method of Blind Recognition of RS Code Based on Matrix Transformation. *IEEE International Conference on Communication Technology*, 2013.

[55] P. Loidreau. Using algebraic structures to improve LDPC code reconstruction over a noisy channel. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2439–2443, July 2019.

[56] L. Lu, K. H. Li, and Y. L. Guan. Blind Detection of Interleaver Parameters for Non-Binary Coded Data Streams. In *IEEE International Conference on Communications (ICC)*, pages 1–4, June 2009.

[57] L. Lu, K. H. Li, and Y. L. Guan. Blind identification of convolutional interleaver parameters. In *7th International Conference on Information, Communications and Signal Processing (ICICS)*, pages 1–5, Dec 2009.

[58] P. Lu, S. Li, X. Luo, and Y. Zou. Blind recognition of punctured convolutional codes. *IEEE International Symposium on Information Theory*, 2004.

[59] P. Lu, S. Li, Y. Zou, and X. Luo. Blind recognition of punctured convolutional codes. *Science in China Series F : Information Sciences*, 48 :484–498, 08 2005.

[60] P. Lu and Y. Zou. Fast computations of gröbner bases and blind recognitions of convolutional codes. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields*, pages 303–317, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[61] X. Luo, P. Wang, P. Lu, and F. Liu. Fast and Blind Restoration Scheme for the Initial States of LFSRs. In *First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)*, volume 1, pages 192–198, June 2006.

[62] M. Marazin. *Reconnaissance en aveugle de codeur à base de code convolutif : contribution à la mise en oeuvre d'un récepteur intelligent.* PhD thesis, Université de Bretagne Occidentale, 2009.

[63] M. Marazin, R. Gautier, and G. Burel. Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs punctured. *MTA review*, XIX(2) :213–232, 2009.

[64] M. Marazin, R. Gautier, and G. Burel. Blind recovery of k/n rate convolutional encoders in a noisy environment. In *EURASIP Journal on Wireless Communications and Networking*, 2011.

[65] M. Marazin, R. Gautier, and G. Burel. Algebraich method for blind recovery of punctured convolutional encoders from an erroneous bitstream. *IET Signal Processing*, 6(2) :122–131, April 2012.

[66] M. Marazin, R. Gautier, and G. Burel. Some intresting dual code properties of convolutional encoders for standards self recognition. *IET Communications*, 6(8) :931–935, May 2012.

[67] F. Millioz and N. Martin. Estimation de la densité spectrale de puissance d'un bruit gaussien basée sur le kurtosis des statistiques minimales. In *GRETSI*, Septembre 2009.

[68] R. Moosavi and E. G. Larsson. Fast Blind Recognition of Channel Codes. *IEEE Transactions on Communications*, 62(5) :1393–1304, 2014.

[69] D. E. Muller. Application of Boolean algebra to switching circuit design and to error detection. *Transactions of the I.R.E. Professional Group on Electronic Computers*, EC-3(3) :6–12, September 1954.

[70] A. Naseri, O. Azmoon, and S. Fazeli. Blind Recognition Algorithm of Turbo Codes for Communication Intelligence Systems. *International Journal of Computer Science Issues*, 8(1) :68–72, 2011.

[71] L. Ouxin, G. Lu, and L. Hongshu. Blind Reconstruction of RS Code. *Asian Journal of Applied Sciences*, 8 :37–45, 01 2015.

[72] P. Elias. Coding for noisy channels. *IRE International Convention Record*, 4 :37–46, 1955.

[73] D. R. Pauluzzi and N. C. Beaulieu. A comparison of SNR estimation techniques for the AWGN channel. *IEEE Transactions on Communications*, 48(10) :1681–1691, October 2000.

[74] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions*, 1962.

[75] S. Ramabadran, A. S. Madhukumar, N. Wee Teck, and C. M. S. See. Parameter Estimation of Convolutional and Helical Interleavers in a Noisy Environment. *IEEE Access*, 5 :6151–6167, 2017.

[76] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4) :38–49, September 1954.

