Codes correcteurs quantiques pouvant se décoder itérativement

Denise Maurice

THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Denise MAURICE

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Codes correcteurs quantiques pouvant se décoder itérativement

soutenue le 26 juin 2014

devant le jury composé de :

M. Jean-Pierre Tillich	Directeur de thèse
M. Gilles Zémor	Rapporteur
M. David Declercq	Rapporteur
M. Jean-Claude Bajard	Examinateur
M. Damian Markham	Examinateur
M. Daniel Augot	Examinateur
Mme. Iryna Andriyanova	Examinatrice

Thèse réalisée à **Inria**, dans l'équipe **SECRET**

Remerciements

Me voici devant la fameuse page, blanche pour le moment[1]*, des remerciements. Quelque part, je suis heureuse d'en arriver là, puisque ça signifie que j'ai (presque) terminé, et que le plus dur est fait. Mais ce qui reste à écrire ne se présente pas comme des équations, beaucoup plus simples à manipuler que des émotions en fin de compte.*

Pour faire une analogie avec un univers que j'aime bien, j'ai l'impression d'être à la fin d'une longue représentation (et pas la moindre !), et de remonter sur scène pour une révérence finale. La comparaison se tient plutôt bien : un salut, c'est ce dernier moment d'un spectacle, où l'artiste redevient humain. C'est cet instant pourtant dérisoire face à la prestation effectuée plus tôt, et pourtant si important.

Or, tout comme un spectacle ne peut avoir lieu sans une batterie de gens, qu'on ne voit pas forcément sur scène ; cette thèse –que je peux fièrement appeler « ma » thèse–, n'aurait jamais eu lieu sans le concours d'un certain nombre de personnes, que je voudrais remercier (par un tonnerre d'applaudissements silencieux).

En tout premier, le maître de cérémonie, c'est-à-dire dit le directeur de thèse. Non seulement très fort dans son domaine, mais également bon vivant, cruxiverbiste acharné, grand amateur de desserts (surtout ceux des autres), champion mondial (presque) de subtilisation discrète (ou pas) de ces derniers, parieur ayant le goût du risque (et du balai), râleur intempestif (surtout contre les gens qui écrivent des mots faux dans les mots croisés, cf un des descriptifs précédents), et j'en passe. Il a su me conseiller, m'orienter, me supporter (argh !), me secouer quand il fallait. Et même s'il était un peu rude parfois, ce n'était jamais sans raison (là, je regarde au plafond avec un air le plus innocent possible) ; il a toujours été juste et honnête envers moi.

On dit parfois que le directeur de thèse est à la fois un peu comme un père, et comme un compagnon. Or, un hasard amusant de l'ordre alphabétique a placé son

1. En fait il s'agit d'un éditeur de texte, mais ça sonne nettement moins bien.

prénom dans mon répertoire téléphonique tout pile entre les prénoms de... mon père et mon compagnon. Promis, ce n'était pas voulu. J'ai eu longtemps la crainte de me tromper d'un cran et de l'appeler sans le faire exprès !

Bref, mon directeur de thèse, il est trop fort. S'il n'existait pas, il faudrait l'inventer (et ça serait un sacré boulot). Merci pour tout ça, et pour tout ce que j'aurais pu oublier.

Prenant part également au spectacle, merci au jury ; qui a accepté de se déplacer pour venir évaluer le fruit de mon travail (et pas seulement en perspective d'un bon pot !). Je ne connais que certains d'entre vous, aussi il m'est difficile de vous adresser quelque chose de plus personnalisé. Mais merci d'être là pour le grand final. Une révérence spéciale pour les rapporteurs, qui ont accepté de relire cette longue thèse. Et pardon pour le fil à retordre qu'elle a pu éventuellement vous donner.

Parmi les autres artistes qui ont participé à cette aventure, il y a notamment les membres du groupe de travail COCQ, dans lequel nous avons tous ensemble discuté, réfléchi, trouvé des choses, échoué, réfléchi encore. Ces réunions sont toujours aussi intéressantes et enrichissantes, et ce, grâce à ceux qui y participent : merci.

À la régie, il y a le projet SECRET. Des canapés confortables, quelques desserts qui n'attendent que d'être mangés, des gâteaux ramenés d'un pays lointain (ou pas), un tas de permanents, doctorants, stagiaires et compagnie installés à discuter en buvant du café ou du thé tout en balançant gentiment des blagues sur celui qui allait devenir mon directeur de thèse[2]. Pas de doute, le premier jour où j'ai mis les pieds en cet endroit, j'ai su que je m'y sentirai bien. Malgré le renouvellement constant (ou peut-être est-ce grâce à ce renouvellement, qui sait ?), ceux qu'on y trouve sont toujours aussi adorables que différents, et leur bonne humeur est terriblement contagieuse. Bref, c'est un vrai plaisir d'y être et d'y travailler. Merci à vous tous pour ces bons moment, ce soutien, ces discussions scientifiques ou non, ces parties de mots croisés sportifs[3]. Vous allez tous me manquer.

Un salut aussi pour mon laboratoire d'accueil, ETIS, pendant cette année d'ATER. Un an, c'est court, surtout quand on a une thèse à finir et une agrégation à préparer, et ce n'est pas toujours évident de nouer des liens ou de s'intégrer. Et pourtant ! Merci pour cet accueil, ces discussions variées, et ces blagues à la pause café/thé.

2. Non, je ne donnerai pas de noms.

3. Si vous ne savez pas comment des mots croisés peuvent être sportifs, vous êtes des lecteurs normalement constitués, sains de corps et d'esprit, n'étant jamais venus faire des mots croisés au projet.

Comme dans tout spectacle, il faut aussi remercier les « petites mains » qu'on ne voit pas sur scène, mais pourtant indispensables. Sans eux, que serais-je devenue ? À tous les copains, merci pour (dans le désordre, mais vous vous retrouverez) les égarements dans les grandes voies, les acrobaties plus ou moins stables, les baffes involontaires à coup de hoop, les blagues potaches en cours de prépa agreg, les passings tordus, les papotages sur les tapis d'acro, les danses partagées (qu'il s'agisse de danses de couple ou de grande chorégraphies), les fous rires incontrôlables, les spectacles présentés, les spectacles regardés, les bons petits plats subtils (ou décadents), les chansons à casser les oreilles des voisins, les séances de jeu de rôle épiquement drôles.

Un petit bonus spécial à tous les « grammar nazis » et sauveurs LaTeX pour le petit (gros) coup de pouce technique pour la rédaction. Je vous dois des chocolats.

Enfin, merci à mes parents, pour l'amour qu'ils m'ont apporté et la curiosité qu'ils m'ont toujours encouragée à cultiver. À mes frères et sœurs, et tout ce que nous avons partagé. Et à mon compagnon depuis maintenant plus de six ans, qui m'a accompagnée, supportée, soutenue tout le long de cette thèse. Pour vous, je suis complètement à court, parce que j'ai trop de choses à dire, mais une boule dans la gorge m'en empêche (aussi paradoxal que ça puisse paraître lorsqu'il s'agit de taper sur un clavier). Alors simplement, merci.

Table des matières

Introduction

Comment lire ce document ?

Les thématiques liées aux codes correcteurs quantiques peuvent intéresser des lecteurs issus du domaine de l'informatique quantique, d'une part, ou des codes correcteurs classiques, d'autre part ; mais pas nécessairement familiers avec *les deux* domaines réunis. Ainsi, ce document présente les bases de l'informatique quantique (du moins le nécessaire pour comprendre les codes stabilisateurs), ainsi que l'essentiel sur les codes correcteurs classiques, notamment les codes LDPC (Low Density Parity Check).

La première conséquence est qu'il peut être lu par des lecteurs n'étant familiers ni avec l'un, ni avec l'autre de ces concepts : des notions fondamentales d'algèbre linéaire et un peu de probabilités sont requises, mais guère plus.

La seconde conséquence est qu'il n'est pas nécessaire de le lire de façon linéaire. Les deux premiers chapitres (présentation des codes classiques d'une part, introduction au quantique d'autre part), entre autres, sont indépendants. Le lecteur peut donc, s'il le souhaite, lire le chapitre 2 avant le chapitre 1 ; voire même, s'il est déjà familier avec l'une ou l'autre des notions présentées, ne pas lire l'une ou l'autre de ces chapitres [4]. De plus, si les chapitres 1 à 4 fournissent les bases des codes LDPC quantiques, les chapitres 5 à 7 présentent différentes constructions particulières étudiées dans cette thèse : à quelques sections près, l'ordre des chapitres est arbitraire (il est, de fait, chronologique).

La figure 1 donne le schéma des dépendances des différentes parties. Le lecteur est donc libre de suivre l'ordre de lecture qu'il souhaite, si celui imposé par la succession des pages ne lui convient pas.

Bonne lecture.

4. Voire encore lire $\frac{1}{\sqrt{2}}|\text{chapitre 1}\rangle + \frac{1}{\sqrt{2}}|\text{chapitre 2}\rangle$. Cela dit, le lecteur ayant compris cette petite remarque est déjà familier avec le quantique, et n'a, de ce fait, pas forcément besoin de lire le chapitre 2.

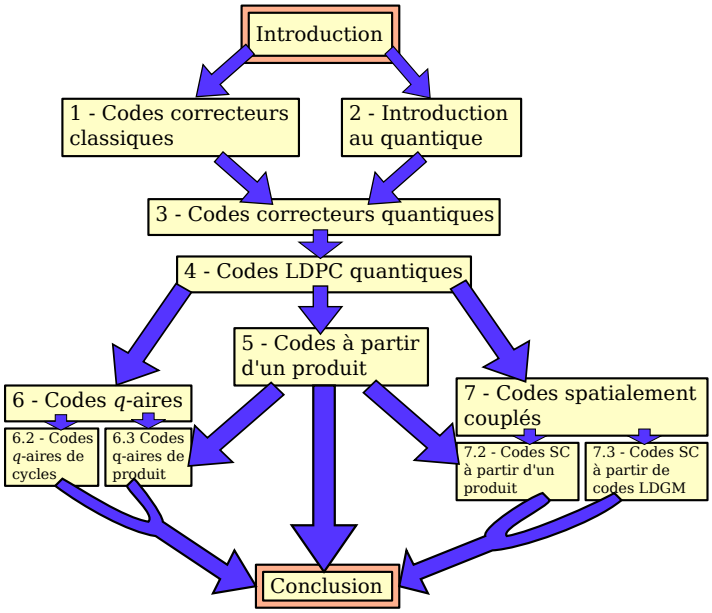


FIGURE 1 – Plan des dépendances de ce document : une flèche entre un chapitre A et un chapitre B indique qu'il est nécessaire d'avoir lu (ou de connaître les notions associées) le chapitre A pour comprendre le B. Sinon, une lecture linéaire fonctionne également.

Pourquoi des codes correcteurs quantiques ?

L'informatique quantique est un domaine récent (les premières expérimentations et théories datent des années 80) et en plein développement. Les applications possibles sont alléchantes : ordinateurs plus puissants, communications secrètes plus sûres, etc. Le mot « quantique » est, de plus, très vendeur auprès du tout public, et sert de prétexte à beaucoup de médias de science-fiction. Mais quel est le principe, au fond ?

L'idée de base est de considérer qu'un objet n'existe pas nécessairement dans un état donné, mais peut être une *superposition d'états différents*, et ce, jusqu'à ce qu'on décide d'observer l'objet en question : à cet instant, on modifie l'objet pour ne le voir que dans un des états possibles. Quand on parle de mécanique quantique, on parle souvent du *chat de Schrödinger*. Il s'agit d'une expérience de pensée dans laquelle on met un chat dans une boîte hermétique, et dans cette boîte, un système ayant exactement une chance sur deux de répandre un poison et de tuer l'animal[5]. Les lois de la mécanique quantique soutiennent alors que, tant que la boîte n'a pas

5. C'est une expérience de pensée uniquement. L'auteur tient à préciser qu'aucun chat n'a été maltraité durant la rédaction de cette thèse ou pendant les travaux liés à celle-ci.

été ouverte, le chat n'est pas dans un état fixe « vivant » ou « mort », mais dans une *superposition* des deux. Ce n'est qu'à l'ouverture de la boîte, en observant le chat, que son état mort ou vivant est fixé (voir figure 2).

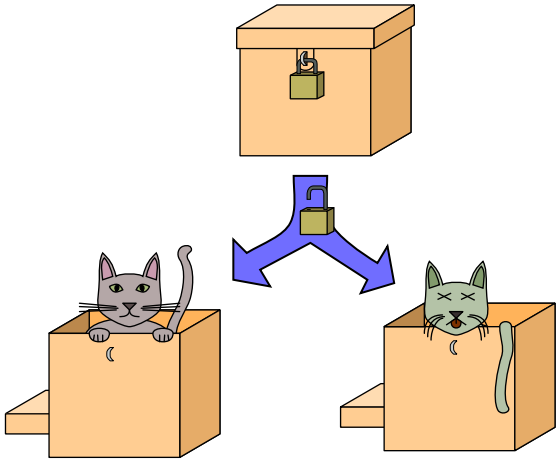


FIGURE 2 – Le célèbre chat de Schrödinger, dans sa boîte. Lorsqu'on l'ouvre, on constate que le chat est mort ou que le chat est vivant, mais tant que la boîte reste fermée, il est dans les deux états à la fois.

Dans la pratique, l'information quantique n'est pas portée par des animaux (les phénomènes quantiques ne se manifestant pas à échelle humaine — ou animale —), mais par des photons, ou des ions. La manipulation de ces objets est extrêmement délicate, et l'informatique quantique est encore très expérimentale. Pourquoi alors s'intéresser aux codes correcteurs quantiques ?

Reprenons notre chat en boîte. Il contient une superposition de deux états : un chat vivant, et un chat mort. Que se passe-t-il si la boîte n'est pas hermétiquement fermée ? On peut éventuellement entendre des bruits venant de la boîte (voir figure 3), ce qui revient à observer le chat sans le faire exprès, et donc à modifier son état : si on entend un miaulement, c'est qu'on a, sans le vouloir, modifié l'état en chat vivant. Plus généralement, un système physique non parfaitement isolé peut subir des mesures involontaires, dont on ignore le résultat. Or ces mesures involontaires induisent évidemment des modifications sur le système quantique. Ainsi, dans un hypothétique ordinateur quantique, si rien n'est fait pour se protéger de ce phénomène, des modifications involontaires vont apparaître sur le système, jusqu'à rendre tout le calcul faux.

C'est là qu'interviennent des codes correcteurs, pour corriger ces erreurs à chaque instant et maintenir les états quantiques comme on le souhaite, que ce soit lors du calcul (on parle de *calcul tolérant aux fautes*), ou du stockage des états quantiques (on parle alors de *mémoires quantiques*). Ces codes correcteurs doivent

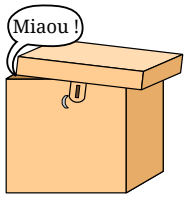


FIGURE 3 – La boîte contenant le chat de Schrödinger mal fermée : on entend du bruit venant de la boîte, ce qui revient à mesurer le chat involontairement.

non seulement corriger correctement, cela va de soi, mais ils doivent aussi le faire rapidement : si les erreurs surviennent plus vite que l'on ne peut les corriger, alors le nombre d'erreurs ne fera que s'accroître. Ainsi, des codes correcteurs corrigeant *vite* et *bien* sont indispensables à la mise en place de solutions à base de quantique (on pourra voir [Got13] par exemple, pour le lien entre calcul tolérant aux fautes et codes).

Même s'ils ne sont pas toujours utilisés dans le même but, les codes correcteurs quantiques ont un certain nombre de points communs avec leurs cousins du monde classique. Parmi ces derniers, les codes LDPC (voir [Gal62] entre autres), « Low Density Parity Check », autrement dit à matrice de parité creuse, offrent à la fois de très bonnes performances de décodage et un algorithme rapide pour y parvenir : un équivalent quantique de ces codes a donc de bonnes chances de donner les propriétés souhaitées.

Cependant, le monde quantique n'est pas aussi simple que le monde classique, et si dans le premier cas on arrive facilement à générer de très bons codes LDPC approchant la *capacité* des canaux (qui représente, informellement, la quantité maximale d'information qu'on peut espérer transmettre), dans le cas quantique ces constructions sont beaucoup moins efficaces. Il existe une notion équivalente de capacité de canal, mais celle-ci n'est pas connue avec précision dans le cas du canal dépolarisant, qui est un canal pourtant très simple. Pire encore, les codes qu'on peut construire sont encore loin d'atteindre la *borne de hachage*, qui n'est pourtant qu'une borne inférieure de la capacité. Une des raisons est que beaucoup de ces codes souffrent d'une distance minimale faible, voire constante. Dans ce document, plusieurs constructions sont expérimentées, avec plus ou moins de succès : dans certains cas, les performances sont décevantes (voir chapitre 5), dans d'autres, les résultats sont beaucoup plus encourageants (voir chapitres 6 et 7). Notamment, dans le dernier cas, les courbes de décodage sont de loin les meilleures connues pour un rendement ciblé (en l'occurence 1/4), et sont proches de la borne en question.

Chapitre 1

Codes correcteurs LDPC classiques

Ce chapitre a pour but de présenter rapidement les codes correcteurs classiques et particulièremet les codes LDPC. Le lecteur déjà familier avec ces notions peut donc passer directement au chapitre suivant, ou éventuellement à la section 1.3 pour les codes LDPC spécifiquement. Il n'a, bien sûr, pas vocation à être complet sur le sujet, et ne présente que ce qui est nécessaire pour la suite ; le lecteur intéressé pourra lire par exemple [RU08].

Tout l'intérêt des codes correcteurs est de transmettre un message à travers un environnement bruité, et surtout à faire en sorte que le destinataire reçoive le message correctement. L'alphabet phonétique de l'OTAN est un exemple de code correcteur. Les signaux radio sont en général bruités, et si on peut raisonnablement transmettre une phrase (le fait qu'il s'agisse d'une langue connue aide à reconstituer le message), lorsqu'on souhaite transmettre une lettre seule (par exemple, une immatriculation), on peut facilement se tromper. Ainsi, on utilise dans ce cas des mots pour désigner des lettres : « Alpha » pour A, « Bravo » pour B, « Charlie » pour C, etc. Si « D » et « T » se ressemblent beaucoup à l'oral, « Delta » et « Tango » beaucoup moins. Ainsi, le message transmis est plus long (un mot pour une lettre), mais plus sûr.

De façon plus générale, un code correcteur ajoute de la *redondance* dans le message envoyé, afin de compenser le bruit de la transmission.

1.1 Canaux et codes

Si un message peut prendre de multiples formes, on ne s'intéresse ici qu'à des messages formés d'une suite de caractères d'un alphabet fini $\mathcal{A}$. On appelle

symboles les éléments de cet alphabet, et *mots* les suites ainsi formées. Le plus souvent, cet alphabet sera $\{0,1\}$ (les symboles sont 0 et 1), et les messages sont alors des suites de bits ; par exemple 01101 est un mot de longueur 5.

1.1.1 Canal de communication

Lorsqu'on transmet un message, on dira qu'il emprunte un *canal de communication* (ou plus brièvement *canal*). Ce canal modélise les aléas —en général inévitables— du système physique de transmission : obstacles, interférences radio, appareils physiques imparfaits, pigeon voyageur qui se perd en route, etc. Il est représenté par des probabilités $p(y \mid x)$: la probabilité qu'on reçoive le message y en supposant qu'on a envoyé le message x. On parle de *canal sans mémoire* lorsque le canal agit *indépendamment* sur chaque symbole envoyé. Un tel canal est alors défini par des $p(y \mid x)$: la probabilité qu'on reçoive le *symbole* y, sachant qu'on a envoyé le *symbole* x.

Définition 1 (Canal sans mémoire)**.** *Soient deux alphabets $\mathcal{A}$ et $\mathcal{B}$. Un* canal sans mémoire *de l'alphabet $\mathcal{A}$ vers l'alphabet $\mathcal{B}$ est la donnée des probabilités suivantes :*

$$\forall x \in \mathcal{A}, \forall y \in \mathcal{B}, \quad p(y \mid x) = \textit{probabilité qu'on reçoive } y \textit{ en ayant envoyé } x$$

Pour un mot de longueur n fixée, la probabilité qu'on reçoive un mot $y = y_1 y_2 \dots y_n$ sachant qu'on a envoyé le mot $x = x_1 x_2 \dots x_n$ est alors :

$$p(y \mid x) = \prod_{i=1}^{n} p(y_i \mid x_i)$$

puisque le canal agit indépendamment sur chaque symbole.

Dans toute la suite, on ne parlera que de canaux sans mémoire. Ces canaux ont l'avantage d'être décrits très rapidement, puisqu'il suffit de donner les probabilités pour chaque couple de symboles. Par exemple si $\mathcal{A} = \mathcal{B} = \{0,1\}$, alors il n'y a que 4 probabilités à connaître. On donne les deux exemples des canaux les plus connus :

Définition 2 (Canal binaire symétrique)**.** *Le* canal binaire symétrique *de probabilité d'erreur p (ou de paramètre p) est le canal de $\{0,1\}$ vers $\{0,1\}$ décrit par les probabilités suivantes :*

$$\begin{aligned} p(0 \mid 0) = p(1 \mid 1) &= 1-p \\ p(1 \mid 0) = p(0 \mid 1) &= p \end{aligned}$$

Autrement dit c'est le canal qui change un bit avec probabilité p, et ne le change pas avec probabilité $1-p$.

Définition 3 (Canal binaire à effacements)**.** *Le* canal binaire à effacements *de probabilité p (ou paramètre p) est le canal de $\{0,1\}$ vers $\{0,1,e\}$, où e est appelé* symbole d'effacement*, décrit par les probabilités suivantes :*

$$\begin{aligned} p(e \mid 0) = p(e \mid 1) &= p \\ p(1 \mid 1) = p(0 \mid 0) &= 1-p \\ p(0 \mid 1) = p(1 \mid 0) &= 0 \end{aligned}$$

Ce canal « perd » un symbole (le remplaçant par e) avec une probabilité p, et le laisse inchangé sinon.

1.1.2 Code correcteur

On peut imaginer intuitivement des moyens pour contrebalancer les erreurs causées par ces canaux. Par exemple, pour le canal à effacements, on peut choisir de doubler chaque bit d'un message donné : pour envoyer 0, on envoie 00, et pour envoyer 1, on envoie 11. Si on reçoit le message $0e$, on sait que le message envoyé était 00. Par contre, si on reçoit ee, on ne peut pas retrouver le message. Pour le canal binaire symétrique, on peut tripler les bits du message : si par exemple on reçoit 101, on peut deviner que le message envoyé était 111. On peut raffiner, selon les besoins, en mettant plus ou moins de redondance : sur un canal à effacements avec une probabilité d'effacements plutôt faible, un *bit de parité* en fin de mot peut suffire. Par exemple pour transmettre le mot 10011, on transmet 100111 (le dernier bit étant la somme binaire des autres). Si un effacement survient sur un seul des bits du mot, on peut retrouver le message d'origine : par exemple si on reçoit $10e111$, on sait que le bit manquant est 0.

Définition 4 (Code)**.** *On appelle* code correcteur *(ou* code*) de longueur n et de dimension k ($k \leqslant n$) sur un alphabet $\mathcal{A}$ tout sous-ensemble $\mathscr{C}$ de $\mathcal{A}^n$ muni d'une fonction bijective, dite* fonction d'encodage *de $\mathcal{A}^k$ dans $\mathscr{C}$. On appelle $\mathscr{C}$ l'*ensemble des mots de code.

On confondra généralement l'ensemble des mots de code et le code lui-même.

Un code consiste donc à affecter à chaque mot de longueur k un autre mot, plus long (de longueur n). Le mot reçu ne sera pas forcément le mot de code envoyé, mais on espère qu'il en est « suffisamment proche » en termes de vraisemblance ; le *décodage* consiste à retrouver le mot de code en question à partir du mot bruité reçu.

Définition 5 (Rendement)**.** *Soit $\mathscr{C}$ un code de longueur n et de dimension k, on appelle* rendement *du code, et on note R la grandeur suivante :*

$$R = \frac{k}{n}$$

Le rendement représente la fraction des symboles comportant de l'information : un rendement de 1/2, par exemple, signifie qu'il faut doubler la longueur du message transmis (par rapport au cas où on envoie le message brut). Un rendement proche de 1 signifie qu'il y a peu de redondance, autrement dit la longueur est peu augmentée (car k est proche de n), tandis qu'un rendement proche de 0 signifie qu'il y a beaucoup de redondance et peu d'information, c'est-à-dire qu'il est nécessaire d'avoir une grande longueur pour la même information transmise (k est nettement plus petit que n).

Définition 6 (Décodage au maximum de vraisemblance)**.** *Soit un code $\mathscr{C}$ et un mot y reçu sur un canal donné. Le décodage au maximum de vraisemblance consiste à trouver le mot $x \in \mathscr{C}$ qui maximise :*

$$\max_{z \in \mathscr{C}} \mathbb{P}[z \text{ est le mot envoyé} \mid y \text{ est le mot reçu}]$$

On peut aussi considérer la proximité des mots en terme de distance de Hamming :

Définition 7 (Distance de Hamming)**.** *Soient deux mots $x = x_1 \dots x_n$ et $y = y_1 \dots y_n$. La distance de Hamming entre ces deux mots est définie par :*

$$d(x, y) = \#\{i, x_i \neq y_i\}$$

$\#\mathcal{A}$ désigne le cardinal d'un ensemble fini $\mathcal{A}$.

Cette distance permet de donner un paramètre fondamental du code, la distance minimale :

Définition 8 (Distance minimale d'un code)**.** *Soit $\mathscr{C}$ un code. La distance minimale d de ce code est donnée par :*

$$d = \min\{d(x, y), x, y \in \mathscr{C}, x \neq y\}$$

Autrement dit, la distance minimale d'un code est la plus courte distance qui sépare deux mots de code. Ainsi, plus la distance minimale d'un code est grande, moins il y a de chances de se tromper au décodage, car il faut plus d'erreurs pour confondre deux mots. Par contre, si cette distance est petite, il y a plus de chances qu'un mot soit altéré au point d'être plus proche d'un autre mot de code, et le décodage échoue plus facilement.

Plus précisément, si un mot reçoit moins de $(d-1)/2$ erreurs (il y a moins de $(d-1)/2$ symboles différents du mot envoyé), alors le décodage peut retrouver le mot d'origine ; au delà, un autre mot de code est plus proche, et le décodage échoue (voir figure 1.1).

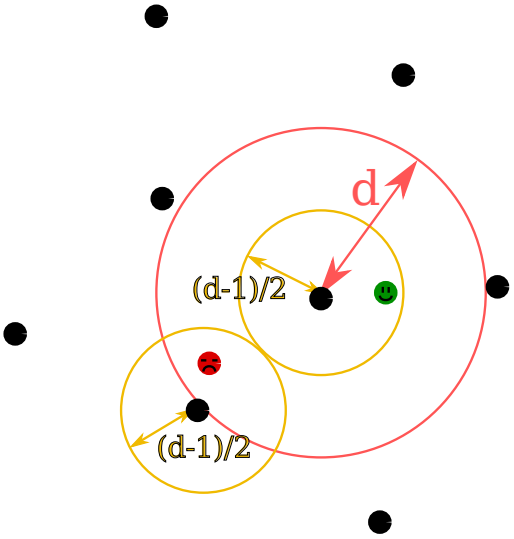


FIGURE 1.1 – Schéma représentant l'espace des mots de code (points noirs), et la distance minimle (d). Si le mot reçu est proche du mot d'origine (point vert), c'est-à-dire à une distance inférieure à $(d-1)/2$, alors il est possible de retrouver le mot de code d'origine. Par contre, si le mot reçu est au delà (point rouge), on risque d'être plus proche d'un autre mot de code, et se tromper.

1.1.3 Capacité d'un canal

Si on se donne un canal, on se pose naturellement la question suivante : quels codes peut-on utiliser sur ce canal pour transmettre fidèlement de l'information ? Peut-on faire tendre la probabilité d'erreur vers zéro ? Il semble effectivement assez intuitif de penser que plus le canal est « bruité » (d'une façon qu'il faut définir), plus il faudra mettre de redondance dans l'information, c'est-à-dire utiliser un code de rendement plus faible ; mais plus le canal est « fidèle », plus le rendement du code pourra être grand.

Les lois de la théorie de l'information répondent à cette question, en définissant la *capacité* d'un canal. Cette capacité représente grossièrement la proportion de bits qu'on peut transmettre : une capacité de 1 représente un canal non bruité, une capacité de 0 un canal entièrement bruité sur lequel on ne peut rien transmettre.

Définition 9 (Capacité d'un canal classique)**.** *Soit un canal C de $\mathcal{A}$ vers $\mathcal{B}$ donné par les probabilités de transition $p(y \mid x)$. La variable aléatoire représentant les mots envoyés est notée X, et celle représentant les mots reçus est notée Y. On appelle* capacité *du canal la grandeur suivante :*

$$\sup_{X,Y} I(X;Y)$$

où $I(X;Y)$ *est l'*information mutuelle *entre* X *et* Y*, et vaut :*

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

où H *est l'*entropie *d'une variable aléatoire, et se calcule :*

$$H(X) = -\sum_x p(x) \log p(x)$$
$$H(X \mid Y) = -\sum_{x,y} p(x,y) \log p(x \mid y)$$

où pour simplifier l'écriture, $p(x) = \mathbb{P}[X = x]$, $p(x,y) = \mathbb{P}[X = x, Y = y]$, $p(x \mid y) = \mathbb{P}[X = x \mid Y = y]$.

Tous les « log » de la définition, ainsi que tous ceux présents dans le document, seront pris en base 2 sauf indication contraire.

On peut ainsi calculer la capacité des canaux connus :

Propriété 1 (Capacité du canal binaire symétrique)**.** *La capacité du canal binaire symétrique de paramètre* p *est donnée par :*

$$C_{cbs}(p) = 1 - h(p, 1-p) \tag{1.1}$$

où $h(p_1, \ldots, p_k) = -\sum_{i=1}^{k} p_i \log p_i$.

Ainsi, par exemple, un canal binaire symétrique de paramètre $1/2$ a une capacité nulle, et rien ne peut y transiter (chaque bit transmis sera aléatoirement 0 ou 1, peu importe ce qu'on y envoie), tandis qu'avec un paramètre proche de 0 (très peu d'erreurs), la capacité est proche de 1, ce qui confirme l'intuition qu'avec un tel canal, on peut transmettre ce qu'on veut sans problèmes.

Propriété 2 (Capacité du canal à effacements)**.** *La capacité du canal à effacements de paramètre* p *est donnée par :*

$$C_{eff}(p) = 1 - p \tag{1.2}$$

La capacité est assez intuitive ici : si on a une proportion p des bits effacés, alors cela signifie qu'on ne peut transmettre qu'une fraction $1 - p$ de bits. Avec $p = 1$ par exemple, tous les symboles sont effacés : il n'y a aucune chance de pouvoir transmettre quoi que ce soit. Avec $p = 0$, il n'y a aucun effacement, donc on n'a pas besoin de code correcteur pour transmettre.

Démonstration. Le calcul détaillé des capacités du canal binaire symétrique et du canal à effacements est entre autres donnée dans [CT06].

■

Cette capacité représente également le rendement maximal d'un code qui permettrait de transmettre l'information fidèlement :

Théorème 1 (Deuxième théorème de Shannon). *Soit un canal de capacité C. Pour tout $R < C$ et $\varepsilon > 0$, il existe un code de rendement R tel que, pour tout mot de code émis, la probabilité d'erreur après décodage soit inférieure à ε.*

Ce qui est moins intuitif, c'est que la probabilité d'erreur après décodage tende vers zéro lorsqu'on choisit un rendement en deçà de la borne (même si ce rendement ne tend pas vers zéro).

Mais de façon générale, le théorème ne fournit pas de code simple pour atteindre cette borne. Plus précisément, le théorème atteint la borne par construction de codes aléatoires ; or de tels codes sont inutilisables en pratique.

1.2 Codes binaires linéaires

Dans la pratique, le décodage au maximum de vraisemblance n'est pas utilisé, car il y faudrait calculer la distance ou les probabilités avec les $(\#\mathcal{A})^k$ mots de code pour chaque mot reçu, ce qui serait beaucoup trop long. De la même façon, si on prend un code général au sens de la définition 4, il faut retenir les $(\#\mathcal{A})^k$ mots de code qui vont avec les mots en entrée, ce qui est trop gros à stocker.

C'est pourquoi on cherche des cas particuliers de ces codes, notamment les codes linéaires, et même dans cette catégorie on cherche des sous-cas simples à décoder.

1.2.1 Définition

Un code linéaire se définit sur n'importe quel alphabet $\mathcal{A}$ muni d'une structure de corps (fini), mais à partir de cette section, on ne considérera (sauf exception) que des codes *binaires* : $\mathcal{A} = \mathbb{F}_2$, c'est-à-dire $\{0, 1\}$ muni de l'addition et de la multiplication binaires.

Définition 10 (Code linéaire). *Un code $\mathscr{C}$ de longueur n est linéaire si l'ensemble de ses mots de code est un sous-espace vectoriel de $\{0, 1\}^n$.*

On remarque quasiment immédiatement que si un code linéaire $\mathscr{C}$ a pour dimension k (en tant qu'espace vectoriel), alors sa dimension (au sens de la définition 4) est k : le terme de « dimension » est donc bien choisi.

On peut alors définir de plusieurs façons un tel code : par l'ensemble image d'une application linéaire (ou une base des mots de code), ou par le noyau d'une autre application linéaire.

Définition 11 (matrice génératrice, matrice de parité)**.** *Soit $\mathscr{C}$ un code de longueur n et de dimension k. On appelle* matrice génératrice *de $\mathscr{C}$ toute matrice de n colonnes $\mathbf{G}$ qui vérifie :*

$$\mathscr{C} = \operatorname{Im} \mathbf{G} = \{x\mathbf{G}, x \in \{0,1\}^k\}$$

On appelle également matrice de parité *(ou parfois* matrice de contrôle*) de $\mathscr{C}$ toute matrice de n colonnes $\mathbf{H}$ qui vérifie :*

$$\mathscr{C} = \operatorname{Ker} \mathbf{H} = \{x \in \{0,1\}^n, \mathbf{H}\,{}^{\mathbf{t}}(x) = 0\}$$

où ${}^{\mathbf{t}}(x)$ désigne le vecteur x transposé (et de même pour une matrice).

Pour un code donné, il existe plusieurs matrices de parité et matrices génératrices possibles, mais comme on définira souvent un code par l'une ou l'autre de ces matrices, on parlera (abusivement) de *sa* matrice génératrice/de parité. Si le code est de dimension k, la matrice génératrice possède k lignes, et la matrice de parité au moins $n-k$ lignes. On a également un lien entre les deux :

Propriété 3. *Soit un code linéaire $\mathscr{C}$, $\mathbf{G}$ sa matrice génératrice, $\mathbf{H}$ sa matrice de parité. On a alors :*

$$\mathbf{G}\,{}^{\mathbf{t}}(\mathbf{H}) = 0$$

Démonstration. Chaque ligne de $\mathbf{G}$ est un mot de code ; chaque ligne g vérifie donc $\mathbf{H}\,{}^{\mathbf{t}}(g) = 0$, autrement dit $g\,{}^{\mathbf{t}}(\mathbf{H}) = 0$ pour toute ligne de $\mathbf{G}$, d'où la propriété. ■

On peut retrouver la dimension via la matrice de parité ou la matrice génératrice : pour un code donné par $\mathbf{G}$, on a $k = \dim \mathscr{C} = \operatorname{rang} \mathbf{G} = n - \operatorname{rang} \mathbf{H}$

1.2.2 Propriétés

On peut noter que la matrice génératrice fournit naturellement une fonction d'encodage de $\{0,1\}^k \to \mathscr{C} \subset \{0,1\}^n$: la fonction d'encodage ne prend plus une place exponentielle (stocker toutes les images des éléments de $\{0,1\}^k$), mais quadratique (une matrice de taille $k \times n$). La matrice de parité, quand à elle, fournit un moyen rapide de vérifier si un mot x appartient au code : il suffit de multiplier H par ${}^{\mathbf{t}}(x)$.

Définition 12 (Syndrome)**.** *Soit $\mathscr{C}$ un code de longueur n et $\mathbf{H}$ sa matrice de parité. Pour un mot x de $\{0,1\}^n$, on appelle* syndrome *de x, et on note $s(x)$ le mot suivant :*

$$s(x) = \mathbf{H}\,{}^{\mathbf{t}}(x)$$

Le syndrome représente l'information qu'on peut utiliser pour le décodage : le syndrome est nul si et seulement si on a affaire à un mot de code.

La distance minimale peut également s'exprimer de façon simplfiée :

Propriété 4 (Distance minimale d'un code linéaire)**.** *Soit un code linéaire $\mathscr{C}$. Sa distance minimale s'exprime :*

$$d = \min\{\mathrm{w}(x), x \in \mathscr{C}, x \neq 0\}$$

où $\mathrm{w}(x) = d(x, 0^n)$ *est le* poids de Hamming.

Démonstration. Par définition, $d = \min\{d(x,y), x, y \in \mathscr{C}, x \neq y\}$. Or, $d(x,y) = \#\{i, x_i \neq y_i\} = d(x-y, 0^n) = \mathrm{w}(x-y)$. Or comme le code est linéaire, $x - y \in \mathscr{C}$, donc $\{x-y, x \in \mathscr{C}, y \in \mathscr{C}, x \neq y\} = \{z, z \in \mathscr{C}, z \neq 0\}$, d'où la propriété. ■

Le syndrome permet de caractériser l'erreur lorsque celle-ci n'est pas trop grosse, car deux mots différents de moins de la distance minimale ont un syndrome différent. En effet soient x et y deux mots reçus ayant le même syndrome, différents de moins de la distance minimale. On a $\mathbf{H}\,^{\mathbf{t}}(x) = \mathbf{H}\,^{\mathbf{t}}(y)$, donc $\mathbf{H}\,^{\mathbf{t}}(x-y) = 0$: $x-y$ est un mot du code, de poids inférieur à la distance minimale, c'est donc que $x - y = 0$.

Les codes linéaires sont donc une sous-famille des codes généraux, plus simples à manipuler. Cependant, le décodage n'est pas encore acquis : de façon générale, il n'y a pas de méthode efficace pour trouver le mot de code le plus proche à partir du syndrome (ou même du mot reçu).

1.2.3 Exemples

Parmi les exemples de codes linéaires connus, le *code à répétition* de longueur n (celui qui répète n fois le symbole 0 ou 1) est un code linéaire : sa matrice génératrice $1 \times n$ est $(11\ldots1)$, et sa matrice de parité $n-1 \times n$ est :

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \vdots & & & \ddots & \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

En effet, on peut constater que si on a un mot de code x la première ligne impose $x_1 = x_2$, la seconde $x_1 = x_3$, etc : on finit par avoir $x_1 = x_2 = \ldots = x_n$. Sa distance minimale est de n puisqu'il n'y a que deux mots de code, 0^n et 1^n.

Le *code de parité* de longueur n (celui qui prend $n-1$ symboles, et ajoute à la fin la somme de ces symboles) est un code linéaire de matrice génératrice :

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \vdots & & & \ddots & \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

et de matrice de parité : $(11\ldots1)$.

1.3 Codes LDPC

De façon générale, le décodage d'un code linéaire quelconque donné par sa matrice de parité (ou sa matrice génératrice), est difficile. C'est pourquoi on regarde encore des cas particuliers, qu'on sait facilement décoder. Les codes LDPC sont un de ces cas-là : ils sont d'abord faciles à générer et à stocker, et il existe pour eux un algorithme de décodage à la fois efficace et rapide, le *décodage par propagation de croyances* (aussi appelé BP, pour « Belief Propagation »). On pourra consulter [Gal62, FZ99, MN97, Mac99, LMSS01] pour plus de détails à propos de ces codes et de leur décodage.

Ces codes sont, comme leur nom en anglais l'indique (« Low Density Parity Check »), des codes à matrice de parité peu dense, c'est-à-dire contenant peu de « 1 » (à partir de là, tous les codes seront donnés par leur matrice de parité).

Définition 13 (Code LDPC)**.** *Une famille de codes linéaires* $(\mathscr{C}_n)_n$ *est dite* LDPC *si il existe pour tout* n *une représentation* creuse *de la matrice de parité* $\mathbf{H}^n$ *de* $\mathscr{C}_n$ *; c'est-à-dire s'il existe deux constantes* A *et* B *telles que :*

$$\forall n, \forall i, \#\{j, \mathbf{H}^n_{i,j} = 1\} \leqslant A$$

$$\forall n, \forall j, \#\{i, \mathbf{H}^n_{i,j} = 1\} \leqslant B$$

Cette définition implique, entre autres, que le nombre de 1 total de la matrice est au plus linéaire en n.

Dans la pratique, on parlera d'un code LDPC « seul » lorsque le nombre de 1 par ligne et colonne est faible, même si rigoureusement la définition n'a de sens que pour une famille de codes.

Le premier avantage d'avoir une matrice de parité creuse est le stockage : si on choisit de ne stocker que les 1 de la matrice, l'espace mémoire nécessaire est en $O(n \log n)$, au lieu de $O(n^2)$, pour un code de longueur n. Par contre, la matrice génératrice prend toujours une place quadratique, mais une matrice

génératrice creuse induirait (par définition) des mots de code de poids faible, ce qui est rarement souhaitable.

Le second avantage des codes LDPC, et le plus important, est qu'on dispose d'un algorithme de décodage très efficace, en temps ($O(n)$), qui fonctionne pour à peu près tous les codes de ce type, et sur tous les canaux (voir section 1.3.2).

1.3.1 Graphe de Tanner

Pour bien comprendre l'algorithme BP expliqué plus loin, on choisit de représenter le code par un graphe, dit *graphe de Tanner* (voir [Tan81]), bien que cette représentation ne soit pas spécifique aux codes LDPC. De plus, cette représentation permet aussi d'utiliser un certain nombre d'outils venus de la théorie des graphes.

Définition 14 (Graphe de Tanner)**.** *Soit $\mathscr{C}$ un code linéaire de matrice de parité* $\mathbf{H}$ *de taille $r \times n$. On définit $\mathscr{G}_{\mathscr{C}}$ le* graphe de Tanner *de $\mathscr{C}$ le graphe biparti suivant :*

$$\mathscr{G}_{\mathscr{C}} = (V, C, E)$$

où :
- *V est l'ensemble des* nœuds variable*, de taille n,*
- *C est l'ensemble des* nœuds de parité*, de taille r,*
- *$E = \{v \rightarrowtail c, v \in V, c \in C, \mathbf{H}_{c,v} = 1\}$ est l'ensemble des arêtes.*

Il faut se rappeler que chaque ligne de la matrice $\mathbf{H}$ représente une équation de parité (d'où le terme de nœud de parité) que doit vérifier tout mot de code, de la forme $x_{i_1} + \ldots + x_{i_l} = 0$, où $x = (x_1, \ldots, x_n)$ représente le vecteur de bits d'un mot de code. On choisit donc de représenter cette équation par un nœud particulier appelé nœud de parité, relié à des nœuds (appelés nœuds variable) indexés par les i_j (voir l'exemple plus bas).

La plupart du temps, V et C sont $\{1, \ldots n\}$ et $\{1, \ldots r\}$, mais il est parfois pratique d'utiliser d'autres indexations.

Par exemple, si on considère la matrice de parité suivante :

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{1.3}$$

Le graphe de Tanner de ce code est donné par la figure 1.2.

1.3.2 Algorithme de propagation de croyances (BP)

L'algorithme de propagation de croyance (également appelé *décodage itératif*, ou en anglais *Belief Propagation*, d'où le raccourci BP), comme son nom l'indique,

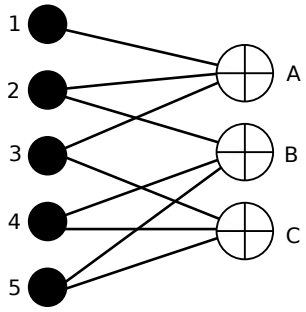


FIGURE 1.2 – Graphe de Tanner du code donné par la matrice de parité de l'équation 1.3. On a choisi d'indexer les nœuds de variable (colonnes de la matrice) par des chiffres, et les nœuds de parité (lignes de la matrice) par des lettres.

consiste à propager des « croyances » le long des arêtes du graphes de Tanner ; des croyances selon lesquelles un bit vaudrait 0 ou 1 sur le mot reçu. Cet algorithme a l'avantage de s'exécuter en un temps linéaire en le nombre d'arêtes du graphes, ce qui le rend évidemment parfaitement adapté pour les codes LDPC, pour qui ce nombre d'arêtes est linéaire en leur longueur. Il est, de plus, très efficace en terme de décodage tant que le graphe de Tanner vérifie certaines conditions, notamment ne pas contenir de petit cycle, et s'adapte à des canaux variés.

Le but de cette section est de présenter rapidement le fonctionnement de cet algorithme (qu'on désignera par « algorithme BP »), afin de comprendre ses mécanismes, sans forcément donner tous les détails précis et rigoureux.

1.3.2.1 Principe de fonctionnement

Pour comprendre l'idée de cet algorithme, on peut observer la figure 1.3. Si on connaît avec certitude les bits sur les nœuds 1, 2 et 3, que le bit 4 est inconnu, et qu'on a une équation de parité $x_1 + x_2 + x_3 + x_4 = 0$, alors on peut en déduire la valeur du bit sur le nœud 4.

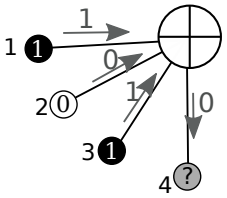


FIGURE 1.3 – Exemple de propagation de croyances sur le graphe de Tanner : on en déduit que le bit en position 4 doit être 1.

Plus généralement, chaque nœud variable envoie à ses voisins non pas juste un

bit (indiquant s'il est 0 ou 1) mais une *probabilité* (d'être 0 ou 1). Ensuite, en tenant compte des probabilités reçues et du fait que les nœuds de parité (équations de parité) doivent être satisfaits, l'algorithme calcule de nouvelles probabilités, qu'il envoie le long des arêtes voisines (d'où le terme de *propagation* de croyances), et ainsi de suite. Lorsqu'un certain nombre d'itérations ont été effectuées, on peut déduire la probabilité finale d'avoir 0 ou 1 sur chaque nœud variable, et selon laquelle des deux est plus élevée, décider d'affecter 0 ou 1 à un bit donné.

1.3.2.2 Définition des variables

On utilise pour cet algorithme des variables indexées par $\{0,1\}$ et par les nœuds variables ou les arêtes. On travaille dans le graphe $\mathscr{G}(\mathscr{C}) = (V, C, E)$ avec $\#V = n, \#C = r$, et on identifiera V à $\{1, \dots n\}$ et C à $\{1, \dots r\}$ pour simplifier. Pour un nœud variable $v \in V$, on définit son *voisinage* comme l'ensemble des arêtes reliées à v : $\mathcal{V}_E(v) = \{v \multimap c \in E\}$. On définit de la même façon le voisinage d'un nœud de parité c : $\mathcal{V}_E(c) = \{v \multimap c \in E\}$.

On suppose que y est le mot reçu sur le canal, et que x est le mot envoyé qu'on essaie de retrouver ($x, y \in \{0,1\}^n$).

On définit quatre types de grandeurs :

- $\forall v \in V$, $b \in \{0,1\}$,
$$p_v^i(b) = \mathbb{P}[x_v = b \mid y_v] \tag{1.4}$$
la probabilité initiale que le bit x_v soit égal à b. Cette valeur est donnée grâce aux données du canal et grâce à y_v. C'est l'entrée de l'algorithme, et elle n'est pas modifiée.
- $\forall e = v \multimap c \in E$, $b \in \{0,1\}$,
$$q_e(b) = \mathbb{P}[x_v = b \mid y_v, r'_e, e' \in \mathcal{V}_E(v), e' \neq e] \tag{1.5}$$
la probabilité qui va d'un nœud variable à un nœud de parité le long de l'arête e, et utilisant les messages entrant sur le nœud v (sauf le message venant de e),
- $\forall e = v \multimap c \in E$, $b \in \{0,1\}$,
$$r_e(b) = \mathbb{P}[x_v = b \mid q'_e, e' \in \mathcal{V}_E(c), e' \neq e] \tag{1.6}$$
la probabilité qui va d'un nœud de parité à un nœud variable, le long de l'arête e, et en utilisant les messages entrant sur le nœud de parité c (sauf celui venant de e),
- $\forall v \in V$, $b \in \{0,1\}$,
$$p_v^f(b) = \mathbb{P}[x_v = b \mid r_e, e \in \mathcal{V}_E(v)] \tag{1.7}$$

la probabilité finale que x_v soit b. Elle utilise les messages r arrivant des arêtes adjacentes à ce nœud variable.

Ces grandeurs sont représentées sur la figure 1.4. Chacune d'elles dépend d'un bit $b \in \{0, 1\}$, donc en pratique, une seule des deux probabilités est stockée (la somme des deux étant égale à 1).

1.3.2.3 Mise à jour des messages

Ensuite il faut comprendre comment ces grandeurs sont calculées et mises à jour au fur et à mesure. Pour cela, on suppose que le graphe est un arbre, et on « suit » (voir figure 1.4) les messages qui convergent vers un nœud variable v_1.

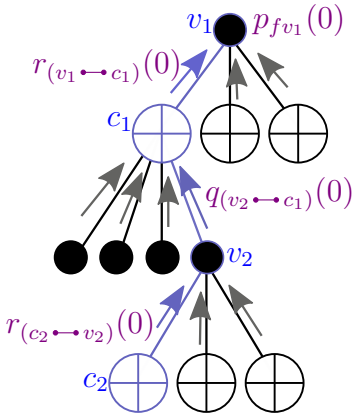


FIGURE 1.4 – Propagation des différentes grandeurs : r_e, q_e, p_v^f : on cherche à calculer p_v^f en tenant compte des messages arrivant le long de l'arbre.

On a alors :

- Calcul des p^f :

$$p_{v_1}^f(b) \propto p_{v_1}^i(b) \prod_{e \in \mathcal{V}_E(v_1)} r_e(b) \tag{1.8}$$

  En effet, comme on peut le voir sur la partie supérieure de la figure 1.4, il y a un bit b en position v_1 si tous les nœuds de parité voisins « acceptent » b comme valeur de bit pour v_1, d'où le produit des $r_e(b)$; et si le bit *initial* en position v_1 est bien b, d'où le produit par $p_{v_1}^i$. On suppose que le graphe est un arbre, les différentes probabilités sont donc bien indépendantes (d'où le produit). Avec $b = 0$ et $b = 1$, tous les cas n'ont pas été traités : il faut donc calculer les deux valeurs, puis renormaliser afin d'avoir $p_{v_1}^f(0) + p_{v_1}^f(1) = 1$.
- Calcul des r_e :

$$r_{v_1 \rightarrow c_1}(b) = \sum_{\substack{b_1 + \ldots + b_d = b \\ d = \#\mathcal{V}_E(c_1) - 1}} \prod_{e_i \in \mathcal{V}_E(c_1) \setminus v_1 \rightarrow c_1} q_{e_i}(b_i) \tag{1.9}$$

En effet, on veut calculer la probabilité que le bit v_1 soit égal à b, tout en satisfaisant le nœud de parité c_1. Il faut donc que la somme de tous les bits des nœuds voisins de c_1 soit égale à 0, autrement dit, si ces bits sont $b_1, \dots b_d$, alors il faut qu'on ait $b_1 + \dots + b_d = b$. Pour chaque telle combinaison de bits, il faut estimer la probabilité qu'elle apparaisse, d'où le produit des q_{e_i}. Il n'est pas nécessaire ici de renormaliser, on peut constater assez rapidement que $r_e(0) + r_e(1) = 1$.
Il peut sembler, à première vue, difficile de calculer en temps court ces valeurs, car il faut faire la somme de 2^d produits. Or si on écrit :

$$\begin{aligned} r_e(0) - r_e(1) &= \sum_{b_1+\dots+b_d=0} \prod_i q_{e_i}(b_i) - \sum_{b_1+\dots+b_d=1} \prod_i q_{e_i}(b_i) \\ &= \sum_{b_1,\dots,b_i} (-1)^{b_1+\dots+b_i} \prod_i q_{e_i}(b_i) \\ &= \sum_{b_1,\dots,b_i} \prod_i (-1)^{b_i} q_{e_i}(b_i) \end{aligned}$$

On a donc une somme de 2^d termes, dont chaque terme est un produit de q_{e_i}, avec un signe $-$ si $b_i = 1$ et un signe $+$ sinon : cette somme est donc égale à :

$$\Delta r_e = r_e(0) - r_e(1) = \prod_i (q_{e_i}(0) - q_{e_i}(1)) = \prod_i \Delta q_e \tag{1.10}$$

qui est facile à calculer en $O(d)$. Avec $r_e(0) + r_e(1) = 1$, on peut en déduire $r_e(0)$ et $r_e(1)$.

- Calcul des q_e :

$$q_{v_2 \bullet\!\!-\!\!\circ c_1}(b) \propto p^i_{v_2}(b) \prod_{e \in \mathcal{V}_E(v_2) \setminus v_2 \bullet\!\!-\!\!\circ c_1} r_e(b) \tag{1.11}$$

En effet, pour que le bit sur v_2 soit égal à b, tout comme dans le cas de p_f, il faut que chaque arête entrante « accepte » b comme valeur (sans l'arête $v_2 \bullet\!\!-\!\!\circ c_1$), et que b soit le bit de départ sur v_2, d'où le produit (là encore, la condition d'indépendance vient du fait que le graphe est un arbre).

- Initialisation des p^i_v :
Cette probabilité ne dépend que du canal : elle est donc calculée une fois pour toutes au début, de la façon suivante :

$$p^i_v(b) = \frac{p(y_v \mid b)}{p(y_v \mid 0) + p(y_v \mid 1)} \tag{1.12}$$

où les $p(y \mid x)$ sont les probabilités de transition associées au canal.
Ainsi, dans le cas du canal binaire symétrique de probabilité p, si le bit reçu

est 0, alors $p_v^i(0) = 1 - p$, $p_v^i(1) = p$, ce qui signifie qu'on « pense » que le bit est plutôt 0, mais qu'il a tout de même une petite probabilité d'être 1. Pour le canal à effacements, si le bit reçu est 0, alors $p_v^i(0) = 1$, tandis que si on reçoit e (effacement), on n'a aucune information : $p_v^i(0) = 1/2$.

Ainsi, on peut calculer toutes les grandeurs, en commençant par les p_v^i grâce à l'information directement reçue du canal, puis les q_e et r_e alternativement, pendant un certain nombre d'itérations. Enfin, on calcule les probabilités finales p_v^f.

1.3.2.4 Déroulement de l'algorithme

Plus succintement, voici le déroulement de l'algorithme, en pseudo-code, en ne gardant que les valeurs pour 0 (autrement dit, on écrit r_e pour $r_e(0)$) :

Algorithme Propagation de croyances
{ *Initialisation* }
Calculer $p_v^i \quad \forall v \in V$ avec 1.12
$p_v^f \leftarrow p_v^i \forall v \in V$
$q_{v \bullet\!\!-\!\!\bullet c} \leftarrow p_v^i \quad \forall v \bullet\!\!-\!\!\bullet c \in E$
{ *Déroulement* }
TantQue nombre d'itérations $\leqslant$ NB_MAX, et que w change **Faire**
  Incrémenter le nombre d'itérations
  Calculer les Δq_e
  En déduire les Δr_e et mettre à jour les r_e avec 1.10
  Mettre à jour les q_e avec 1.11
  Si nombre d'itérations mod $\delta = 0$ **Alors**
    Mettre à jour les p_v^f avec 1.8
    Mettre à jour w : $w(v) = 0$ si $p_v^f \geqslant 0.5$, $w(v) = 1$ sinon
  FinSi
  Retourner w

Le nombre d'itérations NB_MAX n'a pas besoin d'augmenter avec la taille du code : en pratique 150 itérations suffisent. Le δ est une constante servant à accélerer un peu l'algorithme : il n'est pas nécessaire de calculer les p_v^f à chaque itération, on peut le faire uniquement toutes les $\delta = 10$ itérations.

1.3.2.5 Performances

Propriété 5 (Complexité de l'algorithme BP)**.** *L'algorithme BP décrit dans cette section a une complexité temporelle et spatiale en $O(n)$ pour un code LDPC.*

Démonstration. Le nombre d'itérations étant constant, la complexité de l'algorithme est essentiellement celle d'un tour de boucle. Chaque mise à jour (r_e, q_e ou p_f) s'exécute en un temps constant puisque le degré des nœuds est borné. Il y a O(nombre d'arêtes) telles mises à jour par tour de boucle, et comme le nombre d'arêtes est en $O(n)$, on a un temps d'exécution en $O(n)$.

Pour la complexité spatiale, il faut juste stocker toutes les grandeurs : r_e, q_e, p_v^i, p_v^f, qui nécessitent un espace mémoire en O(nombre d'arêtes) $= O(n)$.

■

Le graphe de Tanner est rarement un arbre, mais dans la pratique, tant que ce graphe ne contient pas de petit cycle (de taille 4 ou 6), l'algorithme fonctionne très bien.

Lorsqu'on effectue des tests de performances, l'algorithme étant parfaitement symétrique en 0 et 1, il suffit de le tester pour le mot de code 0^n.

Chapitre 2

Notions d'informatique quantique

Ce chapitre est une brève introduction à l'informatique quantique, et s'adresse à des lecteurs non familiers avec le monde du quantique. Tout comme le chapitre précédent, il n'a pas pour but d'être complet sur le sujet, mais de contenir les notions de base nécessaires à la compréhension des codes quantiques, introduits au chapitre suivant : pour une présentation plus complète, on pourra se référer à [Pre99] par exemple. Un lecteur déjà familier avec le quantique peut (s'il ne passe pas directement au chapitre suivant) trouver le formalisme choisi différent de celui auquel il est habitué : il n'y a pas de manière universelle de le présenter, celle-ci est choisie pour sa simplicité [1].

On rappelle ici l'expérience du *chat de Schrödinger* expliquée dans l'introduction. Les principes de la mécanique quantique soutiennent que, si on enferme un chat dans une boîte avec un poison ayant une chance sur deux de se répandre, le chat n'est ni dans l'état « mort », ni dans l'état « vivant », mais tant que la boîte est hermétiquement fermée, dans une *superposition* des deux, et ce, jusqu'à ce qu'on ouvre la boîte.

Cette expérience peut sembler à première vue non seulement improbable (mais de nombreuses observations témoignent de ce phénomène dans l'infiniment petit), mais surtout inutile : que nous importe-t-il de savoir que le chat est dans un étrange état transitoire superposé, si lorsqu'on ouvre la boîte on obtient un chat mort ou un chat vivant, tous les deux parfaitement normaux, avec la probabilité prévue au départ ? En fait, il est possible d'effectuer des opérations, bien particulières, sur ces objets superposés, et surtout sur des combinaisons particulières de plusieurs de ces chats quantiques. Et contrairement à ce que pourrait nous dicter l'intuition, partir d'états superposés, effectuer des manipulations dessus puis les mesurer n'est pas du tout équivalent à tirer l'état au hasard parmi les différentes superpositions, *puis* effectuer les manipulations dessus.

1. Du moins, du point de vue de l'auteur.

Un certain nombre d'algorithmes ont été développés sur ce modèle, et ils sont effectivement plus efficaces (en temps le plus souvent) que leurs équivalents classiques (ou classiques aléatoires). Le plus connu est probablement l'algorithme de Shor [Sho97], permettant de factoriser un nombre en un temps polynomial en sa taille. Il est également possible d'utiliser les propriétés du quantique pour chiffrer des messages, en utilisant le fait qu'un espion modifiera nécessairement le message transmis en l'observant ; on pourra consulter [BB84] [2] pour plus de détails sur la transmission de clé grâce au quantique.

2.1 Définition d'un état quantique

Le *support* de l'information quantique n'est pas, au grand soulagement des amis des animaux, des chats dans des boîtes, mais est basé sur le spin d'un électron, les niveaux d'énergie d'un atome, ou la polarisation d'un photon. Ici, on traitera uniquement son aspect théorique : on manipulera donc des *qubits*, qu'on définira formellement, sans se préoccuper de son support physique.

Le qubit (pour bit quantique) est donc l'unité de base sur laquelle on travaille, il est à l'informatique quantique ce que le bit est à l'informatique classique. Si un bit (classique) peut valoir 0 ou 1, un qubit est une *superposition* de 0 et de 1 :

Définition 15 (qubit)**.** *On appelle* qubit *un vecteur* $(\alpha, \beta) \in \mathbb{C}^2$ *tel que* $|\alpha|^2 + |\beta|^2 = 1$.

Les nombres complexes α et β représentent des sortes de « probabilités » (on le verra dans la section 2.2.2) qu'on ait 0 (première coordonnée) ou 1 (seconde coordonnée).