[77] I. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2) :300–304, 1960.

[78] B Rice. Determining the parameters of a rate $1/n$ convolutional encoder over GF(q). In *Third International Conference on Finite Fields and Applications*, Glasgow, 1995.

[79] H. Ryu, J. Lee, H. Hong, and D. Yoon. Estimation of interleaver period for unknown signals. In *IEEE 2nd International Conference on Network Infrastructure and Digital Content*, pages 678–680, September 2010.

[80] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3) :379–423, 7 1948.

[81] C. E. Shannon. Probability of Error for Optimal Codes in a Gaussian Channel. *The Bell System Technical Journal*, 38(3) :611–656, 1959.

[82] G. Sicot and S. Houcke. Blind detection of interleaver parameters. *ICASSP*, 2005.

[83] G. Sicot, S. Houcke, and J. Barbier. Blind detection of interleaver parameters. *ELSEVIER Signal Processing*, pages 450–462, 2008.

[84] G. Solomon and J. J. Stiffler. Punctured systematic cyclic coder. *IEEE International Convention Record*, 1 :128–129, 1964.

[85] G. Solomon and J. J. Stiffler. Algebraically punctured cyclic codes. *Information and Control*, 8 :1708–179, 1965.

[86] J. Stern. A method for finding codewords of small weight. *Coding theory and applications-Springer*, pages 106–113, 1989.

[87] S. Su, J. Zhou, Z. Huang, C. Liu, and Y. Zhang. Blind identification of convolutional encoder parameters. *The Scientific World Journal*, May 2014.

[88] F.-W. Sun and A. J. Vinck. An algorithm for identifying rate $(n-1)/n$ catastrophic punctured convolutional encoders. *IEEE Transactions on Information Theory*, 42(3) :1010–1013, May 1996.

[89] R. Swaminathan , A.S. Madhukumar, G. Wang, and T. Shang Kee. Blind recognition of LDPC code parameters over erroneous channel conditions. *IET Signal Processing*, 13 :86–95, February 2019.

[90] R. Swaminathan and A. S. Madhukumar. Classification of Error Correcting Codes and Estimation of Interleaver Parameters in a Noisy Transmission Environment. *IEEE Transactions on Broadcasting*, 63(3) :463–478, September 2017.

[91] R. Swaminathan, A. S. Madhukumar, G. Wang, and T. Shang Kee. Blind Reconstruction of Reed-Solomon Encoder and Interleavers Over Noisy Environment. *IEEE Transactions on Broadcasting*, 64(4) :830–845, December 2018.

[92] R. Swaminathan, A. S. Madhukumar, G. Wang, and S. K. Ting. Parameter Identification of Reed-Solomon Codes over Noisy Environment. In *IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, September 2017.

[93] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5) :533–547, September 1981.

[94] M. Teimouri and A. Hedayat. Parameter Estimation of Turbo Code Encoder. *Advances in Electrical Engineering*, 2014, 09 2014.

[95] J.-P. Tillich, A. Tixier, and N. Sendrier. Recovering the interleaver of an unknown turbocode. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2784–2788, 2014.

[96] A. Tixier. Blind identification of an unknown interleaved convolutional code. In *IEEE International Symposium on Information Theory*, pages 71–75, China, Hong-Kong, 2015.

[97] A. Tixier. *Reconnaissance de code LDPC : Recherche des équations de parité de petits poids*. PhD thesis, INRIA, 2015.

[98] A. Valembois. *Décodage, Détection et Reconnaissance des Codes Linéaires Binaires*. PhD thesis, Université de Limoges, Octobre 2000.

[99] A. Valembois. Detection and recognition of a binary linear code. *ELSEVIER Discrete Applied Mathematics 111*, pages 199–218, 2001.

[100] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2) :260–269, April 1967.

[101] F. Wang, Z. Huang, and Y. Zhou. A method for blind recognition of convolution code based on Euclidean Algorithm. *IEEE International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1414–1417, 2007.