Si un qubit est équivalent à 2 nombres complexes, un système composé de 2 qubits est équivalent à 4 nombre complexes, et plus généralement, un système composé de n qubits comprendra 2^n nombres complexes :

Définition 16 (Système quantique)**.** *On appelle* système quantique *ou* état quantique *de taille n (ou composé de n qubits) un vecteur de* $\mathbb{C}^{2^n}$ *normé, c'est-à-dire un vecteur* $(\alpha_1, \ldots, \alpha_{2^n})$ *tel que* $\sum_{i=1}^{2^n} |\alpha_i|^2 = 1$.

On notera $|\varphi\rangle$ les qubits (ou les systèmes quantiques), et on notera $|0\rangle = (1, 0)$ et $|1\rangle = (0, 1)$ les vecteurs de la base orthonormée canonique pour les qubits individuels. Ainsi, on écrira communément un qubit $\alpha|0\rangle + \beta|1\rangle$.

Pour les systèmes à n qubits, on notera $|0 \ldots 0\rangle$ le système $|0\rangle \otimes |0\rangle \otimes \ldots \otimes |0\rangle$, et de façon générale, pour une chaîne de bits $b_1, \ldots b_n$, le système composé du produit

2. Une version vulgarisée est également disponible sur wikipedia : `http://fr.wikipedia.org/wiki/Cryptographie_quantique#Protocole_BB84`

tensoriel correspondant : $|b_1 b_2 \dots b_n\rangle = |b_1\rangle \otimes \dots \otimes |b_n\rangle$. Les $|b_1 \dots b_n\rangle$ représentent les vecteurs de la base orthonormée canonique de l'espace associé $\mathbb{C}^{2^n}$.

On écrira également, pour un système quelconque $|\varphi\rangle = (\alpha_1, \dots \alpha_{2^n})$, son transconjugué (ou adjoint) :

$$\langle\varphi| = (\alpha_1, \dots, \alpha_{2^n})^* = \begin{pmatrix} \bar{\alpha}_1 \\ \vdots \\ \bar{\alpha}_{2^n} \end{pmatrix}$$

Par exemple, $|0\rangle, \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle, |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ sont des systèmes quantiques à un qubit.

Si on réunit deux systèmes indépendants $|\varphi\rangle$ et $|\varphi'\rangle$, alors le système quantique qu'on obtient est le produit tensoriel des deux :

$$|\psi\rangle = |\varphi\rangle \otimes |\varphi'\rangle$$

Par exemple, si on réunit les système $|0\rangle$ et $|+\rangle$, on a :

$$|0\rangle \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{|00\rangle + |01\rangle}{\sqrt{2}}$$

En revanche, un système quantique ne peut pas forcément s'exprimer comme un produit tensoriel, par exemple le système suivant ne peut pas :

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.1}$$

Ce système particulier est appelé *paire EPR*, ou *paire de qubits maximalement intriqués*, on verra ses propriétés dans la section 2.2.2.

On notera $\langle\varphi|\psi\rangle$ le produit scalaire hermitien de deux systèmes quantiques $|\varphi\rangle$ et $|\psi\rangle$: la notation avec les $|\ \rangle$ prend alors son sens, puisqu'on peut écrire $\langle\varphi|\psi\rangle = \langle\varphi|\,|\psi\rangle$.

2.2 Opérations de base

Sur un système quantique, il est possible d'effectuer deux types d'opérations : l'opération unitaire, qui est réversible et la mesure, qui est irréversible. Avec ces deux opérations, il est possible de construire des circuits quantiques, qui sont des équivalents de circuits classiques avec des portes AND, NOT, OR, etc.

2.2.1 L'opération unitaire

La première opération possible sur les qubits est l'opération unitaire :

Définition 17 (Opération unitaire)**.** *Une opération unitaire sur un système de n qubits est une matrice U de $\mathbb{C}$ de taille $2^n \times 2^n$ qui vérifie :*

$$UU^* = I$$

On rappelle que U^* est le transconjugué (ou adjoint) de U : si $U = (u_{i,j})_{i,j}$ alors $U^* = (\bar{u}_{j,i})_{i,j}$.

Si $|\varphi\rangle$ est un état quantique, l'état transformé par U sera noté $U|\varphi\rangle$.

Entre autres, cela implique que la matrice U est inversible (donc l'opération est réversible), et que le vecteur reste bien normé après application de U. De plus, comme un produit de telles matrices est également unitaire, on peut donc combiner des opérateurs en multipliant les matrices.

Chaque opération unitaire correspond à une *porte quantique*, qui sur un circuit quantique possède n entrées et n sorties si la matrice est de taille 2^n : il s'agit simplement d'une représentation visuelle de l'opérateur.

Par exemple, la porte quantique de la figure 2.1, agissant sur 2 qubits, est appelée *control-not*, et sa matrice dans la base canonique $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ est :

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

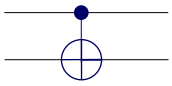


FIGURE 2.1 – Porte dite « control not » sur un circuit quantique.

Si la porte quantique (correspondante à U) est de taille inférieure au nombre de qubits total du système, et qu'on désire appliquer la porte quantique sur les premiers qubits (par exemple), il suffira de considérer l'opérateur $U \otimes I \otimes I \otimes \ldots \otimes I$ en choisissant le bon nombre de I pour avoir un opérateur de la bonne taille (voir figure 2.2 pour un exemple de circuit, avec $H = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$).

On peut noter que le circuit possède toujours autant d'entrées que de sorties : cela est relativement naturel puisqu'il est entièrement réversible.

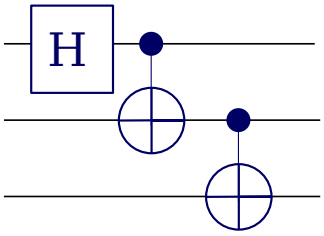


FIGURE 2.2 – Un exemple de circuit quantique à trois qubits et trois portes quantiques. L'opérateur unitaire qui le représente est $(H \otimes I \otimes I) \times (C \otimes I) \times (I \otimes C)$.

2.2.2 La mesure

La mesure est l'autre opération fondamentale en quantique. C'est l'ouverture de la boîte contenant le chat de Schrödinger, et contrairement aux opérations unitaires, cette opération est *irréversible* : une fois qu'on a mesuré l'état mort ou vivant du chat, on ne peut pas revenir à l'état superposé. C'est également la seule qui permette de donner une information sur l'état quantique : si on n'ouvre pas la boîte, impossible de savoir ce qui s'y passe.

Définition 18. *Soit $|\varphi\rangle$ un état quantique de taille n, et $(P_i)_{i \in I}$ un ensemble de projecteurs orthogonaux (pour le produit scalaire hermitien $\langle \ | \ \rangle$) de $\mathbb{C}^{2^n}$ tels que $\sum_i P_i = I$. La mesure de l'état $|\varphi\rangle$ selon les P_i s'effectue de la façon suivante : Avec probabilité $||P_j |\varphi\rangle||^2$, l'état $|\varphi\rangle$ est projeté selon P_i, et devient alors $\frac{P_i|\varphi\rangle}{||P_i|\varphi\rangle||}$, et la mesure donne accès au résultat $i \in I$ ainsi obtenu.*

Cette mesure signifie qu'on projette (puis renormalise) l'état sur un ensemble de sous-espaces de $\mathbb{C}^{2^n}$, et qu'on a alors comme information dans quel sous-espace l'état se retrouve alors (mais pas à l'état ainsi projeté). Par exemple, sur un état à un seul qubit $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, on choisit P_0 le projecteur sur le vecteur $|0\rangle$ ($P_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$) et P_1 le projecteur sur $|1\rangle$ ($P_1 = |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$).
La mesure de l'état $|\varphi\rangle$ donne alors :

- soit $|0\rangle$ avec probabilité $|\alpha|^2$,
- soit $|1\rangle$ avec probabilité $|\beta|^2$.

On remarque que, grâce à la définition 15 que la somme de $|\alpha|^2$ et de $|\beta|^2$ fait 1, et comme il s'agit de nombres réels positifs, ils sont nécessairement compris entre 0 et 1 : ce sont bien des probabilités valides.

Sur un système de taille n, on peut de la même façon projeter sur la base canonique de $\mathbb{C}^{2^n}$. Si cette base s'écrit $(|i\rangle)_{i \in \{0,1\}^n}$, alors on définit $P_i = |i\rangle\langle i|$, et si $|\varphi\rangle = \sum_{i \in \{0,1\}^n} \alpha_i |i\rangle$, alors la probabilité d'obtenir $|i\rangle$ est d'exactement $|\alpha_i|^2$.

Si on veut projeter sur une base orthonormée quelconque $(|\psi_i\rangle)_i$, on peut définir

$P_i = |\psi_i\rangle\langle\psi_i|$; on aura alors une probabilité $|| |\psi_i\rangle\langle\psi_i| |\varphi\rangle ||^2 = \langle\psi_i|\varphi\rangle^2$ d'obtenir $|\psi_i\rangle$.

Dans ces exemples, on mesure l'« ensemble » du système, mais il est possible de ne mesurer que certains qubits. Par exemple, pour un système de taille 2, on peut définir les projecteurs suivants :

$$\begin{aligned} P_0 &= |0\rangle\langle 0| \otimes I \\ P_1 &= |1\rangle\langle 1| \otimes I \end{aligned}$$

La mesure correspondante consiste à mesurer uniquement le premier qubit. Par exemple si notre état s'écrit $|\varphi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$, le résultat de la mesure est alors :

- $\frac{\alpha}{\sqrt{|\alpha|^2+|\beta|^2}}|00\rangle + \frac{\beta}{\sqrt{|\alpha|^2+|\beta|^2}}|01\rangle$ avec probabilité $|\alpha|^2 + |\beta|^2$,
- $\frac{\gamma}{\sqrt{|\gamma|^2+|\delta|^2}}|10\rangle + \frac{\gamma}{\sqrt{|\gamma|^2+|\delta|^2}}|01\rangle$ avec probabilité $|\gamma|^2 + |\delta|^2$

En particulier, si on prend un état quantique factorisé $|\varphi\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$, le résultat de la mesure est :

- $|0\rangle \otimes (c|0\rangle + d|1\rangle)$ avec probabilité $|a|^2$,
- $|1\rangle \otimes (c|0\rangle + d|1\rangle)$ avec probabilité $|b|^2$.

Ce qui est raisonnablement intuitif : un état factorisé représente deux qubits « indépendants », donc la mesure de l'un des deux qubits suit les règles de mesure d'un qubit tout seul, et ne perturbe pas l'autre.

Par contre, si on prend la paire EPR (état maximalement intriqué), i.e. $|\varphi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, la mesure du premier qubit donne :

- $|00\rangle$ avec probabilité 1/2,
- $|11\rangle$ avec probabilité 1/2.

Là, la mesure du premier qubit a perturbé le second qubit : si le premier qubit est mesuré à $|0\rangle$, alors le second sera *forcément* mesuré à $|0\rangle$ également, ce qui peut sembler contre-intuitif au premier abord, et qui en fait est un des principes fondamentaux du quantique : la mesure perturbe potentiellement *tout* le système.

2.2.3 Le non-clonage

La règle de *non-clonage* n'est pas un principe, c'est un résultat, mais mérite une section à son nom. En effet, les principes ci-dessus impliquent qu'il est impossible de cloner un qubit (ou plus généralement un système quantique). Cela vient entre autres du fait que sur un qubit $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ on ne peut pas avoir accès « directement » à α et β : si on mesure l'état pour avoir une estimation des probabilités $|\alpha|^2$ et $|\beta|^2$, l'état est ensuite « perdu ».

Théorème 2 (Théorème du non-clonage [WZ82])**.** *Il n'existe pas d'opération unitaire U agissant sur des systèmes de $2n$ qubits, qui vérifie, pour tout état $|\varphi\rangle$ de n*

qubits :

$$U\left(|\varphi\rangle \otimes |0^n\rangle\right) = |\varphi\rangle \otimes |\varphi\rangle$$

Démonstration. Supposons qu'il existe une telle opération U. On a alors, pour deux systèmes $|\varphi\rangle$ et $|\psi\rangle$:

$$\begin{aligned} U\left(|\varphi\rangle \otimes |0^n\rangle\right) &= |\varphi\rangle \otimes |\varphi\rangle \\ U\left(|\psi\rangle \otimes |0^n\rangle\right) &= |\psi\rangle \otimes |\psi\rangle \end{aligned}$$

On peut donc faire le produit scalaire de ces deux systèmes, on obtient alors :

$$\begin{aligned} \left(\langle 0^n| \langle\varphi|\right) U^* U \left(|\varphi\rangle \otimes |0^n\rangle\right) &= \left(\langle\psi| \otimes \langle\psi|\right)\left(|\varphi\rangle \otimes |\varphi\rangle\right) \\ \langle 0^n| \langle|\psi\rangle| \, |\varphi\rangle \, |0^n\rangle &= \langle\psi| \langle\psi| \, |\varphi\rangle \, |\varphi\rangle \\ \langle\psi|\varphi\rangle &= \langle\psi|\varphi\rangle^2 \end{aligned}$$

Or, cela n'est possible que si $\langle\psi|\varphi\rangle$ est égal à 0 ou 1, donc uniquement dans le cas d'états orthogonaux ou égaux, et pas pour tous $|\varphi\rangle$ et $|\psi\rangle$.

■

Cependant, la preuve met en valeur le fait que, si on sait que le qubit est un des vecteurs d'une base orthonormée qu'on connaît, alors on peut l'identifier entièrement (et donc le recréer à l'identique), en mesurant dans cette base orthonormée : le vecteur sera projeté avec probabilité 1 sur lui-même, et on saura sur quel vecteur il a été projeté. Mais dans le cas général, ce n'est pas possible.

Chapitre 3

Codes correcteurs quantiques

Ce chapitre présente tout d'abord des généralités sur les codes correcteurs quantiques, avant de parler plus précisément des codes *stabilisateurs* (section 3.3), qui sont une sous-classe des codes quantiques, plus simples à manipuler (ils sont l'équivalent des codes linéaires pour le monde classique). Enfin, les codes *CSS* (section 3.4) sont une sous-classe des codes stabilisateurs particulièrement aisée à manipuler, car très proche des codes classiques. Les codes stabilisateurs LDPC (CSS ou non) ont droit à un chapitre à part entière, le chapitre 4. La fin du chapitre (section 3.7) présente un bref état de l'art des codes correcteurs quantiques.

Comme expliqué dans l'introduction, il peut arriver que la fameuse boîte du chat zombie de Schrödinger (contenant un état qu'on notera $\alpha|$  $\rangle + \beta|$  $\rangle$, avec α et β qui correspondent à des probabilités de mesurer le chat dans un état ou un autre) soit mal fermée : si anodin que cela peut paraître au premier abord, cette imperfection peut avoir de graves conséquences, car en quantique, toute observation perturbe le système.

Si, par exemple, on entend un miaulement venant de la boîte, on n'a pas besoin d'ouvrir la boîte pour savoir que le chat est vivant : on vient de le mesurer sans le faire exprès ; l'état quantique est alors projeté sur $|$  $\rangle$. Pire encore, il se pourrait qu'en l'absence du physicien, une petite souris entende les miaulements : non seulement le chat est tout de même projeté, mais on n'a même plus accès au résultat de la mesure.

Ainsi, on peut constater des modifications involontaires sur l'état quantique, qui rappellent (dans le principe) les erreurs qui peuvent survenir sur un canal de communication, dans le cas classique (voir chapitre 1). Dans ce dernier cas, on se prémunit contre ces erreurs via des codes correcteurs ; on cherche donc ici des équivalents quantiques : des *codes correcteurs quantiques*.

Le premier code correcteur quantique a été introduit par Shor dans [Sho95].

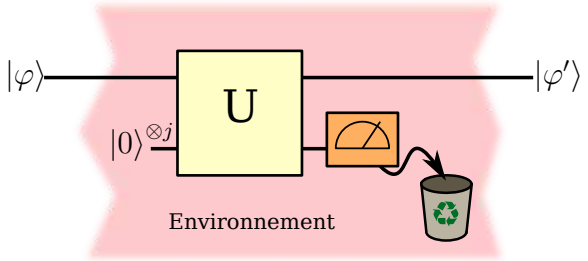


FIGURE 3.1 – Modélisation du canal quantique : l'environnement agit sur le système $|\varphi\rangle$ en appliquant une matrice unitaire au système $|\varphi\rangle \otimes |0\rangle^{\otimes j}$, les $|0\rangle$ étant des qubits venus de l'environnement. Puis l'environnement mesure les qubits supplémentaires, sans utiliser le résultat de la mesure : on obtient alors un état modifié $|\varphi'\rangle$.

Peu après est créée une classe particulière de codes, les codes CSS :[CS96, Ste96], et leur généralisation, les codes stabilisateurs dans [Got97, CRSS98]. Ces codes (contrairement aux codes quantiques généraux, complexes a étudier) ont l'avantage de ressembler beaucoup à leurs homologues classiques, et profitent, dans une certaine mesure, des résultats connus dans le monde des codes classiques.

3.1 Qu'est-ce qu'une erreur quantique ?

Pour un système quantique un peu plus général, il peut arriver à tout moment des interactions avec l'environnement, et des mesures involontaires, qui apportent des modifications au système. On considère ici une version un peu simplifiée, où ces modifications ne peuvent arriver qu'à *certains moments*. Plus précisément, on prendra comme modèle celui de la *communication quantique* : le système quantique doit traverser une zone bruitée, qu'on appellera *canal quantique*, et on supposera qu'avant et après cette étape il n'y a pas d'erreurs sur le système.

L'échange avec l'environnement se modélise comme sur la figure 3.1 : le système interagit avec des qubits de l'environnement, puis ces qubits sont mesurés, et le résultat de la mesure est jeté (on n'y a plus accès).

Le résultat, en pratique (on peut calculer l'effet de ces transformations), se manifeste par des matrices unitaires appliquées avec certaines probabilités. Par exemple, il se peut que chaque qubit subisse une opération $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ avec une certaine probabilité (cette matrice change $|0\rangle$ en $|1\rangle$ et inversement). Si on a un état $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, il y a une certaine probabilité que le qubit en sortie soit $\beta|0\rangle + \alpha|1\rangle$. Comment faire pour corriger ces modifications, sachant qu'on ne peut

pas cloner $|\varphi\rangle$, et qu'on ne peut pas « regarder » l'état sans le modifier ?

On peut tout de même (voir chapitre 2) appliquer des transformations unitaires, faire intervenir des qubits supplémentaires (qu'on supposera initialisés à $|0\rangle$), et faire des mesures, ce qui est déjà bien. Par exemple, on peut utiliser le circuit (dit circuit d'encodage) suivant : une porte control-not sur les deux premiers qubits, et une sur les deux derniers (voir figure 3.2).

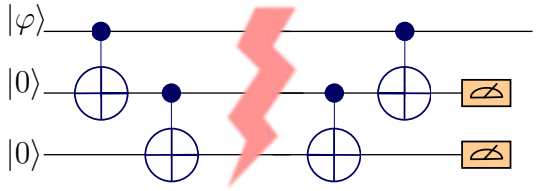


FIGURE 3.2 – Un petit circuit d'encodage, pour un qubit : le qubit $|\varphi\rangle$ est le qubit de départ, on ajoute deux qubits de redondance, avant d'appliquer deux portes control-not. Après le passage dans le canal, on applique la tranformation inverse, et on mesure les qubits supplémentaires.

Ce circuit construit l'état $\alpha\,|000\rangle + \beta\,|111\rangle$ (à partir de $\alpha|0\rangle + \beta|1\rangle$ et de deux qubits supplémentaires). Il ne s'agit pas de clonage : cloner le qubit $\alpha\,|0\rangle + \beta\,|1\rangle$ aurait donné $(\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle)$.

Après le passage dans le canal, on applique le circuit inverse, puis on mesure les deux qubits qu'on avait ajoutés. Alors, s'il n'y a eu qu'une seule erreur telle que décrite plus haut ($|0\rangle$ changé en $|1\rangle$ et inversement), on peut retrouver l'état de départ. En effet, voyons les quatre cas possibles :

- Aucune erreur n'est survenue sur le système. On a donc en sortie du canal $\alpha\,|000\rangle + \beta\,|111\rangle$, ce qui après l'application du circuit de sortie donne $\alpha\,|000\rangle + \beta\,|100\rangle$: on mesure donc deux fois 0 (avec probabilité 1), on en déduit qu'aucune erreur n'est arrivée.
- Une erreur est survenue sur le premier qubit. On a donc en sortie $\alpha\,|100\rangle + \beta\,|011\rangle$, ce qui donne après le circuit $\alpha\,|010\rangle + \beta\,|110\rangle$. On mesure 1 et 0 (avec probabilité 1).
- Une erreur est survenue sur le second qubit. On a donc en sortie $\alpha\,|010\rangle + \beta\,|101\rangle$, et après le circuit $\alpha\,|011\rangle + \beta\,|111\rangle$. On mesure donc 1 et 1 avec probabilité 1.
- Une erreur est survenue sur le troisième qubit. On a donc en sortie $\alpha\,|001\rangle + \beta\,|110\rangle$, et après le circuit $\alpha\,|001\rangle + \beta\,|101\rangle$. On mesure donc 0 et 1 avec probabilité 1.

Dans ces quatre cas, on a eu des mesures différentes, on peut donc associer une correction à chacune de ces mesures.

Ce schéma n'est pas sans rappeler le code à répétition (de longueur 3), sur le

canal binaire symétrique classique : en « triplant » d'une certaine façon le symbole ou le qubit à transmettre, on est capable de corriger jusqu'à une erreur. Bien sûr, cela ne couvre pas tous les cas d'erreur en quantique : il y a d'autres erreurs possibles, d'une part ; et d'autre part l'erreur qui survient n'est pas forcément discrète, c'est-à-dire du type « tout ou rien ».

Dans le premier cas, par exemple, il pourrait survenir l'erreur $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. L'état est alors transformé en $\alpha\,|000\rangle - \beta\,|111\rangle$, et les manipulations décrites plus haut ne permettent pas d'identifier l'erreur.

Dans le second cas, en revanche, tout se passe bien : supposons qu'au lieu d'une erreur nette sur le premier qubit, on obtienne une *superposition*, c'est-à-dire qu'on ait en sortie $\alpha\sqrt{1-\varepsilon}\,|000\rangle + \alpha\sqrt{\varepsilon}\,|100\rangle + \beta\sqrt{1-\varepsilon}\,|111\rangle + \beta\sqrt{\varepsilon}\,|011\rangle$ pour un certain $\varepsilon > 0$. Après passage dans le circuit de décodage, on obtient $\alpha\sqrt{1-\varepsilon}\,|000\rangle + \alpha\sqrt{\varepsilon}\,|110\rangle + \beta\sqrt{1-\varepsilon}\,|100\rangle + \beta\sqrt{\varepsilon}\,|010\rangle$. À la mesure des deux derniers qubits, il y a alors deux possibilités :

- Soit on mesure les qubits supplémentaires à 1 et 0 (probabilité ε). On a alors projeté sur l'état $\alpha\,|110\rangle + \beta\,|010\rangle$, comme si on avait eu une erreur « entière » sur le premier qubit.
- Soit on mesure les qubits supplémentaires à 0 et 0 (probabilité $1-\varepsilon$). On a alors projeté sur l'état $\alpha\,|000\rangle + \beta\,|100\rangle$, comme si on n'avait pas eu d'erreur du tout.

On peut constater que, dans les deux cas, si on suit la même logique que plus haut, on corrigera l'erreur convenablement. Grâce à la projection, on se retrouve dans un cas équivalent au « tout ou rien ».

3.2 Le canal dépolarisant

Comme dans le cas classique, on se restreindra à un type particulier d'erreur, l'équivalent des canaux sans mémoire, où les erreurs touchent les qubits indépendamment. Cela signifie donc que pour un système de n qubits, une erreur $E \in \mathbb{U}_{2^n}$ est composée d'erreurs individuelles appliquées à chaque qubit : $E = E_1 \otimes E_2 \otimes \ldots \otimes E_n$ pour $E_i \in U_2$.

Comme expliqué plus haut, on peut se ramener à un cas d'erreurs discrètes ; ainsi, au lieu de considérer toutes les matrices de rotation complexes possibles, on peut supposer que E_i est l'une des quatre *erreurs de Pauli*, définies plus bas.

Au final, au lieu de travailler avec un espace continu de dimension 2^n sur $\mathbb{C}$, on se ramène à un espace *fini*, beaucoup plus simple à étudier.

Définition 19 (Groupe de Pauli)**.** *Soient les quatre matrices suivantes, dites* ma-

trices de Pauli :

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ Y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix},\ Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

On définit $\mathcal{E}_n$ le groupe de Pauli *(ensemble des erreurs de Pauli) de taille n de la façon suivante :*

$$\mathcal{E}_n = \{1, -1, i, -i\} \ \times \ \{I, X, Y, Z\}^{\otimes n} \tag{3.1}$$

On notera, pour $E \in \mathcal{E}_n$, $E(i) \in \mathbb{U}_2$ le i-ème terme de ce produit :

$$E = \bigotimes_{i=1}^{n} E(i)$$

Pour alléger les notations, au lieu d'écrire $E = E(1) \otimes E(2) \otimes \ldots \otimes E(n)$, on écrira $E = [E(1)\, E(2) \ldots E(n)]$.

On peut noter que $XZ = -iY, XY = -iZ, YZ = iX$, et $XZ = -ZX, XY = -YX, YZ = -ZY$. De plus, $X^2 = Y^2 = Z^2 = I$. On en déduit donc que deux erreurs de Pauli soit commutent, soit anti-commutent : $E_1 E_2 = \pm E_2 E_1$. Plus précisément, deux telles erreurs commutent s'il y a un nombre pair de positions i telles que $E_1(i) \neq E_2(i)$, $E_1(i) \neq I$ et $E_2(i) \neq I$.

Par la suite, on se préoccupera peu de ce qu'on appelle la *phase* : le facteur 1, -1, i, ou $-i$ devant le produit, sauf en ce qui concerne la commutation. On définit donc l'opérateur suivant :

Définition 20 (Opérateur $\star$)**.** *Soient E_1 et $E_2 \in \mathcal{E}_n$ deux erreurs de Pauli. On définit :*

$$E_1 \star E_2 = \begin{cases} 0 \ si\ E_1 E_2 = E_2 E_1 \\ 1 \ sinon \end{cases}$$

On peut également écrire la définition précédente : $E_1 E_2 = (-1)^{E_1 \star E_2} E_2 E_1$.

Propriété 6 (Distributivité de $\star$)**.** *Soient $E_1, E_2, E_3 \in \mathcal{E}_n$. On a la relation suivante :*

$$(E_1 E_2) \star E_3 = E_1 \star E_3 + E_2 \star E_3$$

Démonstration.

$$\begin{aligned} E_1 E_2 E_3 &= E_1 (-1)^{E_2 \star E_3} E_3 E_2 \\ (-1)^{(E_1 E_2) \star E_3} E_3 E_1 E_2 &= (-1)^{E_2 \star E_3} (-1)^{E_1 \star E_3} E_3 E_1 E_2 \\ (-1)^{(E_1 E_2) \star E_3} E_3 E_1 E_2 &= (-1)^{E_2 \star E_3 + E_1 \star E_3} E_3 E_1 E_2 \end{aligned}$$

■

Avec cette proriété, on peut réécrire l'expression de $E_1 \star E_2$ de la façon suivante :

$$E_1 \star E_2 = \sum_{i=1}^{n} E_1(i) \star E_2(i) \mod 2$$

Ainsi, l'opérateur $\star$ est un indicateur de la commutation, aisé à manipuler, et sera très utile par la suite.

À partir du groupe de Pauli, on peut définir le *canal de Pauli* :

Définition 21 (Canal de Pauli). *Soit $|\psi\rangle$ un système quantique de n qubits. Le canal de Pauli de paramètres p_I, p_X, p_Y, p_Z (avec $p_I + p_X + p_Y + p_Z = 1$) appliqué à $|\psi\rangle$ donne l'état $E|\psi\rangle$ où E est définie de la façon suivante : pour tout $1 \leqslant i \leqslant n$, on tire les $E(i)$ indépendamment avec*

- *$E(i) = I$ avec probabilité p_I*
- *$E(i) = X$ avec probabilité p_X*
- *$E(i) = Y$ avec probabilité p_Y*
- *$E(i) = Z$ avec probabilité p_Z*

Ce canal rappelle un peu le canal binaire symétrique du monde classique, sauf qu'en quantique il n'y a pas une seule façon de causer une erreur sur un qubit : on a 3 matrices d'erreurs possibles (X, Y ou Z), au lieu d'une seule erreur possible en classique (inverser un bit).

On traitera ici surtout d'un cas particulier du canal de Pauli, le *canal dépolarisant* :

Définition 22 (Canal dépolarisant). *Le canal dépolarisant de paramètre p est le canal de Pauli ayant pour paramètres $p_I = 1 - p$ et $p_X = p_Y = p_Z = \frac{p}{3}$.*

Il existe d'autres sortes de canaux, y compris parmi les canaux discrets, mais le canal dépolarisant est le plus étudié, car il est l'équivalent quantique du canal binaire symétrique (qui est également le modèle d'erreur le plus étudié dans le monde classique). Pourtant, malgré sa simplicité, sa *capacité* ; c'est-à-dire, informellement, quelle proportion de qubits on peut espérer transmettre avec une probabilité d'erreur qui tende vers zéro lorsque la longueur du code augmente ; est encore inconnue (voir section 3.6).

3.3 Codes stabilisateurs

Les codes correcteurs quantiques généraux sont potentiellement très complexes à étudier et à décoder. Aussi, on s'intéressera à un cas particulier de codes, les codes stabilisateurs, introduits originellement dans [CRSS98]. Ils sont l'équivalent

quantique des codes linéaires : beaucoup plus simples, mais potentiellement tout aussi performants.

La présentation ci-dessous peut varier de celle des références en question (ou d'autres références sur les codes stabilisateurs), essentiellement pour des raisons d'uniformisation.

3.3.1 Schéma d'encodage et de décodage

On va généraliser l'exemple du code de la section 3.1, et pour cela, on considèrera le schéma d'encodage suivant (voir figure 3.3) :

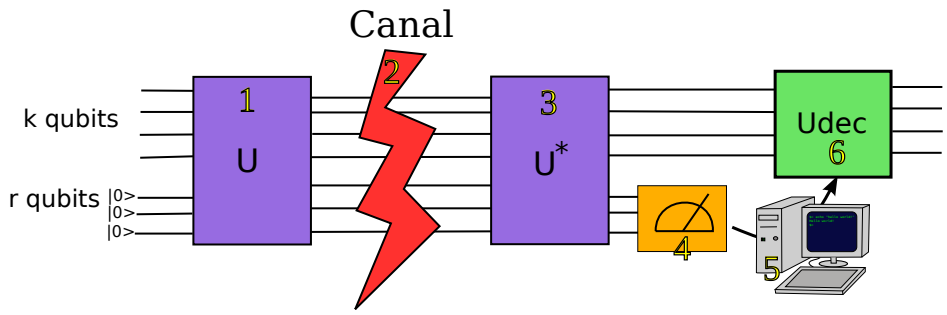


FIGURE 3.3 – Schéma d'encodage et de décodage d'un code stabilisateur. On part d'un message $|\psi\rangle$ de k qubits et de r qubits supplémentaires, et on construit (1) un état encodé $|\varphi\rangle$ qui est envoyé dans le canal (2). À la sortie, on applique U^* (3), puis on mesure les k qubits supplémentaires (4). Avec le résultat de la mesure, on tente de trouver l'erreur causée par le canal (5), et on applique son inverse (6).

On part de l'état à encoder $|\psi\rangle$, composé de k qubits. On se munit de r qubits supplémentaires, initialisés à $|0\rangle$,

1. On applique la transformation $U \in \mathbb{U}_{2^n}$, avec $n = k + r$, on obtient l'état $|\psi\rangle = U\left(|\varphi\rangle \otimes |0\rangle^{\otimes r}\right)$,
2. On envoie l'état $|\psi\rangle$ dans le canal, on obtient donc en sortie $|\varphi'\rangle = E\,|\psi\rangle = EU\left(|\varphi\rangle \otimes |0\rangle^{\otimes r}\right)$, où E est l'erreur du canal,
3. On applique la transformation U^* : on obtient $|\psi'\rangle = U^*\,|\varphi'\rangle = U^*EU\left(|\varphi\rangle \otimes |0\rangle^{\otimes r}\right)$,
4. On mesure les r qubits supplémentaires,
5. Avec la donnée de la mesure, on tente de retrouver quelle est l'erreur E, autrement dit, c'est le décodage (et il est fait classiquement),
6. On applique la transformation U_{dec} appropriée qui est censée corriger l'erreur.

On peut noter que, dans ce schéma, après l'étape 3, on obtient une erreur équivalente U^*EU sur le système d'origine. La matrice U étant fixée, il y a un

nombre fini de telles erreurs, autant que d'erreurs dans le groupe de Pauli. Mieux, on aimerait que U^*EU soit elle-même une erreur de Pauli (car ce groupe est simple à engendrer et à manipuler), il est donc nécessaire que U soit un opérateur de Clifford :

Définition 23 (Opérateur de Clifford)**.** *Un opérateur de Clifford de taille n est une matrice $U \in \mathbb{U}_{2^n}$ qui laisse le groupe de Pauli globalement invariant par conjugaison, c'est-à-dire :*

$$\mathcal{E}_n = U^* \mathcal{E}_n U$$

On peut déjà remarquer que, dans ce schéma, il existe des erreurs bénignes. En effet, si $E = UZ_iU^*$ où Z_i contient une seule erreur Z, en position $k+i$ et I ailleurs ($Z_i(k+i) = Z$, $Z_i(j) = I$ sinon) :

$$\begin{aligned} |\psi'\rangle &= U^*EU\left(|\psi\rangle \otimes |0\rangle^{\otimes r}\right) \\ &= Z_i\left(|\psi\rangle \otimes |0\rangle^{\otimes r}\right) \\ &= |\psi\rangle \otimes |0\rangle^{\otimes r} \end{aligned}$$

Rien n'est détecté par la mesure des r qubits supplémentaires, mais l'état quantique $|\psi\rangle$ n'a pas été affecté.

De façon générale, cette mesure de r qubits renseigne sur la commutation ou non avec ces erreurs particulières :

Propriété 7. *Soit U un opérateur de Clifford, et $S_i = UZ_iU^*$ tel que défini ci-dessus. Alors, si on a une erreur E sur le canal, la mesure du i-ième qubit supplémentaire à l'étape 4 du schéma 3.3 ($|\psi\rangle$) donne $E \star S_i$.*

Démonstration. On note $s_i = E \star S_i$. On a :

$$\begin{aligned} EUZ_iU^* &= (-1)^{s_i} UZ_iU^*E \\ U^*EUZ_i &= (-1)^{s_i} Z_iU^*EU \\ U^*EUZ_i\left(|\psi\rangle \otimes |0\rangle^{\otimes r}\right) &= (-1)^{s_i} Z_iU^*EU\left(|\psi\rangle \otimes |0\rangle^{\otimes r}\right) \\ |\psi'\rangle &= (-1)^{s_i} Z_i |\psi'\rangle \end{aligned}$$

Si $s_i = 1$, cela veut dire qu'appliquer Z_i (une seule erreur Z en position $k+i$) fait changer le signe de $|\psi'\rangle$: cela veut dire qu'on a un $|1\rangle$ sur ce $k+i$-ième qubit. Le raisonnement inverse fonctionne avec $s_i = 0$.

■

Avec ces propriétés, on peut donc constater qu'on a trois types d'erreurs :
- celles qui anti-commutent avec les S_i : elles sont détectées et peuvent être corrigées (éventuellement)

- celles qui sont engendrées par les S_i : elles sont indétectées, mais n'affectent pas le système quantiques (erreurs bénignes)
- celles qui commutent avec les S_i mais qui ne sont pas engendrées par les S_i : elles sont indétectées et affectent le système quantique, ce sont les « vraies » erreurs, appelées *erreurs sérieuses*.

On met en valeur ici un phénomène qui n'arrive pas dans le monde classique : certaines erreurs sont en réalité bénignes et n'affectent pas le système. Si ce phénomène, appelé *dégénérescence*, est plutôt une bonne nouvelle, c'est également la raison pour laquelle la capacité des canaux quantiques est si difficile à évaluer.

3.3.2 Définition des codes stabilisateurs

On constate qu'à cette étape, toutes les caractéristiques du code peuvent être exprimées via les erreurs de Pauli, un groupe discret de taille 4^n (à la phase près), au lieu d'avoir à considérer les états quantiques (dans $\mathbb{C}^{2^n}$). Mieux encore, toute l'étude des erreurs peut se faire via les S_i, qui sont un sous-groupe particulier d'erreurs. On choisira donc de tout redéfinir via ces erreurs particulières :

Définition 24 (Code stabilisateur)**.** *Soit $\mathcal{S}$ un sous-groupe commutatif de $\mathcal{E}_n$, ne contenant pas $-I^{\otimes n}$. On définit le code stabilisateur $\mathscr{C}$ stabilisé par $\mathcal{S}$:*

$$\mathscr{C} = \left\{ |\varphi\rangle \in \mathbb{C}^{2^n}, S\,|\varphi\rangle = |\varphi\rangle\,, \forall S \in \mathcal{S} \right\} \tag{3.2}$$

$\mathcal{S}$ est appelé le stabilisateur *de $\mathscr{C}$, et on appelle les $|\varphi\rangle \in \mathscr{C}$ des états du code.*

On supposera que, pour tout groupe stabilisateur $\mathcal{S}$, il existe un opérateur de Clifford U qui vérifie UZ_iU^*.

On peut remarquer que si $\mathcal{S}$ contient $-I^{\otimes n}$, alors les états du code vérifient $|\varphi\rangle = -\,|\varphi\rangle$, ce qui n'est pas possible. De même, si le groupe $\mathcal{S}$ n'est pas commutatif, on obtient $|\varphi\rangle = S_1S_2\,|\varphi\rangle = -S_2S_1\,|\varphi\rangle = -\,|\varphi\rangle$ et on a le même problème.

On peut définir la dimension, c'est-à-dire le nombre de qubits encodés :

Définition 25 (Dimension d'un code stabilisateur)**.** *Soit un code $\mathscr{C}$ stabilisé par $\mathcal{S}$, de n qubits. On définit la* dimension du code $\mathscr{C}$*, et on note* $\dim(\mathscr{C})$ *la grandeur k telle que $\mathscr{C}$ soit un sous-espace de dimension 2^k sur $\mathscr{C}$.*

Dans le schéma précédent, on avait $\mathcal{S} = \langle UZ_iU^* \rangle$, $\mathcal{S}$ était de dimension r et on obtenait pour $\mathscr{C}$ un espace de dimension $2^k = 2^{n-r}$, autrement dit on encodait $k = n - r$ qubits. On peut le prouver de façon plus générale :

Propriété 8 (Dimension d'un code stabilisateur)**.** *Soit $\mathcal{S}$ le stabilisateur d'un code $\mathscr{C}$ sur n qubits. On note r le nombre de générateurs indépendants de $\mathcal{S}$. Alors*

$$\dim(\mathscr{C}) = n - r$$

Démonstration. On a besoin dans un premier temps du lemme suivant :

Lemme 1. *Soit $\mathcal{S} = \langle S_1, \dots S_r \rangle$ un stabilisateur avec les S_i indépendants. Il existe $(T_i), i = 1 \dots r$ tels que : $S_i \star T_j = \delta_{ij}$.*

Démonstration. On montre le résultat par récurrence. Pour $r = 1$, il suffit de trouver un T qui anti-commute avec S.

Tout d'abord, puisque $\mathcal{S}$ est un stabilisateur, il ne contient pas $-I$, de même il ne peut pas contenir iI ou $-iI$, sinon $-I$ serait dans le stabilisateur. S est donc un élément non trivial de E_n, qu'on écrit $S = [E_{S1} \dots E_{Sn}]$. On peut alors prendre $T = [E_{T1} \dots E_{Tn}]$, avec $E_{Ti} = E_{Si}$, sauf en une des coordonnées j où E_{Sj} n'est pas l'identité, et choisir E_{Tj} de façon à ce que E_{Tj} et E_{Sj} anti-commutent.

Ensuite, si $\mathcal{S} = \langle S_1, \dots S_r \rangle$, avec les S_i indépendants, supposons qu'on peut construire de tels T_i pour $i = 1, \dots r-1$.

On cherche alors à construire un élement T_r tel que :
- $T_r \star S_i = 0$, pour $i \leqslant r-1$,
- $T_r \star S_r = 1$.

Supposons qu'il n'existe pas de tel élément. On écrit alors $S' = S_r \prod_{i \in I} S_i$, où l'ensemble d'indices I est $I = \{i, S_r \star T_i = 1\}$. Cela permet d'assurer que $S' \star T_i = 0$ pour tout i.

Soit $E \in \mathcal{E}_n$, telle que $E \star S' = 1$. Si E commute avec tous les S_i, alors cela signifie que $E \star S' = E \star S_r = 1$, cela contredit l'hypothèse comme quoi il n'existe pas d'élément qui anti-commute avec S_r tout en commutant avec les $S_i, i \leqslant r-1$. Si E anti-commute avec certains S_i, on peut multiplier E par les T_i correspondants : on a alors E' qui commute avec tous les S_i. Or $E' \star S' = E \star S'$, car les T_i commutent avec S' par construction. On a aussi une contradiction.

Donc tous les éléments de $\mathcal{E}_n$ commutent avec S', ce qui implique que S' est triviale : ce n'est pas possible car on a supposé S_r indépendant des S_i pour $i \leqslant r-1$.

Une fois ce T_r trouvé, on a alors $T_r \star S_i = \delta_{ri}$. On peut alors poser $T'_i = T_i$ si $T_i \star S_r = 0$, et $T'_i = T_r T_i$ sinon : les T'_i vérifient alors $T'_i \star S_r = 0$, et les T'_i vérifient toujours $T'_i \star S_j = T_i \star S_j + T_r \star S_j = \delta_{ij}$.

■

On commence par prouver le résultat dans le cas où $r = 1$, donc $\mathcal{S} = \langle S \rangle$. On peut noter que $S^2 = I^{\otimes n}$. En effet, $S^2 = \pm I^{\otimes n}$ par construction de $\mathcal{E}_n$; et si on avait $S^2 = -I^{\otimes n}$, on aurait $-I^{\otimes n} \in \mathcal{S}$, ce qui n'est pas autorisé par la définition 24. Or comme $SS^* = I$, cela signifie que $S^* = S$: la matrice est donc hermitienne et peut être diagonalisée en base orthonormée (dans une base qu'on appellera $(|\varphi_i\rangle), i = 1, \dots 2^n - 1$). De plus l'équation $S^2 = I$ indique qu'il y a deux valeurs propres : 1 et -1.

De là, on peut donc en conclure que, pour tout état $|\varphi_i\rangle$ ci-dessus, on a $S|\varphi_i\rangle = \pm|\varphi_i\rangle$.

Or, comme S est différent de l'identité, il existe dans $\mathcal{E}_n$ un élément T qui anti-commute avec S. On a alors, pour tout état $|\varphi_i\rangle$ de la base :

$$\begin{aligned} S|\varphi_i\rangle = |\varphi_i\rangle &\Leftrightarrow ST|\varphi_i\rangle = -TS|\varphi_i\rangle \\ &\Leftrightarrow S(T|\varphi_i\rangle) = -(T|\varphi_i\rangle) \end{aligned}$$

Ainsi, la multiplication par T construit une bijection entre les vecteurs $|\varphi_i\rangle$ qui sont dans $\mathscr{C}$ (qui vérifient $S|\varphi_i\rangle = |\varphi_i\rangle$) et ceux qui vérifient $S|\varphi_i\rangle = -|\varphi_i\rangle$. Les deux ensembles de vecteurs de base sont donc de taille égale, on en déduit donc qu'il y a $2^n/2 = 2^{n-1}$ vecteurs de base dans $\mathscr{C}$, ce qui prouve ce qu'on voulait démontrer.

Si on suppose qu'on a pu montrer le résultat pour $r-1$ éléments indépendants dans $\mathcal{S}$, soit un code $\mathscr{C}$ stabilisé par $\mathcal{S} = \langle S_1, \dots S_r\rangle$. On appelle $\mathscr{C}'$ le code stabilisé par $\mathcal{S}' = \langle S_1, \dots S_{r-1}\rangle$. $\mathscr{C}'$ est de dimension 2^{n-r+1}. Soit T l'élément T_r construit grâce au lemme 1. Soit $|\varphi\rangle \in \mathscr{C}$. Alors $T_r|\varphi\rangle$ vérifie :

$$ST_r|\varphi\rangle = T_rS|\varphi\rangle = T_r|\varphi\rangle$$

pour $S \in \mathcal{S}'$, donc $T_r|\varphi\rangle \in \mathscr{C}'$. Par contre,

$$S_rT_r|\varphi\rangle = -T_rS_r|\varphi\rangle = -T_r|\varphi\rangle$$

donc $T_r|\varphi\rangle \notin \mathscr{C}$. De la même façon, si $|\varphi\rangle \in \mathscr{C}'\backslash\mathscr{C}$, on obtient $T_r|\varphi\rangle \in \mathscr{C}$. Ainsi la multiplication par T_r établit une bijection entre $\mathscr{C}$ et $\mathscr{C}'\backslash\mathscr{C}$, donc les deux espaces ont même dimension, qui est donc de moitié de la dimension de $\mathscr{C}'$, ce qui donne que la dimension de $\mathscr{C}$ est de $2^{n-r+1}/2 = 2^{n-r}$.

■

3.3.2.1 Détection d'erreurs

Après avoir défini le code, on va voir ici que la détection et la correction d'erreurs peut se définir uniquement à partir de $\mathcal{E}_n$.

Définition 26 (Centralisateur de $\mathcal{S}$)**.** *Soit $\mathcal{S}$ le stabilisateur d'un code, on appelle* centralisateur de $\mathcal{S}$ *et on note $C(\mathcal{S})$ l'ensemble de tous les éléments de $\mathcal{E}_n$ qui commutent avec $\mathcal{S}$:*

$$C(\mathcal{S}) = \{E \in \mathcal{E}_n,\ E \star S = 0,\ \forall S \in \mathcal{S}\}$$

De la même façon que plus haut, on détectera les erreurs par commutation ou non avec les éléments de $\mathcal{S}$, ce qui permet de travailler uniquement dans l'espace $\mathcal{E}_n$. Comme $\mathcal{S}$ est un sous-groupe, il suffit de tester la commutation avec les générateurs de $\mathcal{S}$:

Définition 27 (Syndrome)**.** *Soit $\mathcal{S} = \langle S_1, \dots S_r \rangle$ le stabilisateur d'un code $\mathscr{C}$, et E une erreur de Pauli. On appelle* syndrome de E *et on note* $\mathrm{s}(E)$ *le vecteur de bits suivant :*

$$\mathrm{s}(E) = (E \star S_1, \dots, E \star S_r)$$

On rappelle que $E_1 \star E_2$ vaut 0 si les erreurs commutent, et 1 sinon (voir définition 20).

Ainsi, $\mathrm{C}(\mathcal{S})$ est donc l'ensemble des erreurs de syndrome nul, ou c'est-à-dire l'ensemble des erreurs non détectées.

On peut noter que la définition du syndrome est spécifique au choix des générateurs, mais grâce à la distributivité de $\star$ (voir la propriété 6), ce choix a peu d'importance. En effet, si on utilise un autre choix de générateurs S'_j, alors chaque S'_j s'exprime comme un produit des S_i ; $E \star S'_i$ s'exprime comme une combinaison linéaire des $E \star S_i$. Autrement dit, pour peu qu'on connaisse la façon dont les S'_i s'expriment en fonction des S_i, on peut passer du syndrome défini par l'une des bases au syndrome défini par l'autre.

De même, il est parfois pratique de rajouter dans la définition du syndrome des stabilisateurs redondants (qui peuvent être obtenus par combinaison des autres). Comme un stabilisateur supplémentaire s'exprime comme une combinaison des autres S_i, le bit du syndrome associé s'écrit lui aussi comme une combinaison linéaire des $E \star S_i$: il n'y a pas de mesure supplémentaire à faire, et tout se passe comme si on n'avait pas ajouté ce stabilisateur supplémentaire.

On peut donc voir $\mathcal{S}$ comme une sorte de *matrice de parité* du code quantique (voir le schéma récaptitulatif 3.1) : soit $\mathbf{H}$ la matrice des S_i ligne par ligne (avec coefficients I, X, Y, Z), on peut écrire le syndrome $\mathrm{s}(E) = E \star {}^{\mathrm{t}}(\mathbf{H})$. $\mathbf{H}$ sera donc de dimension $r \times n$ si on a bien pris des générateurs indépendants, ou $r' \times n$ avec $r' > r$ si on a choisi de rajouter des générateurs supplémentaires. On peut voir la section 3.3.2.4 pour un exemple de telle matrice $\mathbf{H}$.

3.3.2.2 Distance minimale

Comme constaté plus haut, une erreur E peut :

- ne pas commuter avec le stabilisateur. $\mathrm{s}(E) \neq 0^r$: E est donc détectée, et on peut essayer de décoder cette erreur ;
- appartenir au stabilisateur : $E \in \mathcal{S}$. E est donc indétectée puisqu'elle commute avec le stabilisateur, mais par construction (voir définition 24) E n'affecte pas les états de $\mathscr{C}$. E est appelée une *erreur bénigne* ;
- être indétectée, mais sans appartenir au stabilisateur : $E \in \mathrm{C}(\mathcal{S})$, $(\mathrm{s}(E) = 0)$ mais $E \notin \mathcal{S}$. Cette erreur est indétectée, et envoie un état du code sur un *autre* état du code : $SE\,|\varphi\rangle = ES\,|\varphi\rangle = E\,|\varphi\rangle$, on a donc $E\,|\varphi\rangle \in \mathscr{C}$. E est appelée une *erreur sérieuse*.

Ces dernières erreurs sont donc les seules erreurs vraiment problématiques. De plus, lorsqu'on essaie de décoder l'erreur dans le premier cas, on peut se tromper à une erreur de syndrome nul près. Or s'il s'agit d'une erreur de $\mathcal{S}$, alors il n'y a pas de problèmes. En effet, si l'erreur E survient et qu'on croit avoir détecté $E' = ES$ pour $S \in \mathcal{S}$, alors on corrigera avec $E'^* = S^*E^*$, et on aura aura l'état $S^*E^*E\,|\varphi\rangle = S^*\,|\varphi\rangle = |\varphi\rangle$. Ainsi, on aura tout de même récupéré l'état d'origine.

De façon générale, tout le décodage se passe modulo le stabilisateur, et un syndrome n'est pas lié à une erreur mais à une classe d'erreurs équivalentes : c'est ce qu'on appelle la *dégénérescence*.

La distance minimale est donc le poids minimal d'une erreur de $\mathrm{C}(\mathcal{S})$ qui ne serait pas dans $\mathcal{S}$:

Définition 28 (Distance minimale)**.** *Soit $\mathscr{C}$ stabilisé par $\mathcal{S}$. La distance minimale de $\mathscr{C}$ d'exprime :*

$$d_{min}(\mathscr{C}) = \min\{\mathrm{w}(E), E \in C(\mathcal{S}), E \notin \mathcal{S}\}$$

Où $\mathrm{w}(E) = \#\{1 \leqslant i \leqslant n, E(i) \neq I\}$

Cette distance est plus grande que celle qu'on aurait pu définir naïvement, par comparaison avec les codes classiques, qui serait le poids minimal d'une erreur non détectée et non triviale :

Définition 29 (Distance minimale non dégénérée)**.** *Soit $\mathscr{C}$ stabilisé par $\mathcal{S}$. La distance minimale non dégénérée de $\mathscr{C}$ d'exprime :*

$$d'_{min}(\mathscr{C}) = \min\{\mathrm{w}(E), E \in C(\mathcal{S}), E \neq I^{\otimes n}\}$$

Comme on le verra plus tard (voir le chapitre 4 sur les codes LDPC quantiques), lors du décodage, il arrive souvent que ce soit cette distance qui soit prise en compte, car dans la patique, le décodeur ne sait pas toujours faire la différence entre les erreurs de $\mathcal{S}$ et les autres.

3.3.2.3 Décomposition de l'espace en base symplectique

Il est souvent utile d'avoir à sa disposition une décomposition de l'espace $\mathcal{E}_n$. Cela permet, en plus de clarifier les notions vues plus haut, d'avoir un critère de détection pour les erreurs de $\mathrm{C}(\mathcal{S})\backslash\mathcal{S}$. En effet, lorsqu'on cherche la distance minimale du code, il ne suffit pas de trouver une erreur non triviale indétectée : il faut encore vérifier que cette erreur n'est pas dans le stabilisateur.

Rappelons que le groupe $\mathcal{E}_n$ est, à la phase près, de taille 4^n : il est donc engendré (en tant que groupe multiplicatif) par $2n$ générateurs.

Propriété 9 (Base symplectique de l'espace $\mathcal{E}_n$ relative à un code stabilisateur)**.** *Soit $\mathcal{S} = \langle S_1, \dots S_r \rangle$ le stabilisateur d'un code $\mathscr{C}$, de dimension $k = n - r$. On peut écrire*

$$\mathcal{E}_n = \left\langle S_1, \dots S_r, T_1, \dots T_r, \bar{X}_1 \dots \bar{X}_k, \bar{Z}_1, \dots \bar{Z}_k \right\rangle$$

Avec :

$$\forall 1 \leqslant i, j \leqslant r, \qquad S_i \star T_j = \delta_{ij} \tag{3.3}$$
$$\forall 1 \leqslant i, j \leqslant r, \qquad T_i \star T_j = 0 \tag{3.4}$$
$$\forall 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant k, \quad S_i \star \bar{X}_j = 0 \text{ et } S_i \star \bar{Z}_j = 0 \tag{3.5}$$
$$\forall 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant k, \quad T_i \star \bar{X}_j = 0 \text{ et } T_i \star \bar{Z}_j = 0 \tag{3.6}$$
$$\forall 1 \leqslant i \leqslant k, \qquad \bar{X}_i \star \bar{Z}_i = \delta_{ij} \tag{3.7}$$
$$\forall 1 \leqslant i, j \leqslant k, \quad \bar{X}_i \star \bar{X}_j = 0, \ \bar{Z}_i \star \bar{Z}_j = 0 \tag{3.8}$$

Démonstration. On commence par établir le résultat lorsque le code est de dimension 0. Dans ce cas, on a alors $r = n$, et il n'y a pas de $\bar{X}_i$ ou de $\bar{Z}_i$ à construire, il suffit de construire les T_i de façon à vérifier les deux premières propriétés décrites ci-dessus.

On utilise le lemme 1, cela fournit une famille (T'_i) vérifiant $S_i \star T'_j = \delta_{ij}$: c'est la propriété (3.3). Il reste à faire en sorte de vérifier la propriété (3.4). Pour cela, on définit les (T_i) :

- $T_1 = T'_1$
- $T_i = T'_i \prod_{j \in I_i} S_j$, où $I_i = \{j < i, T_j \star T'_i = 1\}$

Ainsi, on a toujours (3.3) : $S_i \star T_j = S_i \star T'_j = \delta_{ij}$, et également, pour $i < j$:

$$\begin{aligned}
T_i \star T_j &= T_i \star (T'_j \prod_{\{k<j, T_k \star T'_j = 1\}} S_k) \\
&= T_i \star T'_j + \sum_{\{k<j, T_k \star T'_j = 1\}} T_i \star S_k \\
&= T_i \star T'_j + \begin{cases} T_i \star S_i & \text{si } i \text{ vérifie } T_i \star T'_j = 1 \\ 0 & \text{sinon} \end{cases} \\
&= 0
\end{aligned}$$

On a donc bien (3.4).

Si la dimension du code n'est pas 0, alors on peut définir $\bar{X}_1$ tel que $\bar{X}_1 \in \mathrm{C}(\mathcal{S}) \backslash \mathcal{S}$. Comme il commute avec tous les S_i par définition, on peut écrire $\mathcal{S}' = \left\langle S_1, \dots S_r, \bar{X}_1 \right\rangle$, il s'agit du stabilisateur d'un code de dimension $r - 1$. On peut donc réitérer l'opération tant que la dimension du code est strictement positive, et on a ainsi construit $\bar{X}_1, \dots \bar{X}_k$ qui commutent entre eux (puisque chaque $\bar{X}_i$ ajouté est dans le centralisateur de $\left\langle \mathcal{S}, \bar{X}_1, \dots \bar{X}_{i-1} \right\rangle$). On a donc un code stabilisateur $\mathcal{S}_c$

de dimension 0, et on applique la construction précédente pour trouver des T_i qui conviennent : on les nomme, dans l'ordre, $T_1, \ldots, T_r, \bar{Z}_1, \ldots, \bar{Z}_k$.

Les S_i et T_i vérifient bien les équations (3.4) et (3.3). Les $\bar{X}_i$ commutent entre eux et avec les S_i par construction, et avec tous les T_j et les $\bar{Z}_j$ sauf $\bar{Z}_i$ grâce aux équations (3.3) appliquées à $\mathcal{S}_c$. De même, avec (3.4), les $\bar{Z}_i$ commutent entre eux et avec les T_j. On a donc toutes les conditions requises pour la base symplectique. ■

Dans cette décomposition, les T_i sont les « porteurs » du syndrome. Les $\bar{X}_i$ et $\bar{Z}_i$ sont des erreurs qui ne sont pas engendrées par les T_i (donc indétectées) et les S_i (donc n'appartiennent pas à $\mathcal{S}$). Ce sont donc les erreurs dites sérieuses : la construction en donne une base.

Si, de plus, on connaît cette base d'erreur, cela donne un critère pour savoir dans quelle catégorie elle se situe : on vérifie d'abord si elle est détectée ou non en testant la commutation avec les S_i. Si elle est indétectée, on vérifie si elle commute avec tous les $\bar{X}_i$ et les $\bar{Z}_i$. Si oui, on a affaire à une erreur du stabilisateur (donc bénigne). Si non, il s'agit d'une erreur sérieuse, et on peut même donner sa décomposition :

Propriété 10. *Soit $\mathcal{S}$ le stabilisateur d'un code $\mathscr{C}$, et E une erreur indétectée. Alors on peut décomposer E :*

$$E = \prod_{i=0}^{k} \bar{X}_i^{x_i} \times \prod_{i=0}^{k} \bar{Z}_i^{z_i} \times S$$

où

- $S \in \mathcal{S}$
- $x_i = E \star \bar{Z}_i$
- $z_i = E \star \bar{X}_i$

Démonstration. Les $S_i, \bar{X}_i, \bar{Z}_i$ étant une base de l'espace $\mathrm{C}(\mathcal{S})$, une telle décomposition existe. Il reste à vérifier les égalités sur les x_i et z_i :

$$\begin{aligned} E \star \bar{Z}_j &= \prod_{i=0}^{k} \bar{X}_i^{x_i} \times \prod_{i=0}^{k} \bar{Z}_i^{z_i} \times S \star \bar{Z}_j \\ &= \sum_{i=0}^{k} \bar{X}_i^{x_i} \star \bar{Z}_j + \sum_{i=0}^{k} \bar{Z}_i^{z_i} \star \bar{Z}_j + S \star \bar{Z}_j \\ &= x_i \end{aligned}$$

car $\bar{Z}_i \star \bar{Z}_j = 0$, $S \star \bar{S} = 0$, et $\bar{X}_i \star \bar{Z}_j = 0$ si $i \neq j$ et 1 sinon. Le calcul est similaire pour les z_i.

■

On peut noter que la notation $\bar{X}_i$ ne signifie pas forcément que l'erreur $\bar{X}_i$ soit composée uniquement de X, de même pour $\bar{Z}_j$, ou même qu'il soit possible de trouver des erreurs sous cette forme, mais ce sera le cas pour les codes CSS (voir la section 3.4 sur les codes CSS). De plus, cette notation rappelle celle utilisée précédemment pour les Z_i : X_i serait un vecteur contenant l'identité, sauf X en position i, de même pour Z_i. Ces vecteurs vérifient alors les propriétés des $\bar{X}_i$ et $\bar{Z}_i$ du point de vue des commutations, et forment une base de l'espace $\mathcal{E}_n$.

On peut également remarquer qu'il n'est pas nécessaire de connaître toute la base symplectique pour savoir si une erreur indétectée E n'est pas dans le stabilisateur : une base des $\bar{X}_i, \bar{Z}_i$ suffit. Et même sans connaître cette base, si on trouve une autre erreur indétectée E' qui ne commute pas avec E, alors E et E' sont dans $\mathrm{C}(\mathcal{S})\backslash\mathcal{S}$. C'est une propriété évidente mais qui est très utile : lorsqu'on cherche une borne supérieure sur la distance minimale d'un code stabilisateur, on peut trouver des erreurs indétectées d'un certain poids, il suffit alors de trouver une autre erreur indétectée (peu importe son poids) qui ne commute pas avec celle-ci pour prouver qu'il s'agit d'une erreur sérieuse.