[102] J. Wang, Y. Yue, and J. Yao. A Method of Blind Recognition of Cyclic Code Generator Polynomial. In *IEEE 6th International Conference on Wireless Communications Networking and Mobile Computing*, September 2010.

[103] J. Wang, Y. Yue, and J. Yao. Statistical Recognition Method of Binary BCH Code. *Communications and Network*, 3(1) :17–22, March 2011.

[104] L. Wang and D. Li. A New Method for BCH Codes of Blind Recognition. *Advanced Materials Research*, 631-632 :1403–1408, January 2013.

[105] Y. Wang, F.-H. Wang, and Z.-T. Huang. Blind Recognition of (n, k, m) Convolutional Code Based on Local Decision in a Noisy Environment. In *International Conference on Automation, Mechanical Control and Computational Engineering*, January 2015.

[106] L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes. *Patent*, 4 633 470, December 1986.

[107] Z. Wu, L. Zhang, Z. Zhong, and R. Liu. Blind Recognition of LDPC Codes over Candidate Set. *IEEE Communications Letters*, pages 1–1, November 2019.

[108] T. Xia and H.-C. Wu. Blind Identification of Nonbinary LDPC Codes Using Average LLR of Syndrome a Posteriori Probability. *IEEE Communications Letters*, 17(7) :1301–1304, July 2013.

[109] T. Xia and H.-C. Wu. Blind identification of binary LDPC codes for M-QAM signals. In *IEEE Global Communications Conference (GLOBECOM)*, pages 3532–3536, 2014.

[110] T. Xia and H.-C. Wu. Joint Blind Frame Synchronization and Encoder Identification for LDPC Codes. In *IEEE Wireless Communications Symposium*, pages 5221–5226, 2014.

[111] T. Xia and H.-C. Wu. Novel Blind Identification of LDPC Codes Using Average LLR of Syndrome *a Posteriori Probability*. *IEEE Transaction on Signal Processing*, 62(3) :1393–1304, February 2014.

[112] H. Xie, X. Chai, F. Wang, and Z. Huang. A method for blind identification of rate 1/2 convolutional code based on improved Euclidean algorithm. In *IEEE 11th International Conference on Signal Processing*, volume 2, pages 1307–1310, October 2012.

[113] X. Yang and N. Wen. Recognition Method of BCH Codes Based on Roots Information Dispersion Entropy and Roots Statistic. *Journal of Detection and Control*, pages 69–73, 2010.

[114] A. D. Yardi, S. Vijayakumaran, and A. Kumar. Blind Reconstruction of Binary Cyclic Codes. In *20th European Wireless Conference*, May 2014.

[115] A. D. Yardi, S. Vijayakumaran, and A. Kumar. Blind Reconstruction of Binary Cyclic Codes From Unsynchronized Bitstream. *IEEE Transactions on Communications*, 64(7) :2693–2706, July 2016.

[116] P. Yu, J. Li, and H. Peng. A Least Square Method for Parameter Estimation of RSC Sub-Codes of Turbo Codes. *IEEE Communications Letters*, 18(4) :644–647, April 2014.

[117] P. Yu, J. Li, and H. Peng. Gibbs sampling based parameter estimation for RSC sub-codes of turbo codes. In *2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–5, October 2014.

[118] H. Zhang, H.-C. Wu, and H. Jiang. Novel Blind Encoder Identification of Reed-Solomon Codes with Low Computational Complexity. *IEEE Globecom*, 2013.

[119] J. Zhou, Z. Huang, C. Liu, S. Su, and Y. Zhang. Information-Dispersion-Entropy-Based Blind Recognition of Binary BCH Codes in Soft Decision Situations. *Entropy*, 15 :1705–1725, 2013.

[120] J. Zhou, Z. Huang, S. Su, and S. Yang. Blind recognition of binary cyclic codes. *EURASIP Journal on Wireless Communications and Nteworking*, 2013.

[121] J. Zhou, Z. Huang, S. Su, and Y. Zhang. Blind identification of convolutional codes in soft-decision situations. *International Journal of Modern Communication Technologies & Research (IJMCTR)*, 2(4) :4–6, April 2014.