3.3.2.4 Un exemple

On définit le code sur 4 qubits, de dimension 1, avec la matrice de stabilisateurs suivante :

$$\mathbf{H} = \begin{pmatrix} X & Y & Z & I \\ Y & Z & I & X \\ Z & I & X & Y \end{pmatrix}$$

Cela revient à écrire $\mathcal{S} = \langle [XYZI], [YZIX], [ZIXY] \rangle$. On voit que le groupe est bien commutatif : deux lignes anti-commutent toujours en exactement deux positions (positions 1 et 2 pour S_1 et S_2, positions 1 et 3 pour S_1 et S_3, et positions 1 et 4 pour S_2 et S_3).

Les T_i associés sont donc :
- $T_1 = [IZII]$
- $T_2 = [IIIY]$
- $T_3 = [IIZI]$

On constate en effet que T_1 anti-commute avec S_1, mais commute avec S_2 et S_3, et de même pour les autres T_i. De plus, les T_i commutent entre eux.

Enfin on peut écrire $\bar{X}_1$ et $\bar{Z}_1$:

$$\bar{X}_1 = [IYIY], \qquad \bar{Z}_1 = [ZZIY]$$

On peut vérifier que $\bar{X}_1 \star \bar{Z}_i = 1$, et que ces erreurs commutent avec tous les S_i et tous les T_i.

3.4 Codes CSS

Pour simplifier encore le modèle, on s'intéresse ici à un cas où les S_i qui engendrent le stabilisateur sont composés soit uniquement de I et de X, soit uniquement de I et de Z. On obtient alors un code dit *CSS*. L'appellation vient de Calderbank, Shor et Steane, qui ont introduit ces codes pour la première fois, dans [CS96, Ste96].

Comme expliqué plus bas, un code CSS peut être vu comme une paire de codes classiques (avec quelques conditions particulières), pour lesquels on sait beaucoup plus de choses. On peut ainsi utiliser les outils des codes classiques pour ce type de code, entre autres, on peut utiliser les algorithmes de décodage classiques.

3.4.1 Définition via les codes stabilisateurs

Définition 30 (Code CSS). *Soit $\mathcal{S}$ le stabilisateur d'un code $\mathscr{C}$. $\mathscr{C}$ est un* code CSS *si $\mathcal{S}$ peut s'écrire :*

$$\mathcal{S} = \langle S_{X1}, \ldots, S_{Xr_X}, S_{Z1}, \ldots, S_{Zr_Z} \rangle$$

avec $S_{Xi} \in \{I, X\}^{\otimes n}$ et $S_{Zi} \in \{I, Z\}^{\otimes n}$. On notera $\mathcal{S}_X = \langle S_{X1}, \ldots S_{Xr_X} \rangle$, $\mathcal{S}_Z = \langle S_{Z1}, \ldots S_{Zr_Z} \rangle$, et $\mathcal{S} = \langle \mathcal{S}_X, \mathcal{S}_Z \rangle$.

On peut alors, pour toute erreur $E \in \mathcal{E}_n$, la décomposer en un produit de deux erreurs, composées de X et de Z respectivement : $E = E_X E_Z$, où $E_X \in \{I, X\}^{\otimes n}$, et $E_Z \in \{I, Z\}^{\otimes n}$. La condition de commutation avec les éléments de $\mathcal{S}$ peut alors s'exprimer plus simplement : comme E_X commute avec les $\mathcal{S}_X$, il n'y a qu'à vérifier la commutation avec les $\mathcal{S}_Z$, et inversement. De même, vérifier si E est dans $\mathcal{S}$ est équivalent à vérifier $E_X \in \mathcal{S}_X$ et $E_Z \in \mathcal{S}_Z$.

Mieux, si on considère les mots *binaires* w_X et w_Z associés à E_X et E_Z, alors toutes les propriétés du code $\mathscr{C}$ vont s'exprimer comme des propriétés analogues à celles de code binaires. Pour cela, il faut décomposer les erreurs de $\mathcal{E}_n$ en mots binaires :

Définition 31 (Équivalence binaire - erreur de Pauli). *Soit $E \in \mathcal{E}_n$. On note $[E]^{\mathbf{x}}$ le mot binaire tel que :*

$$([E]^{\mathbf{x}})_i = 0 \text{ si } E(i) = I \text{ ou } Z$$
$$([E]^{\mathbf{x}})_i = 1 \text{ si } E(i) = X \text{ ou } Y$$

De même, on note $[E]^{\mathbf{z}}$ le mot binaire tel que :

$$([E]^{\mathbf{z}})_i = 0 \text{ si } E(i) = I \text{ ou } X$$
$$([E]^{\mathbf{z}})_i = 1 \text{ si } E(i) = Z \text{ ou } Y$$

Soient deux mots binaires w_X et w_Z. On écrit $[w_X, w_Z]^{\mathcal{E}}$ l'erreur de Pauli telle que :

$$[w_X, w_Z]^{\mathcal{E}} = X^{w_X} Z^{w_Z}$$

à la phase près.

Naturellement, cette notation sert à passer de $\mathcal{E}_n$ aux mots binaires :

Propriété 11. *Si $E \in \mathcal{E}_n$, alors on a*

$$\left[[E]^{\mathbf{X}}, [E]^{\mathbf{Z}}\right]^{\mathcal{E}} = E$$

Et inversement, si $w_X, w_Z \in \{0,1\}^n$, alors

$$\left[[w_X, w_Z]^{\mathcal{E}}\right]^{\mathbf{X}} = w_X, \qquad \left[[w_X, w_Z]^{\mathcal{E}}\right]^{\mathbf{Z}} = w_Z$$

Le produit $\star$ s'exprime aussi avec les mots binaires associés :

Propriété 12. *Soient $E_1, E_2 \in \mathcal{E}_n$.*

$$E_1 \star E_2 = [E_1]^{\mathbf{X}\ \mathbf{t}}\big([E_2]^{\mathbf{Z}}\big) + [E_2]^{\mathbf{X}\ \mathbf{t}}\big([E_1]^{\mathbf{Z}}\big) \mod 2$$

Démonstration.

$$\begin{aligned} E_1 \star E_2 &= \left[[E_1]^{\mathbf{X}}, [E_2]^{\mathbf{Z}}\right]^{\mathcal{E}} \star \left[[E_2]^{\mathbf{X}}, [E_2]^{\mathbf{Z}}\right]^{\mathcal{E}} \\ &= X^{[E_1]^{\mathbf{X}}} Z^{[E_1]^{\mathbf{X}}} \star X^{[E_2]^{\mathbf{X}}} Z^{[E_2]^{\mathbf{X}}} \\ &= X^{[E_1]^{\mathbf{X}}} \star Z^{[E_2]^{\mathbf{X}}} + X^{[E_2]^{\mathbf{X}}} Z^{[E_1]^{\mathbf{X}}} \\ &= [E_1]^{\mathbf{X}\ \mathbf{t}}\big([E_2]^{\mathbf{Z}}\big) + [E_2]^{\mathbf{X}\ \mathbf{t}}\big([E_1]^{\mathbf{Z}}\big) \end{aligned}$$

■

On peut désormais exprimer toutes les propriétés d'un code quantique avec cette représentation binaire :

Définition 32 (Codes CSS, partie en X/en Z)**.** *Soit $\mathcal{S} = \langle \mathcal{S}_X, \mathcal{S}_Z \rangle$ le stabilisateur d'un code CSS $\mathscr{C}$. On note $[\mathscr{C}]_X$ (respectivement $[\mathscr{C}]_Z$) le code correcteur* classique *dont les lignes de la matrice de parité sont données par les $[S_{Xi}]^{\mathbf{X}}$ (respectivement les $[S_{Zi}]^{\mathbf{Z}}$) :*

$$[\mathscr{C}]_X = \{w \in \{0,1\}^n, \mathbf{H}_X\ ^{\mathbf{t}}(w) = 0\}$$
$$[\mathscr{C}]_Z = \{w \in \{0,1\}^n, \mathbf{H}_Z\ ^{\mathbf{t}}(w) = 0\}$$

où $\mathbf{H}_X$ *et* $\mathbf{H}_Z$ *sont les matrices suivantes :*

$$\mathbf{H}_X = \begin{pmatrix} [S_{X1}]^{\mathbf{X}} \\ \dots \\ [S_{Xr_X}]^{\mathbf{X}} \end{pmatrix}, \qquad \mathbf{H}_Z = \begin{pmatrix} [S_{Z1}]^{\mathbf{X}} \\ \dots \\ [S_{Zr_Z}]^{\mathbf{X}} \end{pmatrix}$$

Ces matrices sont de tailles respectives $r_X \times n$ et $r_Z \times n$, et n'ont pas forcément besoin d'être de rang plein. En revanche, il y a une condition qui doit être respectée, et qui vient du fait que $\mathcal{S}$ doit être commutatif :

Propriété 13 (Orthogonalité)**.** *Soit* $\mathscr{C}$ *un code CSS, et* $\mathbf{H}_X$ *et* $\mathbf{H}_Z$ *les matrices de parité respectives de* $[\mathscr{C}]_X$ *et* $[\mathscr{C}]_Z$*. On a :*

$$\mathbf{H}_X \, {}^{\mathbf{t}}(\mathbf{H}_Z) = 0 \tag{3.9}$$

Démonstration. Le stabilisateur $\mathcal{S}$ est comutatif, ce qui signifie que, pour tous $1 \leqslant i \leqslant r_x$, $1 \leqslant j \leqslant r_z$, on a $S_{Xi} \star S_{Zj} = 0$. Autrement dit, avec l'équivalence binaire :

$$\begin{aligned} 0 &= S_{Xi} \star S_{Zj} \\ &= [S_{Xi}]^{\mathbf{X}\,\mathbf{t}}([S_{Zj}]^{\mathbf{Z}}) + [S_{Xi}]^{\mathbf{Z}\,\mathbf{t}}([S_{Zj}]^{\mathbf{X}}) \\ &= [S_{Xi}]^{\mathbf{X}\,\mathbf{t}}([S_{Zj}]^{\mathbf{Z}}) \end{aligned}$$

Puisque $[S_{Xi}]^{\mathbf{Z}} = [S_{Zj}]^{\mathbf{X}} = 0^n$. Toutes les lignes des matrices $\mathbf{H}_X$ et $\mathbf{H}_Z$ sont donc orthogonales entre elles, ce qui donne l'équation (3.9).

■

3.4.2 Caractérisation binaire des codes CSS

Toutes les propriétés des codes CSS peuvent s'exprimer uniquement dans $\{0, 1\}$, on peut donc redéfinir un code CSS directement par sa paire de matrices de parité classiques, et donner ses propriétés en fonction de ces matrices classiques.

Non seulement cette caractérisation permet à des lecteurs peu familiers avec les codes quantiques de tout de même comprendre le fonctionnement les codes CSS (sans lire les preuves), mais surtout cette nouvelle façon de voir les codes quantiques permet d'utiliser tous les outils connus dans le monde des codes classiques, qu'il s'agisse de décodage (on le verra évidemment avec les codes LDPC, voir la section 1.3 pour la version clssique et le chapitre 4 pour la version quantique), de construction de « bons » codes, de limites sur la capacité de correction, etc. Quand bien même les outils ne seraient pas pertinents pour les codes quantiques, le parallèle permet tout de même de comparer les performances de codes classiques et quantiques, et donner ainsi une idée de ce qu'il est possible de faire.

Définition 33 (Code CSS, version classique)**.** *Un code CSS $\mathscr{C}$ de longueur n est la donnée de deux matrices binaires $r_X \times n$ et $r_Z \times n$ respectives, vérifiant :*

$$\mathbf{H}_X \, {}^{\mathbf{t}}(\mathbf{H}_Z) = 0 \tag{3.10}$$

On note $[\mathscr{C}]_X$ et $[\mathscr{C}]_Z$ les codes correcteurs classiques associés à $\mathbf{H}_X$ et à $\mathbf{H}_Z$ respectivement, c'est-à-dire :

$$[\mathscr{C}]_X = \{w \in \{0,1\}^n, \mathbf{H}_X \, {}^{\mathbf{t}}(w) = 0\}$$
$$[\mathscr{C}]_Z = \{w \in \{0,1\}^n, \mathbf{H}_Z \, {}^{\mathbf{t}}(w) = 0\}$$

On peut maintenant donner ses propriétés principales : dimension, distance minimale, et les erreurs dites sérieuses, qu'on appellera parfois « mots de codes » par référence aux codes classiques.

Propriété 14 (Dimension d'un code CSS)**.** *Soit $\mathscr{C}$ un code CSS défini par $\mathbf{H}_X$ et $\mathbf{H}_Z$. Alors la dimension du code s'exprime :*

$$\dim(\mathscr{C}) = n - \mathrm{rang}(\mathbf{H}_X) - \mathrm{rang}(\mathbf{H}_Z) \tag{3.11}$$

Démonstration. Il suffit de voir que le rang de $\mathbf{H}_X$ correspond au nombre de $[S_{Xi}]^{\mathbf{x}}$ indépendants, qui correspond au nombre de S_{Xi} indépendants, et de même pour $\mathbf{H}_Z$. On a donc

$$\begin{aligned} \dim(\mathscr{C}) &= n - \text{nombre de stabilisateurs indépendants} \\ &= n - (\mathrm{rang}(\mathbf{H}_X) + \mathrm{rang}(\mathbf{H}_Z)). \end{aligned}$$

■

On peut voir assez rapidement que l'équation (3.10) fournit des mots de code naturels pour $[\mathscr{C}]_X$ et $[\mathscr{C}]_Z$: chaque ligne de $\mathbf{H}_Z$ est un mot de code de $[\mathscr{C}]_X$, et inversement. Cependant, comme on l'a vu dans la section sur les codes stabilisateurs, certaines erreurs sont bénignes et ne modifient pas le système : ce sont précisément ces erreurs-là.

Propriété 15 (Erreurs sérieuses d'un code CSS)**.** *Les erreurs sérieuses d'un code CSS $\mathscr{C}$ sont caractérisées par un couple de mots binaires (w_1, w_2) tel que :*

$$\mathbf{H}_X \, {}^{\mathbf{t}}(w_1) = 0 \tag{3.12}$$
$$\mathbf{H}_Z \, {}^{\mathbf{t}}(w_2) = 0 \tag{3.13}$$
$$w_1 \notin \mathrm{VectLi}(\mathbf{H}_Z) \quad ou \quad w_2 \notin \mathrm{VectLi}(\mathbf{H}_X) \tag{3.14}$$

où $\mathrm{VectLi}(A)$ *pour une matrice A désigne l'espace engendré par les lignes de la matrice A.*

Démonstration. Pour un code stabilisateur, une erreur sérieuse est une erreur $E \in \mathcal{E}_n$ qui vérifie $E \in \mathrm{C}(\mathcal{S})$ et $E \notin \mathcal{S}$. On pose $w_1 = [E]^{\mathbf{Z}}$ et $w_2 = [E]^{\mathbf{X}}$. On décompose $E = E_X E_Z$ où $E_X = X^{[E]^{\mathbf{X}}}$ et $E_Z = Z^{[E]^{\mathbf{Z}}}$. On a donc, pour la première condition :

$$\begin{aligned} 0 &= E \star S_{Xi} \\ 0 &= E_X E_Z \star S_{Xi} \\ 0 &= E_Z \star S_{Xi} \end{aligned}$$

Ce qui se traduit en binaire par $w_1 \, {}^{\mathbf{t}}\big([S_{Xi}]^{\mathbf{X}}\big) = 0$. C'est vrai pour tout $1 \leqslant i \leqslant r_X$, donc on a 3.12. Par le même raisonnement en inversant X et Z, on obtient 3.13.

La condition $E \notin \mathcal{S}$ implique qu'au moins l'une des deux erreurs E_X ou E_Z ne soit pas dans $\mathcal{S}$: si les deux étaient dans $\mathcal{S}$, alors E serait dans $\mathcal{S}$. S'il s'agit de E_Z, alors E_Z n'est pas engendrée par les S_{Zi} (E_Z étant composée d'erreurs en Z, elle ne peut pas être générée par les S_{Xi}). Cela se traduit donc en binaire par $w_1 \notin \mathrm{Vect}([S_{Z1}]^{\mathbf{Z}}, \dots [S_{Zr_Z}]^{\mathbf{Z}})$, qui est exactement $\mathrm{VectLi}(\mathbf{H}_Z)$, d'où la première partie de la propriété 3.14. La seconde vient du cas où il s'agit d'E_X.

De plus, si les deux erreurs E_X et E_Z ne sont pas engendrées par $\mathcal{S}$, alors le produit ne l'est pas non plus : s'il l'était, alors il serait une combinaison des S_{Xi} et des S_{Zi}, ce qui signifie qu'E_X serait la combinaison des S_{Xi} correspondante et E_Z serait celle des S_{Zi}.

■

À partir de là, la distance minimale est donc le poids minimal d'une erreur sérieuse :

Propriété 16 (Distance minimale des codes CSS)**.** *Soit $\mathscr{C}$ un code CSS défini par $\mathbf{H}_X$ et $\mathbf{H}_Z$. La distance minimale de $\mathscr{C}$ s'exprime :*

$$d_{min} = \min\{d_X, d_Z\} \tag{3.15}$$

où

$$d_X = \min\{\mathrm{w}(w), w \in \{0,1\}^n, w \in [\mathscr{C}]_X, w \notin \mathrm{VectLi}(\mathbf{H}_Z)\} \tag{3.16}$$
$$d_Z = \min\{\mathrm{w}(w), w \in \{0,1\}^n, w \in [\mathscr{C}]_Z, w \notin \mathrm{VectLi}(\mathbf{H}_X)\} \tag{3.17}$$

Démonstration. Vues les propriétés précédentes, il suffit de montrer qu'une erreur sérieuse de poids minimal est atteinte par une erreur composée uniquement de X ou de Z.

Soit $E \in \mathcal{E}_n$ telle que $E \in \mathrm{C}(\mathcal{S})$ et $E \notin \mathcal{S}$. On commence par écrire $E = E_X E_Z$, comme dans la preuve de la propriété précédente. On sait que les deux erreurs sont dans $\mathrm{C}(\mathcal{S})$, et qu'au moins une d'entre elles n'est pas dans $\mathcal{S}$, disons E_X. Comme

$\mathrm{w}(E_X) \leqslant \mathrm{w}(E)$, si E atteignait la distance minimale, E_X aussi (et de même en Z). Ainsi, la distance minimale est atteinte sur les erreurs composées uniquement de X ou de Z.

■

La majoration de la distance minimale se fait généralement en exhibant un mot binaire w, d'un certain poids, qui soit dans $[\mathscr{C}]_X$ mais pas dans $\mathrm{VectLi}(\mathbf{H}_Z)$ (ou l'inverse). Pour un tel mot, il est facile de vérifier qu'il est dans $[\mathscr{C}]_X$ (il suffit de le multiplier par la matrice de parité $\mathbf{H}_X$), mais il n'est pas forcément évident de montrer que w n'est pas dans $\mathrm{VectLi}(\mathbf{H}_Z)$. Pour cela, on peut s'aider de la propriété suivante :

Propriété 17. *Soit $\mathscr{C}$ un code CSS défini par $\mathbf{H}_X$ et $\mathbf{H}_Z$, et deux mots binaires w_1 et w_2 tels que $w_1 \in [\mathscr{C}]_X$ et $w_2 \in [\mathscr{C}]_Z$. Alors si $w_1 \, {}^{\mathrm{t}}(w_2) = 1$, on a*

$$w_1 \notin \mathrm{VectLi}(\mathbf{H}_Z)$$
$$w_2 \notin \mathrm{VectLi}(\mathbf{H}_X)$$

Autrement dit, w_1 et w_2 sont des erreurs sérieuses. Pour majorer la distance minimale, il suffit donc, si on possède un bon candidat de petit poids w_1 pour $[\mathscr{C}]_X$, de trouver un mot du code de $[\mathscr{C}]_Z$ qui ne soit pas orthogonal à w_1.

Démonstration. On construit les erreurs $E_1 = Z^{w_1}$ et $E_2 = X^{w_2}$. Clairement, ces erreurs sont indétectées, et $E_1 \star E_2 = 1$. E_1 et E_2 ne sont donc pas dans le stabilisateur, ce qui signifie que $w_1 \notin \mathrm{VectLi}(\mathbf{H}_Z)$ et $w_2 \notin \mathrm{VectLi}(\mathbf{H}_X)$.

■

Tout comme dans la section 3.3.2.3, on va chercher à décomposer l'espace en une base symplectique, simple à manipuler.

Propriété 18 (Décomposition en base symplectique)**.** *Soit un code CSS défini par $\mathbf{H}_X$ et $\mathbf{H}_Z$, de dimension k. On note $r_X = \mathrm{rang}(\mathbf{H}_X)$ et $r_Z = \mathrm{rang}(\mathbf{H}_Z)$. Il existe deux bases de $\{0,1\}^n$ de la forme : $(s_{X1}, \dots s_{Xr_X}, t_{X1}, \dots t_{Xr_Z}, \bar{x}_1, \dots \bar{x}_k)$ et $(s_{Z_1}, \dots s_{Zr_Z}, t_{Z_1}, \dots t_{Zr_X}, \bar{z}_1, \dots \bar{z}_k)$ telles que :*

- *les s_{Xi} soient les lignes indépendantes de la matrice $\mathbf{H}_X$,*
- *les s_{Zi} soient les lignes indépendantes de la matrice $\mathbf{H}_Z$,*

$$\forall 1 \leqslant i,j \leqslant r_X, \quad s_{Xi}\,{}^{\mathbf{t}}(t_{Zj}) = \delta_{i,j} \tag{3.18}$$
$$\forall 1 \leqslant i,j \leqslant r_Z, \quad s_{Zi}\,{}^{\mathbf{t}}(t_{Xj}) = \delta_{i,j} \tag{3.19}$$
$$\forall 1 \leqslant i,j \leqslant r_k, \quad \bar{x}_i\,{}^{\mathbf{t}}(\bar{z}_j) = \delta_{i,j} \tag{3.20}$$
$$\forall 1 \leqslant i \leqslant k, \forall 1 \leqslant j \leqslant r_Z, \quad \bar{x}_i\,{}^{\mathbf{t}}(s_{Zj}) = 0 \tag{3.21}$$
$$\forall 1 \leqslant i \leqslant k, \forall 1 \leqslant j \leqslant r_X, \quad \bar{x}_i\,{}^{\mathbf{t}}(t_{Zj}) = 0 \tag{3.22}$$
$$\forall 1 \leqslant i \leqslant k, \forall 1 \leqslant j \leqslant r_X, \quad \bar{z}_i\,{}^{\mathbf{t}}(s_{Xj}) = 0 \tag{3.23}$$
$$\forall 1 \leqslant i \leqslant k, \forall 1 \leqslant j \leqslant r_Z, \quad \bar{z}_i\,{}^{\mathbf{t}}(t_{Xj}) = 0 \tag{3.24}$$
$$\forall i \leqslant i \leqslant r_X, \forall 1 \leqslant j \leqslant r_Z, \quad t_{Xi}\,{}^{\mathbf{t}}(t_{Zj}) = 0 \tag{3.25}$$

Les $\bar{x}_i$ et $\bar{z}_i$ forment alors des bases sur $\{0,1\}^n$ des erreurs sérieuses du code.

Démonstration. Il s'agit de la même construction qu'en section 3.3.2.3, sauf qu'on peut exprimer les $\bar{X}_i$ comme des mots sur X uniquement, et les $\bar{Z}_i$ sur Z uniquement. On pourrait faire la construction à partir de cette base générique, mais il est en réalité plus simple de la construire directement.

On commence par écrire les s_{Xi} et s_{Zi}, avec des lignes indépendantes des matrices $\mathbf{H}_X$ et $\mathbf{H}_Z$. Ensuite, on sait que $\mathrm{Im}(\mathbf{H}_X)$ est de dimension r_X : si on peut générer tous les mots de longueur r_X par multiplication de mots de $\{0,1\}^n$ et de $\mathbf{H}_X$, alors il existe r_X vecteurs $t_{Z1}, \dots t_{Zr_X}$ qui vérifient $\mathbf{H}_X\,{}^{\mathbf{t}}(t_{Zi}) = (0,\dots,1,0,\dots 1)$, avec un 1 en position i. Ainsi, on peut avoir des t_{Zi} vérifiant la propriété (3.18). On peut faire de même pour les t_{Xi}, et on obtient (3.19).

Pour obtenir la condition (3.25), si un t_{Zi} donné n'est pas orthogonal à un t_{Xj}, on peut ajouter s_{Zj} à t_{Zi} : le syndrome reste le même puisqu'on a ajouté une ligne de $\mathbf{H}_Z$ (qui est orthogonale à $\mathbf{H}_X$), et on a $(t_{Zj} + t_{Zi})\,{}^{\mathbf{t}}(t_{Xj}) = 1 + 1 = 0$.

On a donc deux bases partielles de $\{0,1\}^n$: (s_{Xi}, t_{Xi}) et (s_{Zi}, t_{Zi}), qui vérifient les propriétés voulues (on sait que les t_{Xi} ne peuvent pas être engendrés par les s_{Xi} puisque les s_{Xi} sont orthogonaux aux s_{Zi} et pas les t_{Xi} par construction).

On peut alors compléter les bases pour obtenir k nouveaux vecteurs pour chacune (car $k = n - r_X - r_Z$), $\bar{x}_i$ et $\bar{z}_i$. On peut supposer que les $\bar{x}_i$ sont orthogonaux aux s_{Zj}, quitte à leur ajouter des t_{Xj} correspondants. On peut également les supposer orthogonaux aux t_{Zj} pour les mêmes raisons, et de même en symétrique pour les $\bar{z}_i$: on a alors (3.21), (3.22), (3.23) et (3.24).

Il ne reste plus qu'à faire en sorte de vérifier (3.20). Or pour $\bar{x}_1$, il existe forcément un vecteur parmi les $\bar{z}_i$ qui n'est pas orthogonal à $\bar{x}_1$: si c'était le cas, comme $\bar{x}'_1$ est orthogonal aux s_{Zi} et aux t_{Zi}, il serait orthogonal à tout l'espace $\{0,1\}^n$. On appelle donc ce vecteur $\bar{z}_1$. On ajoute ensuite $\bar{z}_1$ à tous les autres $\bar{z}_i$ qui ne seraient pas orthogonaux à $\bar{x}_1$, de façon à ce qu'ils le soient, et on ajoute également $\bar{x}_1$ à tous les autres $\bar{x}_i$ qui ne seraient pas orthogonaux à $\bar{z}_1$: on a toujours une base du même sous-espace, vérifiant les mêmes propriétés d'orthogonalité qu'au

début. On peut donc réitérer l'opération, et avoir les $\bar{x}_i$, $\bar{z}_i$ souhaités.

■

Dans la preuve, on constate qu'il n'est nécessaire que d'effectuer des opérations linéaires sur des mots binaires de longueur n, ainsi, on peut construire une telle base en un temps polynômial si $\mathbf{H}_X$ et $\mathbf{H}_Z$ sont données. Dans la pratique, comme on a peu besoin des t_{Xi}, t_{Zi}, on se contente de construire les $\bar{x}_i$ et les $\bar{z}_i$, qui permettent de savoir si, pour un mot donné de syndrome nul, il s'agit d'un mot du stabilisateur ou non.

En définitive, corriger une erreur dans un code CSS revient à décoder *deux* codes classiques, $[\mathscr{C}]_X$ et $[\mathscr{C}]_Z$, modulo un sous-espace. En pratique, il est parfois difficile de décoder modulo le sous-espace, on peut alors effectuer le décodage « brut », sans tenir compte des erreurs bénignes. On définira donc la *distance minimale non dégénérée* :

Définition 34 (Distance minimale non dégérénée)**.** *Soit $\mathscr{C}$ un code CSS. La distance minimale non dénégérée est le minimum des distances minimales des deux codes classiques le composant :*

$$d'_{min} = \min\{d'_X, d'_Y\} \tag{3.26}$$

où

$$d'_X = \min\{\mathrm{w}(w), w \in \{0,1\}^n, w \in [\mathscr{C}]_X, w \neq 0^n\} \tag{3.27}$$
$$d'_Z = \min\{\mathrm{w}(w), w \in \{0,1\}^n, w \in [\mathscr{C}]_Z, w \neq 0^n\} \tag{3.28}$$

Enfin, pour tout ramener au classique, il faut avoir un canal équivalent :

Propriété 19 (Canal binaire symétrique équivalent)**.** *Si $\mathscr{C}$ est un code CSS utilisé sur un canal dépolarisant de probabilité p, et que le décodage de $[\mathscr{C}]_X$ est sensiblement identique à celui de $[\mathscr{C}]_Z$, on peut le simuler par le décodage de $[\mathscr{C}]_X$ sur un canal binaire symétrique de probabilité $p' = \frac{2p}{3}$. La probabilité d'erreur résultante est alors $p_{rq} = 2p_{rc} - p_{rc}^2$, où p_{rc} est la probabilité d'erreur résultante de ce décodage classique.*

Démonstration. Pour décoder correctement une erreur E, il faut décoder correctement les erreurs classiques associées $[E]^{\mathbf{x}}$ et $[E]^{\mathbf{z}}$. Si on a la probabilité p_{rc} de décoder correctement l'un des deux, par symétrie l'autre est égale à p_{rc}, et on a la probabilité finale : $p_{rq} = 1 - (1 - p_{rc})^2 = 2p_{resc} - p_{rc}$.

Ensuite, $[E]^{\mathbf{x}}{}_i = 1$ si et seulement si $E(i) = X$ ou Y, autrement dit, sur le canal dépolarisant avec une probabilité $2p/3$.

■

Pour décoder un code CSS, il suffit donc d'être capable de générer deux codes classiques vérifiant (3.10), et de savoir les décoder correctement. Mais cette condition est assez forte et il n'est pas facile d'obtenir deux bons codes vérifiant cette propriété.

3.4.3 Exemple

Soit le code CSS défini par les matrices H_X et H_Z suivantes :

$$H_X = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \qquad H_Z = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

On peut vérifier rapidement que $H_X \, {}^{\mathbf{t}}(H_Z) = 0$. Il n'y a pas de ligne redondante dans les matrices, donc la dimension est $5 - 4 = 1$. Il faut donc exhiber deux erreurs sérieuses, qui vérifient la propriété 17. On choisit $w_1 = 11000$: $w_1 \in [\mathscr{C}]_X$. On prend ensuite $w_2 = 10010 \in [\mathscr{C}]_Z$. On constate que $w_1 \, {}^{\mathbf{t}}(w_2) = 1$: ce sont donc deux erreurs sérieuses.

3.5 Schéma récapitulatif

Le tableau 3.1 présente un petit résumé des différentes propriétés des codes classiques, stabilisateurs et CSS. Le petit graphique représente le graphe de Tanner, et sa version pour les codes stabilisateurs est introduite dans le chapitre suivant, dans la section 4.1.1.

3.6 Borne de hachage et capacité de correction

Comme dans le cas classique, on aimerait pouvoir donner des bornes sur la capacité de correction des codes stabilisateurs : jusqu'à quelle probabilité d'erreur peut-on corriger (sur un certain canal), et pour quel rendement. Malheureusement, dans le cas quantique, il n'y a pas autant de résultats que dans le cas classique. En effet, s'il existe bien une notion de capacité de canal quantique, cette capacité est inconnue sauf dans quelques cas de canaux particuliers (canal à effacements par exemple), entre autres, pour le canal dépolarisant la capacité n'est pas connue.

Définition 35 (Capacité d'un canal quantique)**.** *Pour un canal quantique donné, on appelle* capacité *de ce canal (et on note Q) le rendement maximal d'un code quantique qui permet de corriger les erreurs du canal donné, autrement dit pour tout $R < Q$, pout tout $\varepsilon > 0$, il existe un code de rendement au moins R tel que la probabilité d'erreur après un décodage optimal soit inférieure à ε.*

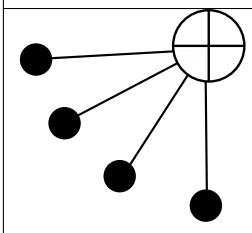
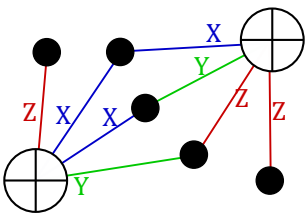
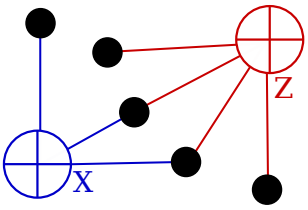
Classique	Quantique	Quantique CSS
Code linéaire de longueur n, de dimension k Espace : $\{0,1\}^n$	Code stabilisateur de longueur n, de dimension k Espace : $\mathcal{E}_n$	Code CSS de longueur n, de dimension k Espace : $(\{0,1\}^n)^2$
Matrice de parité $\mathbf{H}$ quelconque $r \times n$	Matrice des stabilisateurs $\mathbf{H}$ $r \times n$, telle que $\forall i,j,\ \mathbf{H}_i \star \mathbf{H}_j = 0$	Matrices de parité binaires $\mathbf{H}_X$ $(r_X \times n)$ et $\mathbf{H}_Z$ $(r_Z \times n)$, telles que $\mathbf{H}_X\ {}^{\mathbf{t}}(\mathbf{H}_Z) = 0$
Dimension : $k = n - \mathrm{rang}(\mathbf{H})$	$k = n - \mathrm{rang}(\mathbf{H})$	$k = n - \mathrm{rang}(\mathbf{H}_X) - \mathrm{rang}(\mathbf{H}_Z)$
Syndrome : $\mathrm{s}(w) = w \cdot {}^{\mathbf{t}}(\mathbf{H})$	$\mathrm{s}(E) = E \star {}^{\mathbf{t}}(\mathbf{H})$	$(s_X(w_1), s_Z(w_2))$, où $s_X(w_1) = w_1 \cdot {}^{\mathbf{t}}(\mathbf{H}_X)$, $s_Z(w_2) = w_2 \cdot {}^{\mathbf{t}}(\mathbf{H}_Z)$
Mot de code non nul : $w \in \{0,1\}^n$, $\mathrm{s}(w) = 0^r$ et $w \neq 0^n$	Erreur sérieuse : $E \in \mathcal{E}_n$, $\mathrm{s}(E) = 0^r$ et $E \notin \langle \mathbf{H}_1, \ldots, \mathbf{H}_r \rangle$	Erreurs sérieuses : $w_1, w_2 \in \{0,1\}^n$, $s_X(w_1) = 0, w_1 \notin \mathrm{VectLi}(\mathbf{H}_Z)$, ou $s_Z(w_2) = 0, w_2 \notin \mathrm{VectLi}(\mathbf{H}_X)$.
Distance minimale : $d = \min\{\mathrm{w}(w),$ $\mathrm{s}(w) = 0, w \neq 0^n\}$	$d = \min\{\mathrm{w}(E),$ $\mathrm{s}(E) = 0, E \notin \langle \mathbf{H}_i \rangle)\}$	$d = \min\{d_X, d_Z\}$ où $d_X = \min\{\mathrm{w}(w),$ $\mathrm{s}_X(w) = 0, w \notin \mathrm{VectLi}(\mathbf{H}_Z)\}$ $d_Z = \min\{\mathrm{w}(w),$ $\mathrm{s}_Z(w) = 0, w \notin \mathrm{VectLi}(\mathbf{H}_X)\}$
		

Tableau 3.1 – Récapitulatif des propriétés des codes classiques, quantiques stabilisateurs et quantiques CSS. La dernière ligne représente un exemple de graphe de Tanner, qui sera vu dans le chapitre suivant, section 4.1.1.

Cette définition est l'équivalent de la capacité des canaux classiques (via le théorème de Shannon, voir la section 1.1.3). Si, à cause de la dégénérescence (un syndrome peut correspondre à plusieurs erreurs), on ne sait pas borner supérieurement la capacité du canal dépolarisant, on peut tout de même la borner inférieurement avec le théorème suivant :

Propriété 20 (Théorème de la borne de hachage [Llo97]). *La capacité $Q(p)$ du canal dépolarisant de paramètre p vérifie :*

$$Q(p) \geqslant 1 - h(1-p, p/3, p/3, p/3) \tag{3.29}$$

On rappelle (voir section 1.1.3), que pour un vecteur de probabilités $(p_1, \dots p_n)$,

$$h(p_1, \dots p_n) = -\sum_{i=1}^{n} p_i \log p_i$$

et que sauf indication contraire, les log qui apparaissent dans les formules sont en base 2.

Pour un rendement donné, inversement, on appelle la probabilité correspondante la *borne de hachage* :

Définition 36 (Borne de hachage). *On appelle* borne de hachage *du canal dépolarisant la probabilité p_h qui, pour un rendement R donné, vérifie :*

$$R = 1 - h(1-p_h, p_h/3, p_h/3, p_h/3) \tag{3.30}$$

Par exemple, pour $R = 1/4$, la borne de hachage vaut $p_h \simeq 0,1269$.

On peut donc voir la propriété 20 de la façon suivante : pour un R donné, il existe des codes stabilisateurs de rendement R qui corrigent les erreurs sur le canal dépolarisant de paramètre au moins p_h.

Cependant la propriété ne fournit pas de code dépassant cette borne qui soit rapide à décoder. D'autre part, si on ne tient pas compte de la dégénérescence, alors cette borne est une borne supérieure. En effet, on peut alors considérer le canal comme un canal classique, sur un alphabet à 4 symboles (I, X, Y, Z), avec des règles de transition correspondant à celles du canal dépolarisant :

$$\begin{aligned}
T &\to \quad T \quad && \text{avec probabilité } 1-p \\
T &\to T + X && \text{avec probabilité } p/3 \\
T &\to T + Y && \text{avec probabilité } p/3 \\
T &\to T + Z && \text{avec probabilité } p/3
\end{aligned}$$

avec les relations correspondant au produit des matrices de Pauli : $X + Y = Z$, $X + Z = Y$, $Y + Z = X$.

La capacité du canal (classique) est alors :

$$C(p) \leqslant 1 - h(1-p, p/3, p/3, p/3)$$

Ce qui signifie que, tant qu'on ne tient pas compte de la dégénérescence, on ne peut pas aller au delà de la borne de hachage.

Dans le cas des codes CSS, on peut littéralement transformer le code quantique sur le canal dépolarisant en deux codes classiques (voir section 3.4) sur un canal binaire symétrique. Si on n'utilise pas la dégénérescence, on obtient là encore une borne supérieure.

Définition 37 (Borne de hachage CSS)**.** *On appelle* borne de hachage CSS *du canal dépolarisant la probabilité p_c qui vérifie, pour un rendement R donné :*

$$\frac{R+1}{2} = 1 - h(1 - 2p_c/3, 2p_c/3) \tag{3.31}$$

Cette borne donne la probabilité maximale qu'on peut corriger avec un code CSS de rendement R, si on décode les deux parties en X et en Z de façon indépendante. En effet, si on considère uniquement la partie en X, et qu'on appelle F le mot reçu et E le mot envoyé :

$$\begin{aligned} \mathbb{P}[F(v) \star X = 1] &= \mathbb{P}[F(v) = Y] + \mathbb{P}[F(v) = Z] \\ \mathbb{P}[F(v) \star X = 0] &= \mathbb{P}[F(v) = I] + \mathbb{P}[F(v) = X] \end{aligned}$$

De là, on peut en déduire que :

$$\begin{aligned} \mathbb{P}[F(v) \star X = 1 \mid E(v) \star X = 0] &= \mathbb{P}[F(v) = Y \text{ ou } Z \mid E(v) = X \text{ ou } I] \\ &= \mathbb{P}[\textit{ l'erreur reçue en } v \text{ soit } Y \textit{ ou } Z] \\ &= 2p/3 \end{aligned}$$

Et de même $\mathbb{P}[F(v) \star X = 0 \mid E(v) \star X = 1] = 2p/3$. Par le même raisonnement, on obtient également $\mathbb{P}[F(v) \star X = 0 \mid E(v) \star X = 0] = \mathbb{P}[F(v) \star X = 1 \mid E(v) \star X = 1] = 1 - 2p/3$. Ainsi, on se ramène à un canal binaire symétrique (classique) de probabilité $2p/3$.

On peut alors utiliser la capacité (classique) de ce nouveau canal : $C = 1 - h(1 - 2p/3, 2p/3)$. On a donc $R_X = 1 - h(1 - 2p/3, 2p/3)$ au maximum (R_X étant le rendement de la partie en X). Comme R_X et R_Z vérifient $R_X + R_Z - 1 = R$, où R est le rendement du code quantique et que $R_X = R_Z$, cela donne $R = 2(1 - h(1 - 2p/3, 2p/3)) - 1$, d'où l'équation (3.31).

Par exemple, pour $R = 1/4$, on a $p_c \simeq 0,1087$, ce qui est un peu inférieur à la borne de hachage générale. Cependant, cela ne signifie pas que les codes CSS sont limités à cette probabilité de décodage : si on décide de décoder un tel code sans « séparer » X et Z, alors on peut toujours en théorie atteindre la borne de hachage.

3.7 Résultats pratiques

Dans le monde des codes correcteurs classiques, non seulement on sait estimer la capacité des canaux, mais on sait également construire des codes efficaces qui approchent le rendement maximal correspondant à la capacité. Dans le monde (plus récent) des codes quantiques, on n'a pas encore toutes ces réponses, même si beaucoup de progrès ont été faits.

Tout d'abord, hors certaines constructions avec un décodage spécifique (notamment le code torique, voir section 4.1.3), le décodage ne tient pas compte de la dégénérescence : ainsi, impossible d'espérer dépasser la borne de hachage (ou la borne de hachage CSS si on décode séparément les deux parties d'un code CSS). Mais même cette borne est difficile à atteindre. On peut voir sur la figure 3.4 un certain nombre de courbes de décodages de codes, ainsi que les deux bornes de hachage.

On distingue principalement trois classes de codes quantiques. La plus fournie est celle des *codes LDPC quantiques*, dont il est principalement question ici (voir chapitre 4 pour plus de détails). On trouve également les *turbo-codes quantiques* (voir [Abb13, PTO09, WH11]) et, apparus plus récemment, les *codes polaires* ([RDR12, WG13]) qui atteignent en théorie la borne de hachage.

Les différentes courbes de décodage présentées sur le graphique de la figure 3.4 sont issues de [GFL08] (Garcia-Liu), [PTO09] (turbo-codes de différentes longueur), [MMM04] (MacKay), [LGF06] (Lou-Garcia), [COT07] (Camara-Ollivier-Tillich).

Parmi les différentes constructions présentées dans ce document, certaines ne sont pas meilleures que celles de la littérature (voir chapitre 5), en revanche, d'autres sont très encourageantes, notamment celle présentée dans le chapitre 7, qui avec un décodage adapté dépasse la borne CSS et s'approche de la borne de hachage.

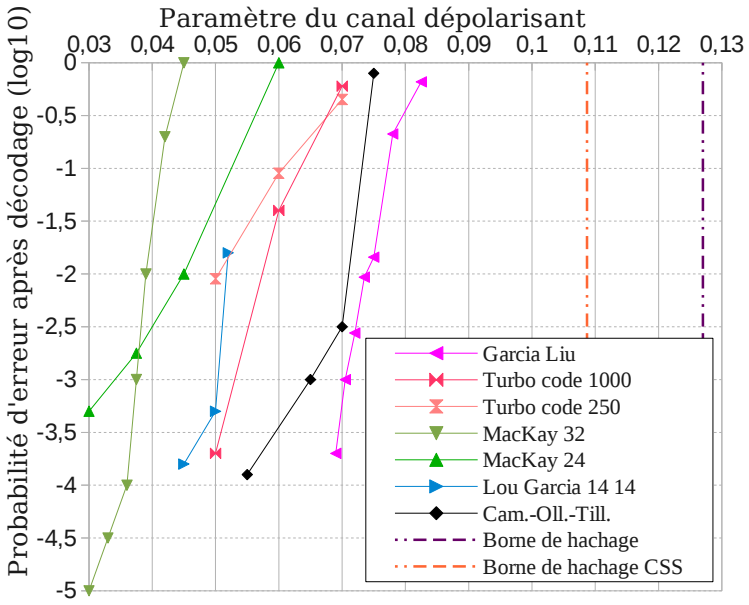


FIGURE 3.4 – Courbes de décodages de certains codes correcteurs d'erreur quantiques. En abscisse, on a la probabilité d'erreur sur le canal dépolarisant, et en ordonnée le logarithme en base 10 de la probabilité d'erreur résultante ; autrement dit, plus la courbe est en bas à droite, mieux le décodage se passe. Les lignes verticales représentent respectivement la borne de hachage et la borne de hachage CSS.

Chapitre 4

Codes LDPC quantiques et leur décodage

Ce chapitre présente les codes LDPC quantiques, dont le fonctionnement s'inspire des codes LDPC classiques (voir 1.3). On y introduit notamment le graphe de Tanner d'un code stabilisateur (voir 1.3.1 pour sa version classique), puis d'un code CSS, ce qui qui permet d'introduire l'algorithme de propagations de croyances quantique, dans la section 4.2. Enfin on étudie différentes variantes de cet algorithme.

Les codes LDPC classiques (voir section 1.3) sont des codes extrêmement pratiques : ils sont simples à générer pour n'importe quel rendement, et surtout on dispose pour les décoder d'un bon algorithme, le BP (algorithme de propagation de croyances) : il est non seulement peu coûteux en temps, mais il est aussi très efficace en termes de capacité de correction. Or, en quantique, on peut rappeler que ces deux aspects sont cruciaux : pour espérer faire fonctionner une machine quantique, il faut pouvoir corriger les erreurs *efficacement* et *rapidement*, c'est-à-dire plus vite qu'elles n'apparaissent.

On compte plusieurs grandes catégories de codes LDPC quantiques existantes, on peut retrouver certaines des principales constructions dans le tableau ci-dessous.

Rendement tend vers 0	Rendement constant	
Distance minimale (d) qui tend vers $+\infty$		d constante
• Codes de surface 2D : [Kit03] (le code torique, voir section 4.1.3), [BK98, Bac06, BMD07, BMD06, Bom10, DZ10]. • Codes de surface 3D : [BLT11] (non CSS), [Haa11] • Homologies : [Aud13] • Graphes de Cayley : [CDZ11]	• Codes produit (distance en n^α) : [TZ09] (voir chapitre 5), [BH13] (généralisation du précédent, pas forcément LDPC au sens habituel) • Codes sur des surfaces de grand genre : [FML02, Kim07, Zé09] (distance en $\log n$), [GL13] (distance en n^α)	• Codes quasi-cycliques : [MMM04, HI07, Aly07, HBD08], [TL10] (non CSS) • Matrice génératrice creuse : [GFL08] • Groupes finis : [COT07]

4.1 Définition des codes LDPC quantiques

Comme dans le cas des codes classiques, un code LDPC est un code dont la matrice de parité (ou plutôt son équivalent, qui est la matrice des stabilisateurs) est creuse, c'est-à-dire qu'elle contient un nombre constant d'entrées non I sur les lignes et les colonnes :

Définition 38 (Code LDPC quantique)**.** *Une famille de codes $(\mathscr{C}_n)_n$ de longueur n stabilisée par $(\mathcal{S}^n)_n$ est dite LDPC s'il existe pour tout n une représentation matricielle $\mathbf{H}^n$ de $\mathcal{S}^n$ creuse ; autrement dit s'il existe A et B tels que :*

$$\forall n, \forall i, \quad \#\{j, \mathbf{H}^n_{i,j} \neq I\} \leqslant A$$

$$\forall n, \forall j, \quad \#\{i, \mathbf{H}^n_{i,j} \neq I\} \leqslant B$$

Comme dans le cas classique, on définit le caractère LDPC pour une famille de codes pour que la définition ait du sens, mais on parlera plus généralement d'*un* code LDPC, lorsque le poids des lignes et des colonnes est faible en comparaison de la taille de la matrice.

On peut alors constater une première mauvaise nouvelle : la distance minimale non dégénérée (voir définition 28) est constante. En effet, chaque ligne de $\mathbf{H}$ est un élément du stabilisateur, et est à ce titre une erreur non détectée. Comme ces lignes sont de poids borné, cela fournit naturellement des erreurs indétectées de poids constant.

Pourtant ces erreurs appartiennent au stabilisateur, donc ce ne sont pas des erreurs sérieuses, et elles n'ont en réalité aucun impact sur le système quantique, mais un décodeur éventuel qui s'inspirerait du décodage classique ne saurait pas en tenir compte (voir la section 4.2 pour le cas du décodage itératif).

4.1.1 Graphe de Tanner

Là encore, comme dans le cas des codes classiques, on va représenter un code stabilisateur sous forme de graphe. L'algorithme BP quantique se décrit plus facilement sur ce modèle, et même sans tenir compte du décodage, un certain nombre de propriétés du code sont liées au propriétés de son graphe de Tanner : les outils de la théorie des graphes peuvent alors être utiles.

Dans le schéma binaire (voir la section 1.3.1), on met une arête entre deux nœuds lorsqu'il y a un 1 dans la matrice de parité, et une absence d'arête lorsqu'il y a un 0. Ici, on a quatre possibilités : I, X, Y, Z. On utilisera une absence d'arête pour représenter I, et pour distinguer s'il s'agit de X, Y ou Z, on parlera d'*arête en X, Y ou Z*, et on les représentera graphiquement par des arêtes de différentes couleurs.

Définition 39 (Graphe de Tanner d'un code stabilisateur)**.** *Soit $\mathscr{C}$ un code stabilisateur de longueur n stabilisé par $\mathcal{S} = \langle S_1, \dots S_r \rangle$. On définit son graphe de Tanner de la façon suivante :*

$$\mathscr{G}(\mathscr{C}) = (V, C, E_X, E_Y, E_Z)$$

où :

- $V = \{1, \dots, n\}$ *est l'ensemble des* nœuds variable,
- $C = \{1, \dots, r\}$ *est l'ensemble des* nœuds de parité,
- $E_X = \{v \bullet\!\!-\!\!\circ c, v \in V, c \in C, S_c(v) = X\}$ *est l'ensemble des* arêtes en X,
- $E_Y = \{v \bullet\!\!-\!\!\circ c, v \in V, c \in C, S_c(v) = Y\}$ *est l'ensemble des* arêtes en Y,
- $E_Z = \{v \bullet\!\!-\!\!\circ c, v \in V, c \in C, S_c(v) = Z\}$ *est l'ensemble des* arêtes en Z.

La définition ci-dessus implique que, pour un graphe donné, il n'y a pas d'arête donnée deux fois : si $v \bullet\!\!-\!\!\circ c \in E_X$, alors $v \bullet\!\!-\!\!\circ c \notin E_Y$ et $v \bullet\!\!-\!\!\circ c \notin E_Z$, et de même pour E_Y et E_Z.

De façon générale, on n'utilisera pas toujours des entiers de 1 à n pour indexer les nœuds variable (ou de 1 à r) pour les nœuds de parité : n'importe quel ensemble fini du bon cardinal fait l'affaire. Pour les petits exemples notamment, on choisira parfois des lettres pour les nœuds de parité (afin de bien les distinguer des nœuds variable).

On peut reprendre l'exemple de la section 3.3.2.4, c'est à dire le code ayant pour matrice de stabilisateurs :

$$\mathbf{H} = \begin{pmatrix} X & Y & Z & I \\ Y & Z & I & X \\ Z & I & X & Y \end{pmatrix} \tag{4.1}$$

Son graphe de Tanner est donné par celui de la figure 4.1.

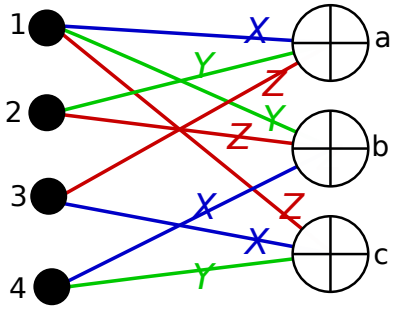


FIGURE 4.1 – Le graphe de Tanner du code dont la matrice de stabilisateurs est définie dans (4.1). De façon générale, on prendra la convention : bleu pour X, vert pour Y, rouge pour Z.

Si on manipule le graphe de Tanner, on peut avoir besoin de regarder (à l'inverse) le code quantique défini par un graphe de Tanner, aussi on définit :

Définition 40 (Stabilisateur associé à un nœud de parité)**.** *Soit $\mathscr{G}(\mathscr{C})$ le graphe de Tanner d'un code stabilisateur, pour $c \in C$ on note $\mathcal{S}(c)$ le stabilisateur associé à c, autrement dit :*

$$\mathcal{S}(c)(i) = \begin{cases} X & si\ i \multimap c \in E_X \\ Y & si\ i \multimap c \in E_Y \\ Z & si\ i \multimap c \in E_Z \\ I & sinon \end{cases}$$

Ainsi, vérifier si deux stabilisateurs commutent, au niveau du graphe, revient à regarder deux nœuds de parité c_1 et c_2 et d'observer à quels nœuds variable ils sont reliés : chaque fois qu'ils sont reliés à un même nœud variable avec des arêtes de couleur différente, ils anti-commutent en ce nœud. Par exemple, sur la figure 4.1, les nœuds de parité a et b anti-commutent sur les nœuds 1 (X pour a, Y pour b) et 2 (Y pour a, Z pour b) : $\mathcal{S}(a)$ et $\mathcal{S}(b)$ commutent donc.

4.1.2 Graphe de Tanner d'un code CSS

Dans le cas particulier des codes CSS (voir section 3.4), on peut avoir une représentation plus simple. E_Y n'existe plus, et tous les nœuds de parité sont reliés

à un même type d'arêtes : il y a ceux reliés aux arêtes en X et ceux reliés aux arêtes en Z. On choisit donc de les représenter d'une façon un peu plus adaptée :

Définition 41 (Graphe de Tanner d'un code CSS)**.** *Soit $\mathscr{C}$ un code CSS déterminé par $\mathbf{H}_X$ et $\mathbf{H}_Z$. On définit son graphe de Tanner de la façon suivante :*

$$\mathscr{G}(\mathscr{C}) = (V, C_X, C_Z, E_X, E_Z)$$

où :

- *V est l'ensemble des* nœuds variable*, un ensemble de cardinal n,*
- *C_X est l'ensemble des* nœuds de parité en X*, de cardinal r_X,*
- *C_Z est l'ensemble des* nœuds de parité en Z*, de cardinal r_Z,*
- *$E_X = \{v \rightarrowtail c, v \in V, c \in C_X, (H_X)_{c,v} = 1\}$ est l'ensemble des* arêtes en X*,*
- *$E_Z = \{v \rightarrowtail c, v \in V, c \in C_Z, (H_Z)_{c,v} = 1\}$ est l'ensemble des* arêtes en Z*.*

On peut alors reprendre l'exemple de la section 3.4.3, et tracer le graphe de Tanner équivalent, sur la figure 4.2 :

$$H_X = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \qquad H_Z = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{4.2}$$

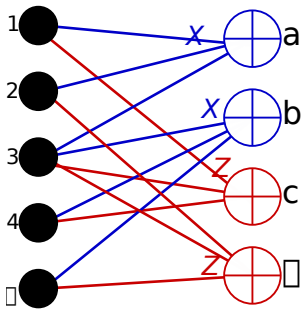


FIGURE 4.2 – Le graphe de Tanner du code CSS dont les matrices de parité sont définies par (4.2).

4.1.3 Un exemple : le code torique de Kitaev

Le code LDPC quantique le plus connu est très certainement le code torique, appelé aussi code de Kitaev [Kit03]. Il s'agit d'un code CSS, avec pour poids des lignes 4 et poids des colonnes 2.

Définition 42 (Code torique de Kitaev)**.** *Le* code torique de Kitaev *de paramètre n est le code CSS défini par le graphe de Tanner $\mathscr{G} = (V, C_X, C_Z, E_X, E_Z)$ tel que :*

- $V = \{(2i, 2j), (2i+1, 2j+1), 0 \leqslant i, j < n\}$
- $C_X = \{(2i+1, 2j), 0 \leqslant i, j < n\}$
- $C_Z = \{(2i, 2j+1), 0 \leqslant i, j < n\}$
- $E_X = \{(2i, 2j) \rightarrowtail (2i+1, 2j),\ (2i, 2j) \rightarrowtail (2i-1, 2j),$
  $(2i+1, 2j+1) \rightarrowtail (2i+1, 2j),\ (2i+1, 2j+1) \rightarrowtail (2i+1, 2j+2), 0 \leqslant i, j < n\}$
- $E_Z = \{(2i, 2j) \rightarrowtail (2i, 2j+1),\ (2i, 2j) \rightarrowtail (2i, 2j-1),$
  $(2i+1, 2j+1) \rightarrowtail (2i, 2j+1),\ (2i+1, 2j+1) \rightarrowtail (2i+2, 2j+1), 0 \leqslant i, j < n\}$

avec toutes les coordonnées prises modulo $2n$.

Le choix des coordonnées comme indice des nœuds (au lieu d'entiers) permet de représenter le code facilement en associant les nœuds à des coordonnées cartésiennes (voir figure 4.3). De plus, le fait d'avoir des coordonnées paires et impaires permet de repérer rapidement quel nœud représente un nœud variable, un nœud de parité en X ou un nœud de parité en Z.

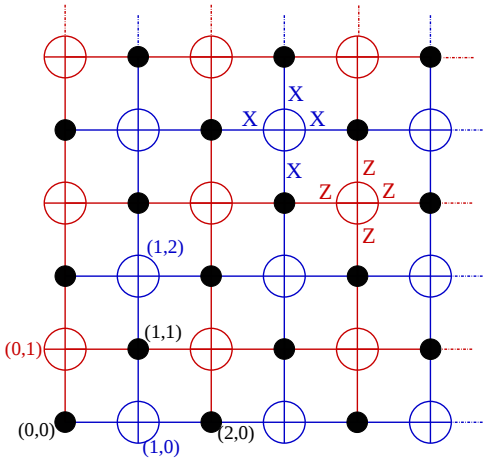


FIGURE 4.3 – Le code torique de taille 2×3^2. Les extrémités gauche et droite sont reliées, ainsi que les extrémités haute et basse, d'où le nom de « code torique ».

Propriété 21 (Propriétés du code torique)**.** *Le code torique de paramètre* n *a pour longueur* $2n^2$*, dimension* 2 *et distance minimale* n*.*

Démonstration. Toutes ces propriétés sont prouvées dans [Kit03], mais on verra ici d'autres façons de retrouver les résultats.

Pour la dimension, la preuve formelle est faite dans la section 5.2.2. Cependant, on peut constater que s'il y a autant de nœuds variable que de nœuds de parité (donc la dimension devrait être nulle), la somme de tous les nœuds de parité en

X est triviale : il y a donc au moins un nœud redondant, et de même en Z (ce qui donne une dimension de 2).

Pour la distance minimale, il y a plusieurs preuves dans ce document (voir les sections 5.2.2 ou 6.2.3), mais on peut voir l'idée générale : si on regarde le graphe en X (pour les erreurs composées de I et de Z uniquement), alors chaque nœud variable est relié à exactement deux nœuds de parité. Les mots de code sont donc des cycles dans le graphe de Tanner. Or tous les petits cycles (voir figure 4.4) sont en réalité des stabilisateurs, et seuls les cycles faisant le « tour » du tore ne sont pas des combinaisons de petits cycles, et le tour du tore a pour taille n.

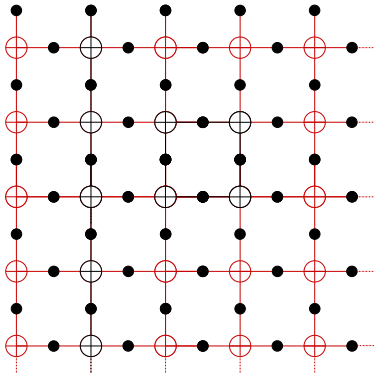


FIGURE 4.4 – Le code torique de paramètre 3 (c'est-à-dire de taille 2×3^2), avec deux erreurs sérieuses composées de Z. La première (le trait vertical) fait le tour du tore : c'est une erreur sérieuse. La seconde (le « carré ») est une erreur du stabilisateur.

■

On peut noter que ce code possède un algorithme spécifique qui permet de le décoder de façon très efficace, en tenant compte de la dégénérescence. Malheureusement, cet algorithme ne s'adapte pas aux autres codes LDPC.

4.2 Décodage

Le principal intérêt des codes LDPC, dans le monde classique, est leur décodage. L'algorithme de propagation de croyances (BP), présenté en partie 1.3.2, est non seulement rapide dans son exécution (linéaire en le nombre d'arêtes du graphe), mais également très efficace pour décoder. Il en existe diverses versions pour les codes stabilisateurs quantiques.

4.2.1 Généralités : simulation de décodage

Pour bien comprendre l'adaptation de l'algorithme BP, il faut se souvenir de la façon dont on décode, en classique et en quantique. Dans le premier cas (voir la section 1.3.2), on peut « voir » les bits reçus, et on cherche à savoir quelle est la probabilité, pour chaque bit, qu'il soit 0 ou 1. Dans le cas quantique, en revanche, on ne peut pas « voir » le qubit reçu, la seule information disponible est le syndrome, autrement dit avec quels stabilisateurs (ou nœuds de parité) l'erreur commute ou non. À partir de cette donnée, on cherche alors à savoir quelle erreur de Pauli est arrivée au système.