[122] Y. Zrelli. *Identification aveugle de codes correcteurs d'erreurs basés sur des grands corps de Galois et recherche d'algorithme de type décision souple pour les codes convolutifs*. PhD thesis, Université de Bretagne Occidentale, 2013.

[123] Y. Zrelli, R. Gautier, M. Marazin, E. Rannou, and E. Radoi. Focus on theoretical properties of blind convolutional codes identification methods based on rank criterion. *MTA Review*, XXII(4) :213–234, Décembre 2012.

[124] Y. Zrelli, R. Gautier, E. Rannou, M. Marazin, and E. Radoi. Blind identification of code word length for non-binary error correcting codes in noisy transmission. *EURASIP Journal on Wireless Communications and Networking, Special Issue on Ten Years of Cognitive Radio : State of the Art and Perspectives*, 2015(43), March 2015.

[125] Y. Zrelli, M. Marazin, E. Rannou, and R. Gautier. Blind Identification of Convolutional Encoder Parameters over GF(2^m) in the Noiseless Case. In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–5, July 2011.

Listes des publications

Conférences à comité de lecture

- A. Bonvard, S. Houcke, M. Marazin et R. Gautier. "Identification des paramètres d'un code correcteur par déficience du nombre de classes". *GRETSI 2015 : 25-ième colloque du Groupement de Recherche en Traitement du Signal et des Images*, France, Lyon, 2015, pages 1-5.
- A. Bonvard, S. Houcke, M. Marazin and R. Gautier. "Order Statistics on Minimal Euclidean Distance for Blind Linear Block Code Identification". *In IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018, pages 1-5.

Revue à comité de lecture

- A. Bonvard, S. Houcke, R. Gautier and M. Marazin. "Classification Based on Euclidean Distance Distribution for Blind Identification of Error Correcting Codes in Noncooperative Contexts". *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2572-2583, May, 2018.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Algorithmes de détection et de reconstruction en aveugle de code correcteur d'erreurs basés sur des informations souples

Mot clés : Internet des Objets - Radio intelligente - Codes linéaire - Identification aveugle - Reconstruction de codes

Résumé : Les dernières décennies ont connu l'essor des communications numériques. Ceci a provoqué une prolifération des standards de communication, ce qui demande une plus grande adaptabilité des systèmes de communication. Une manière de rendre ces systèmes plus flexibles consiste à concevoir un récepteur intelligent qui serait capable de retrouver l'ensemble des paramètres de l'émetteur à partir du signal reçu. Dans ce manuscrit, nous nous intéressons à l'identification en aveugle des codes correcteurs d'erreurs. Nous proposons des méthodes originales basées sur le calcul de distances euclidiennes entre des séquences de symboles bruités.

Tout d'abord, un premier algorithme de classification permet la détection d'un code puis l'identification de la longueur de ses mots de code. Un second algorithme basé sur le nombre de collisions permet quand à lui d'identifier la longueur des mots d'informations. Ensuite, nous proposons une autre méthode utilisant cette fois les distances euclidiennes minimales pour l'identification de la longueur d'un code en bloc. Enfin, une méthode de reconstruction du code dual d'un code correcteur d'erreurs est présentée.

Title: Algorithms for blind detection and reconstruction of error-correcting code from soft information

Keywords: Internet of Things - Cognitive radio - Linear codes - Blind identification - Code reconstruction

Abstract: Recent decades have seen the rise of digital communications. This has led to a proliferation of communication standards, requiring greater adaptability of communication systems. One way to make these systems more flexible is to design an intelligent receiver that would be able to retrieve all the parameters of the transmitter from the received signal. In this manuscript, we are interested in the blind identification of error-correcting codes. We propose original methods based on the calculation of Euclidean distances between noisy symbol sequences.

First, a classification algorithm allows the detection of a code and then the identification of its code words length. A second algorithm based on the number of collisions allows to identify the length of the information words. Then, we propose another method using the minimum Euclidean distances to identify block codes length. Finally, a method for reconstructing the dual code of an error-correcting code is presented.