Ainsi, si dans le cas classique, on part avec (grossièrement) les bits reçus sur les nœuds de variable, et on cherche à satisfaire tous les nœuds de parité ; pour les codes quantiques on part avec I (l'erreur supposée) sur les nœuds de variable, et on cherche à satisfaire ou non-satisfaire les nœuds de parité selon le syndrome sur le nœud en question.

Dans la pratique, on peut se placer dans un modèle simplifié (un peu comme en classique on ne teste que le mot de code 0^n), et l'algorithme ressemble beaucoup plus au cas classique (voir partie 4.2.3), à savoir qu'on « voit » une certaine erreur, et qu'on essaie de faire converger l'algorithme vers une erreur indétectée (donc satisfaire les nœuds de parité) proche.

De façon générale, ces variantes ne tiennent pas compte de la dégénérescence, ce qui signifie que l'algorithme cherche l'erreur *exacte*, et non pas au stabilisateur près. La première conséquence est que le décodage est toujours limité par la borne de hachage (voir section 3.6). Comme de toutes façons la plupart des constructions actuelles sont encore loin en deçà de cette borne, elle n'est pas très limitante.

La seconde conséquence, propre aux codes LDPC, vient de la présence de stabilisateurs de poids faible (par définition des codes LDPC), autrement dit il existe des erreurs indétectées de poids faible, que l'algorithme ne sait pas distinguer des erreurs sérieuses. On pourrait croire que ce n'est pas grave, car dans la réalité quantique ces erreurs n'ont aucun impact sur le système, et de plus lors de simulations, il est possible de déterminer après le décodage si l'erreur trouvée est dans le stabilisateur ou non (et attester de la réussite ou non du décodage). Mais cela n'est pas suffisant.

Par exemple, comme on le voit sur la figure 4.2.1, on peut avoir un mot de code de poids 4 (un stabilisateur) et des erreurs sur la moitié des nœuds concernés. L'algorithme peut alors avoir différents comportements. Soit il réussit à converger vers le mot de code de poids 4, ou aucune erreur, ce qui est très bien dans les deux cas, soit il « alterne » en permanence entre les deux erreurs de départ et son complémentaire, et ne parvient pas à converger. Ce cas-là est plus problématique. De façon générale, l'algorithme ne parvient pas toujours à converger lorsqu'il est

confronté à des erreurs de poids 1/2 de la distance minimale (non dégénérée).

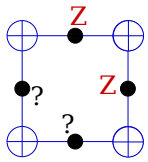


FIGURE 4.5 – Exemple d'erreur de poids 2 (en Z) dans un cas où il existe une erreur indétectée de poids 4 (a priori un stabilisateur).

Il est plus difficile de régler ce problème, on peut cependant essayer d'avoir des stabilisateurs de poids moins faible pour que ces cas surviennent moins souvent.

4.2.2 BP quantique général

Si on exclut le fait qu'on part du syndrome et non du mot reçu (qu'on verra plus loin), le décodage est assez similaire au cas classique, à part qu'il y a 4 possibilités (I, X, Y ou Z) au lieu de 2 (0 ou 1). Par exemple sur la figure 4.6, on suppose qu'on a 3 des 4 nœuds de variable connus et que le syndrome sur le nœud de parité est de 1. On peut alors en déduire que l'erreur sur le quatrième nœud est Y ou Z (car pour avoir un syndrome 1, on doit avoir $E(4) \star X = 1$). Il faut une information supplémentaire (par exemple, est-ce que $E(4)$ commute ou non avec Y) pour déduire complètement l'erreur en 4.

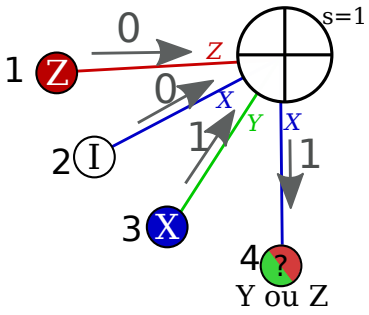


FIGURE 4.6 – Exemple de propagation de messages sur un morceau de graphe de Tanner quantique : on peut en déduire que $E(v) \star X = 1$.

De façon générale, les informations pertinentes qui peuvent circuler sur les arêtes ne sont pas du type, comme en classique « est-ce que le nœud adjacent est 0 ou 1 », mais plutôt « est-ce que l'erreur du nœud adjacent commute ou non avec celle de l'arête » : en effet, c'est la seule information que le nœud de parité peut utiliser pour propager. Ensuite, en recevant les informations combinées de plusieurs arêtes, on peut déduire l'erreur sur le nœud.

Comme dans le cas classique, on suppose que le graphe est un arbre (contrairement au cas classique, on verra dans la pratique que c'est loin d'être exact et qu'il y a beaucoup de cycles de taille 4), et des messages vont circuler le long des arêtes comme sur la figure 4.7.

4.2.2.1 Définition des messages

On se place dans un code $\mathscr{C}$ de graphe de Tanner $\mathscr{G}[\mathscr{C}] = (V, C, E_X, E_Y, E_Z)$, où on a le syndrome s et on cherche l'erreur E qui est arrivée : on cherche donc à évaluer pour tout v, $\mathbb{P}[E(v) = I]$, $\mathbb{P}[E(v) = X]$, $\mathbb{P}[E(v) = Y]$, et $\mathbb{P}[E(v) = Z]$.

On note $\mathcal{E}_1 = \{I, X, Y, Z\}$ et $\mathcal{E}' = \{X, Y, Z\}$. On note également, pour $v \in V$, le voisinage dans E un ensemble d'arêtes du nœud v : $\mathcal{V}_E(v) = \{v \bullet\!\!-\!\!-\!\!\bullet c \in E\}$, et de la même façon le voisinage dans E d'un nœud de parité c : $\mathcal{V}_E(c) = \{v \bullet\!\!-\!\!-\!\!\bullet c \in E\}$. On note également $E_{\mathcal{E}'} = E_X \cup E_Y \cup E_Z$.

On définit alors :

- $\forall v \in V$, $\forall T \in \mathcal{E}_1$
$$p_v^i(T) = \mathbb{P}\left[E(v) = T\right] \tag{4.3}$$
Il s'agit de la *probabilité initiale*, ou *probabilité a priori* sur un nœud : c'est une grandeur qui ne change pas, elle est donnée par le canal. Dans le cas général, ce sera toujours la même (on ne connaît pas encore de moyen de différencier les nœuds car au début du décodage on n'a accès qu'au syndrome, mais rien sur les nœuds).
- $\forall U \in \mathcal{E}'$, $e = v \bullet\!\!-\!\!-\!\!\bullet c \in E_U$, $b \in \{0, 1\}$,
$$q_e(b) = \mathbb{P}[E(v) \star U = b \mid r_{e'}, e' \in \mathcal{V}_{E_{\mathcal{E}'}}, e' \neq e] \tag{4.4}$$
$q_e(0)$ est la probabilité que $E(v)$ commute (et $q_e(1)$ que $E(v)$ ne commute pas) avec l'erreur (U) associée à l'arête e, et ce, en tenant compte des informations en provenance des arêtes adjacentes à v (sauf e). Par exemple, si l'arête e est une arête en X, $q_e(0)$ est la probabilité que $E(v) \star X = 0$, sachant les informations venant des arêtes adjacentes à v, sauf de e elle-même.
- $\forall U \in \mathcal{E}'$, $e = v \bullet\!\!-\!\!-\!\!\bullet c \in E_U$, $b \in \{0, 1\}$,
$$r_e(b) = \mathbb{P}[E(v) \star U = b \mid q_{e'}, e' \in \mathcal{V}_{E_{\mathcal{E}'}}(c), e' \neq e, s(c)] \tag{4.5}$$
$r_e(0)$ est la probabilité que $E(v)$ commute (et $r_e(1)$ celle que $E(v)$ ne commute pas) avec l'erreur associée à l'arête e, et ce, en tenant compte des informations en provenance des arêtes adjacentes à c (sauf e), et bien sûr du syndrome en ce nœud de parité c. Ainsi, si e est une arête en X et que $s(c) = 1$, $r_e(0)$ est la probabilité que $E(v) \star X = 0$, sachant les informations venant des arêtes adjacentes à c (sauf de e elle-même), et que $s(c) = 1$, autrement dit que le nœud de parité *ne doit pas* être satisfait.

– $\forall v \in V$, $\forall T \in \mathcal{E}_1$,

$$p_v^f(T) = \mathbb{P}[E(v) = T \mid r_e, e \in \mathcal{V}_{E_{\mathcal{E}'}}(v)] \tag{4.6}$$

La probabilité finale que l'erreur sur le nœud soit T. On tient compte des messages provenant des arêtes adjacentes.

4.2.2.2 Mise à jour des messages

On cherche à exprimer ces différentes grandeurs les unes en fonction des autres.

– Calcul des p_v^f : $\forall v_1 \in V$, $\forall T \in \mathcal{E}_1$,

$$p_{v_1}^f(T) \propto p_{v_1}^i(T) \prod_{U \in \mathcal{E}'} \prod_{e \in \mathcal{V}_{E_U}(v_1)} r_e(T \star U) \tag{4.7}$$

En effet, l'erreur en v est T si toutes les arêtes adjacentes acceptent T comme erreur. Pour une arête adjacente e de type U où U commute avec T, il faut donc prendre $r_e(0)$, et pour une erreur qui ne commute pas avec T, il faut prendre $r_e(1)$, d'où le $r_e(T \star U)$. Par exemple, si on cherche $p_v^f(X)$, comme sur la figure 4.7 :

$$p_{v_1}^f(X) \propto p_v^i(X) \prod_{e \in \mathcal{V}_{E_X}(v_1)} r_e(0) \prod_{e \in \mathcal{V}_{E_Y}(v_1)} r_e(1) \prod_{e \in \mathcal{V}_{E_Z}(c)} r_e(1)$$

Après avoir calculé les quatre grandeurs, il faut les renormaliser de façon à avoir $p_v^f(I) + p_v^f(X) + p_v^f(Y) + p_v^f(Z) = 1$.

– Calcul des r_e : $\forall U \in \mathcal{E}'$, $e = v_1 \multimap c_1 \in E_U$, $b \in \{0, 1\}$,

$$r_e(b) = \sum_{\substack{b_1 + \ldots + b_d \\ = s_{c_1} + b}} \prod_{\substack{e_i \in \mathcal{V}_{E_{\mathcal{E}'}}(c_1) \\ e_i \neq e}} q_{e_i}(b_i) \tag{4.8}$$

En effet, le nœud de parité c_1 doit être satisfait si $s_{c_1} = 0$ (S_{c_1} est le syndrome associé à ce nœud de parité) ou non-satisfait si $s_{c_1} = 1$. On doit donc faire la somme sur toutes les combinaisons de bits dont la somme donne $b + S_{c_1}$ de la probabilité que cette combinaison apparaisse. Une telle probabilité est égale au produit des probabilités q_{e_i} pour les arêtes adjacentes e_i (hors l'arête considérée $v_1 \multimap c_1$). Comme dans le cas classique (voir le calcul de (1.9)), on

peut effectuer cette somme en un temps raisonnable, en considérant :

$$\begin{aligned}\Delta r_e &= r_e(0) - r_e(1)\\ &= \sum_{b_1+\ldots+b_d=s_{c_1}} \prod_{e_i} q_{e_i}(b_i) - \sum_{b_1+\ldots+b_d=s_{c_1}+1} \prod_{e_i} q_{e_i}(b_i)\\ &= (-1)^{s_{c_1}} \sum_{b_1,\ldots,b_i} (-1)^{b_1+\ldots b_i} \prod_{e_i} q_{e_i}(b_i)\\ &= (-1)^{s_{c_1}} \prod_{e_i} (q_{e_i}(0) - q_{e_i}(1))\end{aligned}$$

Ainsi, on peut bien calculer cette grandeur en un temps $O(d)$ où $d+1$ est le degré du nœud de parité (et d est borné à cause du caractère LDPC du code).

– Calcul des q_e : $\forall U \in \mathcal{E}'$, $e = v_2 \bullet\!\!-\!\!\circ c_1 \in E_U$, $b \in \{0,1\}$,

$$\begin{aligned}q_e(b) \propto \sum_{T, T\star U=b} p^i_{v_2}(T) \prod_{\substack{e' \in \mathcal{V}_{E_X}(v_2)\\ e' \neq e}} r_{e'}(T \star X) \\ \prod_{e' \in \mathcal{V}_{E_Y}(v_2)\setminus e} r_{e'}(T\star Y) \prod_{e' \in \mathcal{V}_{E_Z}(v_2)\setminus e} r_{e'}(T \star Z)\end{aligned} \tag{4.9}$$

En effet,

$$\begin{aligned}q_e(b) &= \mathbb{P}\left[E(v_2) \star U = b \mid r'_e, e' \in \mathcal{V}_{E_{\mathcal{E}'}}(v_2), e' \neq e\right]\\ &= \sum_{T, T\star U=b} \mathbb{P}[E(v_2) = T \mid r_{e'}, e' \in \mathcal{V}_{E_{\mathcal{E}'}}(v_2), e' \neq e]\end{aligned}$$

Or $\mathbb{P}[E(v_2) = T \mid r'_e]$ peut se calculer de la même façon que (4.7), sans tenir compte de l'arête $v_2 \bullet\!\!-\!\!\circ c_1$. Si on écrit

$$t(v_2, T) = p^i_{v_2}(T) \prod_{U \in \mathcal{E}'} \prod_{e' \in \mathcal{V}_{E_U}(v_2)\setminus e} r_{e'}(T \star U)$$

alors on a :

$$\mathbb{P}[E(v_2) = T \mid r'_e] = \frac{t(v_2, T)}{\sum_{T' \in \mathcal{E}_1} t(v_2, T')}$$

et donc

$$q_{v_2 \bullet\!\!-\!\!\circ c_1}(b) = \frac{1}{\sum_{T' \in \mathcal{E}_1} t(v_2, T')} \sum_{T, T\star U=b} t(v_2, T)$$

Ainsi, le facteur de renormalisation est le même pour $b = 0$ et $b = 1$, on peut donc écrire la formule (4.9) et renormaliser ensuite.

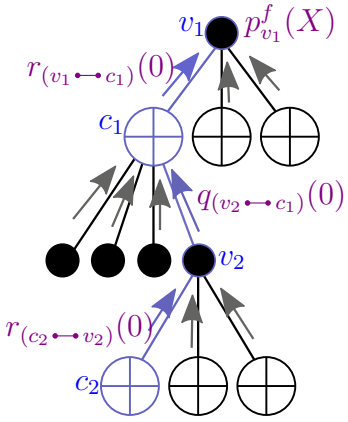


FIGURE 4.7 – Propagation des différentes grandeurs : r_e, q_e, p^f_v.

– Calcul des p^i_v : cela dépend du canal utilisé. Dans le cas du canal dépolarisant de paramètre p, on aura : $\forall v \in V$

$$\begin{aligned} p^i_v(I) &= 1-p \\ p^i_v(X) &= p/3 \\ p^i_v(Y) &= p/3 \\ p^i_v(Z) &= p/3 \end{aligned} \tag{4.10}$$

Tout comme dans le cas classique, il n'est pas nécessaire de stocker toutes les valeurs : notamment pour les q_e et r_e, on peut ne garder que $q_e(0)$ et $r_e(0)$. De même pour les $p^i_v(T)$ et $p^f_v(T)$, on peut ne stocker que 3 valeurs sur les 4.

4.2.2.3 Déroulement de l'algorithme

Algorithme Propagation de croyances quantique
{ *Initialisation* }
Initialiser les $p^i_v(T)$ $\forall v \in V,\ \forall T \in \mathcal{E}_1$ avec (4.10)
$p^f_v(T) \leftarrow p^i_v(T)$ $\forall v \in V,\ \forall T \in \mathcal{E}_1$
$q_{v \bullet\!\!-\!\!\bullet c} \leftarrow p^i_v$ $\forall v \bullet\!\!-\!\!\bullet c \in E$
{ *Déroulement* }

TantQue nombre d'itérations $\leqslant$ NB_MAX, et que E change **Faire**

Calculer les Δq_e

En déduire les Δr_e et mettre à jour les r_e avec (4.8), en utilisant les $s(c)$

Mettre à jour les q_e avec (4.9)

Si nombre d'itérations mod $\delta = 0$ **Alors**

Mettre à jour les p_v^f avec (4.7)

Mettre à jour E l'erreur : $E(v) = \underset{T}{\operatorname{argmax}}(p_v^f(T))$

FinSi

FinTantQue

Retourner E

On verra les détails des performances dans la section suivante, dans le modèle simplifié.

4.2.3 Modèle simplifié

Le modèle simplifié est en quelque sorte l'équivalent du « mot de code 0^n » dans le cas classique : l'algorithme du BP classique étant symétrique en 0 et en 1, on teste ses performances en n'utilisant que le mot de code 0^n comme mot de code transmis.

Ici, l'algorithme est invariant par l'opération suivante sur un nœud variable, qu'on appellera « inverser I et X » :

$$\begin{aligned} I &\to X \\ X &\to I \\ Y &\to Z \\ Z &\to Y \end{aligned}$$

à condition qu'on adapte le syndrome en conséquence, et qu'on inverse les probabilités initiales en conséquence ($p_v^i(X)$ devient $p_v^i(I)$, etc).

De la même façon, les opérations « inverser I et Z » et « inverser I et Y » sur un nœud variable laissent l'algorithme invariant.

Soit un syndrome s causé par une erreur E. Pour chaque v, on effectue l'opération adéquate (c'est-à-dire qu'on inverse I et X si $E(v) = X$, etc) sur l'algorithme : on obtient alors un départ d'algorithme qui ressemble beaucoup au classique : le syndrome est nul, autrement dit on cherche à satisfaire tous les nœuds de parité, et on a des probabilités initiales selon l'erreur E. Pour effectuer une simulation de décodage, il faut générer l'erreur E selon le canal, puis en déduire s : dans le modèle simplifié, on met directement l'erreur sur les nœuds.

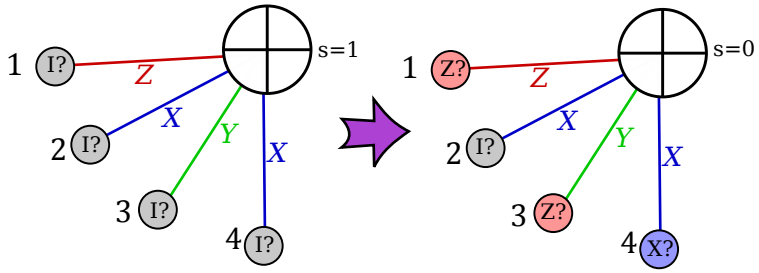


FIGURE 4.8 – Décodage BP quantique dans le modèle simplifié. À gauche, l'algorithme tel qu'il doit être implémenté en pratique : on a le syndrome, et on recherche l'erreur qui l'a causé, en supposant a priori que l'erreur est I. À droite, le modèle simplifié : on commence avec les probabilités a priori des erreurs reçues (et un syndrome nul), et on cherche à savoir si l'algorithme revient bien à I partout.

Par exemple, supposons qu'on ait affaire au graphe de Tanner de la figure 4.8. On génère une erreur selon un canal dépolarisant de probabilité p, et on aboutit à l'erreur $E = [ZIZX]$. On en déduit alors le syndrome sur le seul nœud de parité : 1.

Selon le modèle d'origine, il faudrait initialiser le syndrome à 1, et toutes les probabilités p^i à :

$$\begin{aligned} p_v^i(I) &= 1-p & p_v^i(X) &= p/3 \\ p_v^i(Y) &= p/3 & p_v^i(Z) &= p/3 \end{aligned}$$

Or l'algorithme s'exécute de façon équivalente si on initalise le syndrome à 0 et les probabilités p^i à :

$$\begin{aligned} p_1^i(I) &= p/3 & p_1^i(X) &= 1-p \\ p_1^i(Y) &= p/3 & p_1^i(Z) &= p/3 \end{aligned}$$

$$\begin{aligned} p_2^i(I) &= 1-p & p_2^i(X) &= p/3 \\ p_2^i(Y) &= p/3 & p_2^i(Z) &= p/3 \end{aligned}$$

$$\begin{aligned} p_3^i(I) &= p/3 & p_3^i(X) &= p/3 \\ p_3^i(Y) &= p/3 & p_3^i(Z) &= 1-p \end{aligned}$$

$$p_4^i(I) = p/3 \qquad p_4^i(X) = 1-p$$
$$p_4^i(Y) = p/3 \qquad p_4^i(Z) = p/3$$

Si l'algorithme est capable de retomber sur l'erreur I après ce décodage, c'est qu'il est capable avec exactement les mêmes étapes de retrouver l'erreur E en lui donnant le syndrome : on peut donc, dans un souci de recherche de performances, utiliser ce modèle simplifié.

Ce modèle simplifié est, de plus, très proche du modèle classique, où on génère l'erreur, et on initialise les probabilités a priori sur les nœuds variable, sans avoir besoin de calculer le syndrome.

4.2.3.1 Performances

Comme l'algorithme BP classique, un nombre constant d'itérations $(100-150)$ suffit à faire converger l'algorithme. Cependant, l'hypothèse sur le fait que le graphe est un arbre n'est clairement plus valable : la condition de commutation des stabilisateurs signifie que, pour tous nœuds de parité c_1 et c_2, s'il existe un nœud de variable v et deux arêtes $v \bullet\!\!-\!\!\circ c_1$ et $v \bullet\!\!-\!\!\circ c_2$ de types différents (par exemple, $v \bullet\!\!-\!\!\circ c_1 \in E_X$ et $v \bullet\!\!-\!\!\circ c_2 \in E_Y$), alors il existe *nécessairement* au moins un autre nœud variable v' et deux arêtes $v' \bullet\!\!-\!\!\circ c_1$ et $v' \bullet\!\!-\!\!\circ c_2$ de types différents.

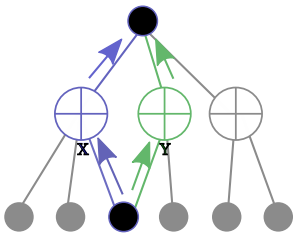


FIGURE 4.9 – Détail de l'algorithme BP avec un 4-cycle sur des arêtes de différents types. Le graphe de Tanner n'est pas un arbre, mais les messages de gauche et de droite sont sur des arêtes de type différents, ainsi, les messages sont peu corrélés (contrairement au cas classique si des 4-cycles sont présents).

Ces cycles sont susceptibles de faire échouer l'algorithme, mais sont moins gênants que dans le cas classique, du fait que les erreurs en X et Z (ou X et Y, ou Y et Z) sont faiblement corrélées (voir l'explication sur la figure 4.9).

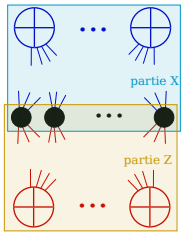


FIGURE 4.10 – Graphe de Tanner d'un graphe CSS : il est en deux parties distinctes d'une seule couleur chacune, donc on peut faire tourner l'algorithme BP classique sur chaque partie.

4.2.4 BP CSS simple

Cette variante de l'algorithme BP est la plus utilisée, et est valable pour les codes CSS. Le graphe de Tanner se compose de deux parties bien distinctes, $\mathscr{G}([\mathscr{C}]_X)$ et $\mathscr{G}([\mathscr{C}]_Z)$ (voir figure 4.10). On peut alors utiliser l'algorithme BP classique sur chacune des parties.

Plus précisément, on initialise la partie en X de la sorte :

$$\begin{aligned} p_v^i(0) = 1 - 2p/3, \quad & p_v^i(1) = 2p/3 \qquad \text{si } E(v) = I \text{ ou } X \\ p_v^i(1) = 2p/3, \quad & p_v^i(1) = 1 - 2p/3 \qquad \text{sinon} \end{aligned}$$

Et de façon similaire en Z. Autrement dit (voir section 3.6 pour plus de détails), on effectue deux décodages séparés sur un canal binaire symétrique de probabilité $p' = 2p/3$.

En séparant les deux décodages, on évite le problème des petits cycles de taille 4 entre nœuds de parité en X et en Z. Mieux encore, lorsque le code est raisonnablement symétrique en Z et X, pour effectuer une simulation de décodage, on peut s'abstenir de faire le décodage en Z et supposer que les performances sont les mêmes qu'en X : on a alors une probabilité d'erreur résultante égale à $2p_{\text{res}} - p_{\text{res}}^2$, où p_{res} est la probabilité d'erreur résultante après le décodage en X seulement.

Par contre, on n'exploite pas les corrélations entre X et Z : la probabilité d'obtenir Y (donc $E(v) \star X = 1$ et $E(v) \star Z = 1$ simultanément) est bien plus élevée ($p/3$) que dans une situation de deux canaux binaire symétriques indépendants (où on aurait la probabilité $p'^2 = 4p/9$). On est donc limité par la borne de hachage CSS, qui est plus faible que la borne de hachage générale, voir la section 3.6 pour plus de détails. Entre autres, pour un rendement $R = 1/4$, on obtient une probabilité d'erreur maximale de 0.1087 au lieu de 0.1269 (borne de hachage).

En termes de performances, en revanche, l'algorithme est plus rapide et utilise moins de mémoire, car on ne fait que dérouler deux algorithmes BP classiques,

l'un après l'autre. Plus précisément, si on note n le nombre de nœuds variables, m le nombre d'arêtes (on supposera que le nombre d'arêtes en X et en Z est $m_X = m_Z = m/2$), d le degré des nœuds variables en considérant les arêtes en X et en Z (on suppose que les degrés en X et en Z seulement sont égaux : $d_X = d_Z = d/2$), et d_c le degré des nœuds de parité :

- Pour la mise à jour des q_e : il y a deux valeurs à calculer ($q_e(0)$ et $q_e(1)$) pour chaque arête. Comme on est dans le cas classique, il s'agit d'un produit de d_X termes (voir l'équation (1.11), dans la section 1.3.2) ; on a donc $2(m_X d_X + m_Z d_Z) = 2m/2d/2 + 2m/2d/2 = md$ opérations à effectuer pour une itération (on tient compte du fait qu'il faut faire ce calcul deux fois : une fois pour X et une fois pour Z). Alors que dans le cas général (voir l'équation (4.9)), on doit calculer une somme de deux produits, chacun comportant d termes : $2md$. On a donc au total un temps deux fois plus court pour ces mises à jour.
- La mise à jour des p_v^f, quand à elle, ne gagne pas de temps : on effectue deux fois 2 mises à jour au lieu de 4 (X et Z séparés, et les 4 grandeurs d'un coup dans le cas général). Or le calcul des p_v^f en classique est un produit de $d_X + 1$ (resp $d_Z + 1$) termes (voir l'équation (1.8)), au lieu d'un produit de $d_X + d_Z + 1$ termes dans le cas quantique (voir l'équation (4.7)). On a donc un nombre d'opérations de l'ordre de $n(d_X + d_Z + 2) = n(d + 2)$ face à $n(d_X + d_Z + 1) = n(d + 1)$.
- La mise à jour des r_e est similaire dans les deux cas (le degré est identique, et il y a 2 valeurs à calculer).
- Côté mémoire, les q_e et r_e prennent deux fois moins de place, car on effectue les deux décodages l'un après l'autre, et donc on peut réutiliser la mémoire.
- Les p_v^i et p_v^f prennent 2 fois moins de place. En effet, pour les p_v^i ou les p_v^f, il y a 4 probabilités dans le cas général, on doit donc en stocker 3 (la quatrième étant déduite du fait que la somme fait 1), et dans le cas classique, il y a seulement 2 probabilités : il suffit de n'en stocker qu'une seule. On a donc besoin d'un espace en $3n + 3n$ pour les p_v^i et p_v^f du cas général, alors que dans le cas simple, il faut stocker les deux vecteurs p_v^i (donc $2n$), et d'un seul vecteur final (donc n) : en effet, si on commence par le calcul en X, on peut utiliser le vecteur des p_v^i en X pour stocker les p_v^f une fois celui-ci effectué.

Ainsi, on réduit le temps d'exécution d'un facteur environ 3/4 (si on suppose que les q_e et r_e prennent un temps équivalent et que les p^f ne sont pas mis à jour à chaque fois), et la mémoire d'un facteur 2. Dans la pratique, lors des simulations, on divise encore par deux le temps d'exécution, car on n'effectue que la moitié du décodage (en supposant que les parties X et Z ont les mêmes performances).

Il ne faut pas négliger non plus la simplicité de l'algorithme : le BP classique a déjà été largement étudié, on connaît donc déjà ses forces et faiblesses, ainsi que les différentes façons de l'optimiser, qu'on peut directement utiliser ici.

4.2.5 BP CSS mixte

Cet algorithme n'est rien de plus qu'un cas particulier de l'algorithme général, appliqué à un code CSS. Dans cette version particulière, il est un peu plus rapide que l'algorithme général, même s'il prend un peu plus d'espace mémoire, par contre il permet en principe d'atteindre la borne de hachage, car on y tient compte de la corrélation entre X et Z.

À première vue, il n'est pas plus intéressant que l'algorithme général, car les problèmes de 4-cycles réapparaissent. Cependant, il a été testé dans un cas particulier de graphe de Tanner, où ces 4-cycles n'apparaissent pas directement (voir dans la section 7.3) : les performances sont alors très bonnes. Il est également légèrement plus simple que le BP général à implémenter.

Lors de la mise à jour d'un message q_e (message quittant un nœud variable pour aller vers un nœud de parité), pour une arête $e = v \bullet\!\!-\!\!\bullet c \in E_X$, on peut écrire :

$$\begin{aligned} q_e(0) &= \alpha \left(p_v^i(X) \prod_{E_X \setminus e} r_{e'}(0) \prod_{E_Z} r_{e'}(1) + p_v^i(I) \prod_{E_X \setminus e} r_{e'}(0) \prod_{E_Z} r_{e'}(0) \right) \\ &= \alpha \left(p_v^i(X) \prod_{E_Z} r_{e'}(1) + p_v^i(I) \prod_{E_Z} r_{e'}(0) \right) \prod_{E_X \setminus e} r_{e'}(0) \end{aligned}$$

Or si on compare cette expression à celle du calcul des q_e dans le cas classique, à savoir :

$$q_e(0) = \alpha' p_v^i(0) \prod_{E \setminus e} r_{e'}(0)$$

on constate que, hormis les facteurs α et α' qui correspondent à la renormalisation, on peut voir $p_v^i(X)\prod_{E_Z} r_{e'}(1) + p_v^i(I)\prod_{E_Z} r_{e'}(0)$ comme une sorte de $p_v^i(0)$ virtuel pour l'algorithme BP classique, d'autant que cette grandeur ne change pas pour toutes les mises à jour d'arêtes de type X. On peut constater que cette grandeur représente aussi (à renormalisation près) $\mathbb{P}[E(v) \star X = 0 \mid \text{arêtes en Z seulement}]$: si les erreurs X et Z étaient indépendantes, on ne tirerait aucune information intéressante de cette grandeur (en fait, cette valeur serait constante), mais pour le canal dépolarisant ce n'est pas le cas.

Plus généralement, on effectue une construction équivalente à la construction de la figure 4.11.

Comme le but est d'utiliser le BP classique, on renomme q_e en qx_e si $e \in E_X$, et en qz_e sinon, et de même pour r_e. Pour $v \in V$, $b \in \{0, 1\}$, on définit deux nouveaux messages :

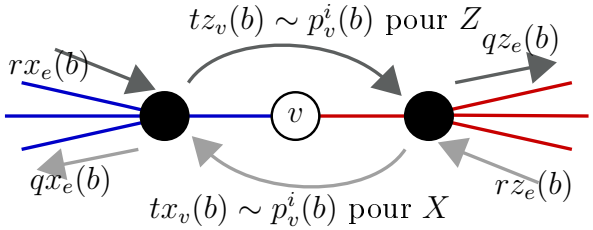


FIGURE 4.11 – Schéma du BP CSS mixte : les parties X et Z sont gérées par un algorithme BP classique, mais les messages tx et tz permettent d'échanger un peu d'information entre les parties X et Z. $qx_e(b)$ représente la grandeur $q_e(b)$ (pour le BP classique) pour la partie en X, et de même pour les autres grandeurs : qz, rz, rx.

$$tx_v(b) = \sum_{T \star X = b} p_v^i(T) \prod_{e' \in \mathcal{V}_{E_Z}(v)} rz_{e'}(Z \star T) \tag{4.11}$$

$$tz_v(b) = \sum_{T \star Z = b} p_v^i(T) \prod_{e' \in \mathcal{V}_{E_X}(v)} rx_{e'}(X \star T) \tag{4.12}$$

On a alors :

$$\forall e \in E_X, b \in \{0,1\}, \quad qx_e(b) \propto tx_v(b) \prod_{e' \in \mathcal{V}_{E_X}(v) \setminus e} rx_{e'}(b) \tag{4.13}$$

$$\forall e \in E_Z, b \in \{0,1\}, \quad qz_e(b) \propto tz_v(b) \prod_{e' \in \mathcal{V}_{E_Z}(v) \setminus e} rz_{e'}(b) \tag{4.14}$$

En effet, pour $e = v \multimap c \in E_X$ et $b \in \{0,1\}$, en utilisant la formule (4.9) :

$$\begin{aligned} qx_e(b) &\propto \sum_{T, T \star X = b} p_v^i(T) \prod_{e' \in \mathcal{V}_{E_X}(v) \setminus e} rx_{e'}(T \star X) \prod_{e' \in \mathcal{V}_{E_Z}(v)} rz_{e'}(T \star Z) \\ &\propto \left(\prod_{e' \in \mathcal{V}_{E_X}(v) \setminus e} rx_{e'}(b) \right) \left(\sum_{T, T \star X = b} p_v^i(T) \prod_{e' \in \mathcal{V}_{E_Z}(v)} rz_{e'}(T \star Z) \right) \\ &\propto \left(\prod_{e' \in \mathcal{V}_{E_X}(v) \setminus e} rx_{e'}(b) \right) tx_v(b) \end{aligned}$$

Et de même pour qz.

La fonction de mise à jour des q_e se fait donc comme dans le BP classique (en prenant tx_v ou tz_v comme vecteur de probabilités initiales), et la fonction de mise

à jour des r_e ne change pas puisque les nœuds de parité sont d'un seul type (X ou Z).

La mise à jour finale doit cependant être faite avec la formule d'origine, autrement dit :

$$p_v^f(T) \propto p_v^i(T) \prod_{e' \in E_X} r_{e'}(T \star X) \prod_{e' \in E_Z} r_{e'}(T \star Z)$$

L'algorithme se réécrit alors :

Algorithme Propagation de croyances, version CSS mixte
{ *Initialisation* }
Initialiser les p_v^i et les p_v^f
$tx_v(0) \leftarrow p_v^i(X) + p_v^i(I)$
$tx_v(1) \leftarrow p_v^i(Z) + p_v^i(Y)$
$tz_v(0) \leftarrow p_v^i(Z) + p_v^i(I)$
$tz_v(1) \leftarrow p_v^i(X) + p_v^i(Y)$
Initialiser les rx_e et rz_e à 1.
{ *Déroulement* }
TantQue nombre d'itérations $\leqslant$ NB_MAX, et que E change **Faire**
  Effectuer une itération du BP classique pour X (calcul des qx_e puis des rx_e)
  Effectuer une itération du BP classique pour Z
  Mettre à jour les messages croisés tx_v et tz_v
  Si nombre d'itérations mod $\delta = 0$ **Alors**
    Mettre à jour les p_v^f avec (4.7)
    Mettre à jour E l'erreur : $E(v) = \underset{T}{\operatorname{argmax}}(p_v^f(T))$
  FinSi
FinTantQue
Retourner E

En termes d'espace mémoire, il faut stocker les mêmes grandeurs que dans le cas général, et il faut y ajouter les vecteurs tx et tz, qui prennent une place $2n$. Pour la complexité en temps :

- La mise à jour des r_e est identique au cas général, tout comme celle des p^f.
- La mise à jour des q_e est plus rapide : une fois les tz et tx calculés, la mise à jour se fait comme dans le BP classique (de chaque côté) : $2m(d_X + d_Z) = md$ au lieu de $2md$ pour le cas général.
- La mise à jour des tx et tz, pour une itération, un nœud variable et un bit

données, consiste à faire la somme de deux produits de $d_X + 1$ termes (ou $d_Z + 1$ termes). On a donc $n(d_X + d_Z + 2) = n(d + 2)$.

Comme n est beaucoup plus faible que m (d'un facteur d_X ou d_Z), le temps perdu à mettre à jour les tx et tz est largement compensé par le temps gagné (un facteur 2 sur la mise à jour des q_e).

4.2.6 BP CSS asymétrique

Cette variante consiste à prendre l'algorithme BP CSS simple, en ajoutant quelques informations entre les parties X et Z. Il a l'avantage d'utiliser les corrélations entre les deux parties (ce que ne fait pas le BP CSS simple), tout en restant dans le même schéma que le BP CSS simple, avec les avantages qui en découlent (en terme de complexités spatiale et temporelle). En revanche, il n'est réellement intéressant que sur des codes CSS asymétriques : où un des rendements (R_X ou R_Z) est plus élevé que l'autre.

4.2.6.1 Déroulement

Le schéma global de l'algorithme est le suivant :

1. Initialiser les p_v^i de la partie en X à partir de l'erreur reçue, donnée sous forme de vecteur de probabilités $pq_v(T), \forall T \in E_1$,
2. Résoudre la partie en X (décodage BP simple), en déduire les p_v^f,
3. Initialiser les $p_v'^i$ de la partie en Z, à partir des pq_v et des p_v^f,
4. Résoudre la partie en Z (décodage BP simple),
5. En déduire l'erreur finale

L'étape 1 se fait de la façon suivante :

$$\begin{aligned} p_v^i(0) &= pq_v(I) + pq_v(X) \\ p_v^i(1) &= pq_v(Y) + pq_v(Z) \end{aligned}$$

Pour l'étape 3, il faut d'abord ré-estimer les probabilités d'obtenir chaque erreur en fonction des données reçues (les p_v^f) de la partie X :

$$\begin{aligned} \mathbb{P}[E(v) = I] &\propto p_v^f(0) pq_v(I) \\ \mathbb{P}[E(v) = X] &\propto p_v^f(0) pq_v(X) \\ \mathbb{P}[E(v) = Y] &\propto p_v^f(1) pq_v(Y) \\ \mathbb{P}[E(v) = Z] &\propto p_v^f(1) pq_v(Z) \end{aligned}$$

Il faut renormaliser pour obtenir les bonnes valeurs, on appellera α le facteur qui permet d'assurer que la somme des probabilités soit égale à 1.

On en déduit les équations pour l'étape 3 :

$$\begin{aligned} p_v'^i(0) &= \mathbb{P}[E(v) = I] + \mathbb{P}[E(v) = Z] \qquad (4.15) \\ &= \alpha \left(p_v^f(0) pq_v(I) + p_v^f(1) pq_v(Z) \right) \\ p_v'^i(1) &= \mathbb{P}[E(v) = X] + \mathbb{P}[E(v) = Y] \qquad (4.16) \\ &= \alpha \left(p_v^f(0) pq_v(X) + p_v^f(1) pq_v(Y) \right) \end{aligned}$$

Comme le même facteur de normalisation apparaît, il n'est pas nécessaire de renormaliser au milieu, on peut effectuer cette opération à la fin.

À l'étape 5, on déduit l'erreur finale de la même façon que dans le BP CSS simple.

4.2.6.2 Performances

Le temps d'exécution est identique à celui du BP CSS simple, si ce n'est qu'il faut ajouter l'initialisation de la partie en Z : comme cette initialisation n'est à faire qu'une fois pour le décodage, la différence est assez négligeable.

L'espace mémoire utilisé est du même ordre de grandeur que pour le BP CSS simple, mis à part que, comme on a besoin du max de l'espace utilisé pour les deux décodages et que les deux codes sont asymétriques, cela nécessite un peu plus d'espace.

Quand à la borne maximale qu'on peut atteindre, il faut regarder un peu plus précisément ce qui se passe pour chacun des décodages. Soit p_m la probabilité maximale qu'on peut atteindre pour un rendement R donné ($R = R_X + R_Z - 1$). Pour le premier décodage, on utilise un canal binaire symétrique de probabilité $p' = 2p_m/3$, on a donc, avec l'équation (1.1) et le théorème de Shannon (théorème 1) :

$$R_X = 1 - h(1 - 2p_m/3, 2p_m/3) \qquad (4.17)$$

Le second canal est un peu moins simple : il s'agit d'un mélange entre un canal à effacement et un canal binaire symétrique, qu'on appelle *canal dépolarisant corrigé* de paramètre p :

Définition 43. *Le* canal dépolarisant corrigé *de paramètre p est défini de l'alphabet $\{0,1\}$ vers $\{0,1,e\}$ de la façon suivante :*

$$\begin{aligned} p(1 \mid 1) = p(0 \mid 0) &= 1 - p \\ p(0 \mid 1) = p(1 \mid 0) &= p/3 \\ p(e \mid 0) = p(e \mid 1) &= 2p/3 \end{aligned}$$

On peut le voir de la façon suivante : si la partie en X est parfaitement décodée, alors :

- si on avait tiré le symbole I (probabilité $1-p$), alors le décodage en X confirme,
- si on avait tiré le symbole Y ou Z (probabilité $2p/3$ pour les deux), le décodage en X infirme cette hypothèse, et on a un effacement,
- si on avait tiré le symbole X (probabilité $p/3$), le décodage en X confirme également cette erreur.

Plus précisément, dans l'algorithme, on vérifie qu'on a bien l'équivalent de ce canal : supposons qu'on parte avec le symbole I. Le décodage de la partie en X étant supposé se passer parfaitement, on obtient $p_v^f(0) = 1$ et $p_v^f(1) = 0$. Avec les équations (4.15) et (4.16), on obtient donc :

$$\begin{aligned} p_v'^i(0) &\propto pq_v(I) \\ p_v'^i(1) &\propto pq_v(X) \end{aligned}$$

- S'il advient une erreur X (probabilité $p/3$), on aboutit à $p_v'^i(0) \propto p/3$ et $p_v'^i(1) \propto 1-p$, ce qui donne $p_v'^i(0) = \frac{p/3}{1-2p/3}$ et $p_v'^i(1) = \frac{1-p}{1-2p/3}$, ce qui correspond bien à un cas de bit qui passe de 0 à 1,
- S'il advient une erreur Y ou Z (probabilité cumulée $2p/3$), on a $p_v'^i(0) \propto p/3$ et $p_v'^i(1) \propto p/3$, donc $p_v'^i(0) = p_v'^i(1) = 1/2$, ce qui correspond bien à un effacement,
- S'il advient une erreur I (probabilité $1-p$), on aboutit à $p_v'^i(0) \propto 1-p$ et $p_v'^i(1) \propto p/3$, donc $p_v'^i(0) = \frac{1-p}{1-2p/3}$ et $p_v'^i(1) = \frac{p/3}{1-2p/3}$, ce qui correspond bien au cas du bit inchangé.

Il suffit donc de connaître la capacité de ce canal particulier pour obtenir la borne finale.

Propriété 22 (Capacité du canal dépolarisant corrigé)**.** *La capacité du canal dépolarisant corrigé de paramètre p vérifie :*

$$C = 1 - 2p/3 - (1-2p/3)\log(1-2p/3) + (1-p)\log(1-p) + p/3\log(p/3)$$

Démonstration. On rappelle que la capacité d'un canal classique est donnée par la formule (voir définition 9) :

$$C = \sup_{X,Y} I(X;Y) = H(Y) - H(Y \mid X)$$

Or le canal en question est *symétrique* : informellement, cela signifie que le canal agit de la même manière sur 0 et 1, et cela permet d'estimer plus facilement la capacité de ce canal. Plus précisément, la définition est la suivante :

Définition 44 (Canal symétrique)**.** *Un canal sans mémoire de $\mathcal{A}$ vers $\mathcal{B}$ est dit* symétrique *si l'alphabet $\mathcal{B}$ peut être partitionné en $\mathcal{B}_1, \dots \mathcal{B}_m$ de telle sorte que, pour chaque $\mathcal{B}_i$, la matrice de transition des $(p(y \mid x))_{x\in\mathcal{A}, y\in\mathcal{B}_i}$ vérifie que toutes les lignes sont des permutations des autres lignes, et de même pour les colonnes (s'il y a plus d'une colonne).*

Ici, on a bien un canal symétrique, puisque si on écrit $\mathcal{B} = \{0, 1\} \cup \{e\}$, on a les deux matrices de transition suivantes :

$$\begin{pmatrix} 1-p & p/3 \\ p/3 & 1-p \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 2p/3 \\ 2p/3 \end{pmatrix}$$

On peut également remarquer que le canal binaire symétrique et le canal à effacements (voir section 1.1.1) sont symétriques.

Or, lorsqu'on a affaire à un tel canal, on peut utiliser le théorème suivant pour trouver la capacité :

Théorème 3 (Capacité d'un canal symétrique [Gal68])**.** *Pour un canal symétrique sans mémoire, la capacité est atteinte pour une distribution équiprobable des symboles en entrée.*

On peut donc supposer que X est équirépartie (c'est-à-dire que $\mathbb{P}[X = 0] = \mathbb{P}[X = 1] = 1/2$), et en déduire les probabilités pour Y :

$$\mathbb{P}[Y = 0] = \frac{1-p}{2} + \frac{p/3}{2} = \frac{1-2p/3}{2} \tag{4.18}$$

$$\mathbb{P}[Y = 1] = \frac{1-2p/3}{2} \tag{4.19}$$

$$\mathbb{P}[Y = e] = 2p/3 \tag{4.20}$$

En effet, pour obtenir 0 comme symbole, il y a deux possibilités : avoir envoyé 0 (probabilité $1/2$) et avoir reçu 0 (probabilité $1-p$), ou avoir envoyé 1 (probabilité $1/2$) qui s'est transformé en 0 (probabilité $p/3$), d'où (4.18). Pour (4.19), c'est la même chose en symétrique. Pour obtenir e comme symbole, on peut avoir envoyé 0 (probabilité $1/2$) et qu'il y ait eu effacement (probabilité $2p/3$), ou avoir envoyé 1 (probabilité $1/2$) et avoir eu un effacement (probabilité $2p/3$), d'où (4.20).

On a donc :

$$\begin{aligned} H(Y) &= -2p/3 \log(2p/3) - 2\frac{1-2p/3}{2} \log\left(\frac{1-2p/3}{2}\right) \\ &= -2p/3 \log(2p/3) - (1-2p/3) \log\left(\frac{1-2p/3}{2}\right) \\ &= -2p/3 \log(2p/3) - (1-2p/3) \log(1-2p/3) + 1 - 2p/3 \end{aligned}$$

On calcule ensuite $H(Y \mid X)$:

$$\begin{aligned} H(Y \mid X) &= -\sum_{x,y} \mathbb{P}[Y = y \mid X = x] \log(\mathbb{P}[Y = y \mid X = x])\mathbb{P}[X = x] \\ &= -\sum_{y} \mathbb{P}[Y = y \mid x = 0] \log(\mathbb{P}[Y = y \mid X = 0]) \\ &= -(1-p)\log(1-p) - p/3\log(p/3) - 2p/3\log(2p/3) \end{aligned}$$

On a donc :

$$\begin{aligned} I(X;Y) &= -2p/3\log(2p/3) - (1-2p/3)\log(1-2p/3) + 1 - 2p/3 \\ &\quad +(1-p)\log(1-p) + p/3\log(p/3) + 2p/3\log(2p/3) \\ &= 1 - 2p/3 - (1-2p/3)\log(1-2p/3) + (1-p)\log(1-p) \\ &\quad +p/3\log(p/3) \end{aligned}$$

■

On peut alors obtenir la borne finale :

$$\begin{aligned} R &= R_X - 1 + R_Z \\ &= 1 - h(1-2p_m/3, 2p_m/3) - 1 \\ &+ 1 - 2p_m/3 - (1-2p_m/3)\log(1-2p_m/3) + (1-p_m)\log(1-p_m) \\ &+ p_m/3\log(p_m/3) \\ &= 1 + (1-p_m)\log(1-p_m) - 2p_m/3 + 2p_m/3\log(2p_m/3) \\ &+ p_m/3\log(p_m/3) \\ &= 1 + (1-p_m)\log(1-p_m) - 2p_m/3 + 2p_m/3\log 2 \\ &+ (2p_m/3 + p_m/3)\log(p_m/3) \\ &= 1 - h(1-p_m, p_m/3, p_m/3, p_m/3) \end{aligned}$$

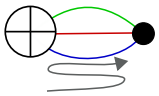
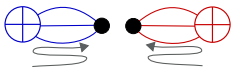
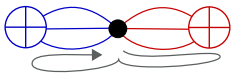
Ce qui donne la borne de hachage (voir équation (3.30)) : ainsi, ce résultat montre que l'algorithme BP CSS asymétrique peut atteindre la borne de hachage (sous l'hypothèse que le BP se déroule parfaitement).

4.2.7 Comparaison

On donne ici un récapitulatif des différents algorithmes vus dans cette section. Ils ont tous en commun les propriétés suivantes :

- Un temps d'exécution et utilisation mémoire en $O(n)$, où n est la longueur du code
- La dégénérescence n'est pas utilisée, autrement dit on ne peut pas dépasser la borne de hachage.

Les différences sont mises en valeur dans le tableau 4.1. La liste n'est bien entendu pas exhaustive, il existe d'autres variantes.

Algorithme	Avantages	Inconvénients
BP général 	• Valable pour tout code LDPC quantique • Utilise la corrélation entre X et Z, et peut atteindre la borne de hachage	• Présence de petits cycles de taille 4 dans le graphe de Tanner (a priori)
BP CSS simple 	• Complexité spatiale deux fois plus faible que le BP général • Complexité temporelle légèrement plus faible que le BP général (environ 3/4) • Pas de problèmes de petits cycles de taille 4	• Valable uniquement sur les codes CSS • N'utilise pas la corrélation entre X et Z : performances limitées par la borne de hachage CSS.
BP CSS mixte 	• Utilise la corrélation entre X et Z, et peut atteindre la borne de hachage • Complexité temporelle légèrement inférieure au BP général	• Valable uniquement sur les codes CSS • Présence de petits cycles de taille 4 dans le graphe de Tanner (a priori)

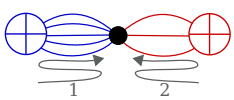
BP CSS asymétrique 	• Utilise la corrélation entre X et Z, et peut atteindre la borne de hachage • Complexité spatiale environ deux fois plus faible que le BP général • Complexité temporelle légèrement plus faible que le BP général (environ 3/4) • Pas de problèmes de petits cycles de taille 4	• Nécessite un code CSS *asymétrique* d'une façon bien précise pour être efficace

Tableau 4.1: Récapitulatif des différents algorithmes BP, avec leurs avantages et inconvénients

Chapitre 5

Codes obtenus par produit

Ce chapitre présente une première construction de codes LDPC quantiques, introduits originellement dans [TZ09], obtenue par produit de deux codes LDPC classiques. Après avoir présenté la construction et ses principales propriétés (dimension, distance minimale), on y présente ses performances pratiques, et une première variante, le code produit tridimensionnel (dans la section 5.4). Deux autres variantes sont étudiées, dans les chapitres suivants : le code produit q-aire (section 6.3) et le code produit spatialement couplé (section 7.2).

On a vu dans le chapitre 4 que les codes LDPC quantiques ne sont pas aussi performants que leurs équivalents classiques, si tant est qu'on puisse les comparer. Notamment, peu de constructions combinent à la fois une distance minimale non bornée et un rendement constant, alors qu'il est très simple de construire des codes classiques ayant un rendement constant et une distance minimale linéaire.

On s'intéresse ici à une de ces exceptions : la construction de codes produits, introduite par J-P. Tillich et G. Zémor dans [TZ09], dont les codes possèdent à la fois une distance minimale en $\sqrt{n}$ (où n est la longueur du code), et un rendement constant lorsque n augmente. De plus, ces codes se construisent à partir d'un couple de codes LDPC classiques quelconques, *sans aucune condition d'orthogonalité*, ce qui permet une grande liberté dans la construction et une facilité à les générer en pratique.

Malheureusement, on constate que ces codes, bien que très prometteurs, ont des performances en pratique (sous l'algorithme BP) très décevantes. C'est pourquoi quelques variantes sont expérimentées dans les sections suivantes, jusqu'ici sans succès, ou au mieux sans résultat : il faudrait plutôt jouer sur l'algorithme de décodage et la dégénérescence pour profiter de ses bonnes propriétés.

5.1 Le code produit

Le produit de codes présenté ici est une technique pour obtenir un code quantique CSS *valide* (c'est à dire vérifiant la condition d'orthogonalité) à partir de deux codes classiques absolument quelconques : aucune condition d'orthogonalité, entre autres, n'est requise. Les propriétés du code produit (dimension, distance minimale) sont héritées de celles des codes composants : cela signifie, grossièrement, qu'il suffit de prendre de raisonnablement « bons » codes composants pour obtenir un raisonnablement « bon » code quantique. On obtient donc une construction avec une grande liberté de choix, puisqu'il est facile de construire des bons codes classiques. De plus, on le verra un peu plus bas, le fait d'être LDPC se transmet également : c'est donc un excellent moyen de construire des codes LDPC quantiques.

5.1.1 Définition

Il est assez pratique de considérer la construction du point de vue des graphes de Tanner : pour un code classique $\mathscr{C}$, on associera donc son graphe de Tanner $\mathscr{G}(\mathscr{C}) = (V, C, E)$ (voir section 1.3.1). De même, à un code CSS $\mathscr{C}_q$, on associera $\mathscr{G}(\mathscr{C}_q) = (V, C_X, C_Z, E_X, E_Z)$ les nœuds variables, les nœuds de parité et les arêtes en X et en Z (voir section 4.1.2).

Définition 45 (Code produit ([TZ09])). *Soient deux codes classiques $\mathscr{C}_1$ et $\mathscr{C}_2$ et leurs graphes de Tanner $\mathscr{G}(\mathscr{C}_1) = (V_1, C_1, E_1)$ et $\mathscr{G}(\mathscr{C}_2) = (V_2, C_2, E_2)$ qu'on appellera* codes composants. *On définit le* code produit $\mathscr{C}_q = \mathscr{C}_1 \boxtimes \mathscr{C}_2$ *ayant pour graphe de Tanner $\mathscr{G}(\mathscr{C}_q) = (V, C_X, C_Z, E_X, E_Z)$ de la façon suivante :*

- $V = V_1 \times V_2 \cup C_1 \times C_2$
- $C_X = C_1 \times V_2$
- $C_Z = V_1 \times C_2$
- $E_X = \{(v_1, v_2) \multimap (c_1, v_2) |\ v_1 \multimap c_1 \in E_1, v_2 \in V_2\}$
  $\cup\ \{(c_1, c_2) \multimap (c_1, v_2) |\ v_2 \multimap c_2 \in E_2, c_1 \in C_1\}$
- $E_Z = \{(v_1, v_2) \multimap (v_1, c_2) |\ v_2 \multimap c_2 \in E_2, v_1 \in V_1\}$
  $\cup\ \{(c_1, c_2) \multimap (v_1, c_2) |\ v_1 \multimap c_1 \in E_1, c_2 \in C_2\}$

Cette construction est résumée sur la figure 5.1. Il faut encore vérifier que ce code est un code quantique valide :

Propriété 23 (Validité du code ([TZ09])). *Le code quantique $\mathscr{C}_q = \mathscr{C}_1 \boxtimes \mathscr{C}_2$ de la définition 45 est un code quantique valide : pour tous $c_X \in C_X, c_Z \in C_Z$,*

$$\mathcal{S}(c_X) \star \mathcal{S}(c_Z) = 0$$

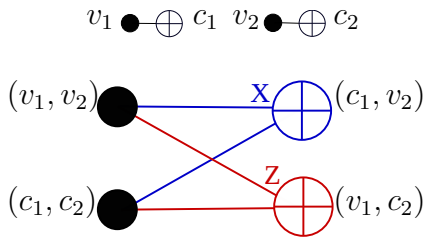


FIGURE 5.1 – Schéma de la construction du code produit (voir définition 45) : pour chaque paire d'arêtes de $\mathscr{C}_1$ et $\mathscr{C}_2$, on obtient un « papillon » d'arêtes. Pour rappel, les arêtes bleues représentent les arêtes en X, les rouges en Z.

Démonstration. Soient deux nœuds de parité $c_X \in C_X$ et $c_Z \in C_Z$.

On a $c_X = (c_1, v_2)$ pour $c_1 \in C_1$ et $v_2 \in V_2$, $c_Z = (v_1, c_2)$ pour $v_1 \in V_1$ et $c_2 \in C_2$ avec les notations de la définition 45.

Soit une position sur laquelle $\mathcal{S}(c_1, v_2)$ et $\mathcal{S}(v_1, c_2)$ anti-commutent. Par symétrie dans la construction, on peut supposer qu'il s'agit d'un nœud de variable de $V_1 \times V_2$: c'est donc nécessairement (v_1, v_2), et il ne peut pas y avoir d'autre telle position par définition.

Cela signifie qu'on a $(v_1, v_2) \rightarrowtail (c_1, v_2) \in E_X$ et $(v_1, v_2) \rightarrowtail (v_1, c_2) \in E_Z$. Il existe donc deux arêtes des codes $\mathscr{C}_1$ et $\mathscr{C}_2$ telles que : $v_1 \rightarrowtail c_1 \in E_1$ et $v_2 \rightarrowtail c_2 \in E_2$. Par construction du code produit, on a également deux arêtes $(c_1, c_2) \rightarrowtail (c_1, v_2) \in E_X$ et $(c_1, c_2) \rightarrowtail (v_1, c_2) \in E_Z$. Comme (c_1, c_2) est le seul nœud de variable de $C_1 \times C_2$ qui puisse être relié à ces deux nœuds de parité, les deux stabilisateurs anti-commutent en exactement deux positions, et donc commutent.

■

On peut également voir les nœuds variables de $\mathscr{C}_q$ comme une paire de matrices de $\mathbb{F}_4$ (indexées par V_1 et V_2 pour la première, C_1 et C_2 pour la seconde), dont les colonnes sont « vérifiées » par des nœuds de parité en X et les lignes par des nœuds de parité en Z dans la première matrice, et l'inverse dans la seconde (voir figure 5.2).

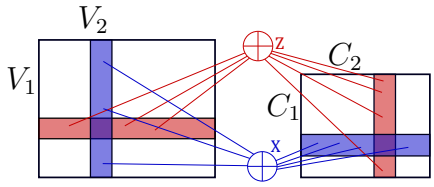


FIGURE 5.2 – Représentation de l'ensemble V du code produit sous forme de paire de matrices. On peut remarquer qu'un nœud de parité « vérifie » une ligne de l'une et une colonne de l'autre uniquement.

5.1.2 Propriétés

En plus de la simplicité de construction, le code produit a deux principaux avantages : pour peu qu'on choisisse correctement les codes composants, on peut obtenir un code quantique ayant à la fois un rendement constant pour $n \to \infty$, et une distance minimale qui croît en $\sqrt{n}$.

La dimension est relativement naturelle : lorsqu'on choisit par exemple pour composants des codes sont tous les nœuds de parité sont indépendants, on obtient un code quantique dont tous les nœuds de parité sont également indépendants.

Théorème 4 (Dimension du code produit ([TZ09]))**.** *Soient deux codes classiques $\mathscr{C}_1$ et $\mathscr{C}_2$. On note k_1 et k_2 leurs dimensions respectives, et $\bar{k}_1$ et $\bar{k}_2$ leurs dimensions duales respectives. La dimension du code $\mathscr{C}_1 \boxtimes \mathscr{C}_2$ est :*

$$k = k_1 k_2 + \bar{k}_1 \bar{k}_2$$

Où la dimension duale d'un code est définie de la façon suivante :

Définition 46 (Dimension duale d'un code)**.** *Soit $\mathscr{C}$ un code de longueur n et de dimension k, donné par sa matrice de parité $\mathbf{H}$ de dimensions $r \times n$. Sa* dimension duale *, notée $\bar{k}$, est par définition :*

$$\bar{k} = r - n + k$$

Cette dimension duale correspond à la dimension des nœuds de parité :

Propriété 24. *Soit $\mathscr{C}$ défini par son graphe de Tanner $\mathscr{G}(\mathscr{C}) = (V, C, E)$, avec n nœuds variable, r nœuds de parité et une dimension k. Alors la dimension duale correspond à la dimension de $\bar{\mathscr{C}}$ tel que $\mathscr{G}(\mathscr{C}) = (C, V, E)$.*

Démonstration. $\bar{\mathscr{C}}$ correspond au code $\mathscr{C}$ dont on a inversé les nœuds variable et les nœuds de parité. Sa matrice de parité est donc ${}^{\mathbf{t}}(\mathbf{H})$. Or la dimension d'un code est sa longueur moins le rang de la matrice de parité. Ici, cela nous donne $\dim \bar{\mathscr{C}} = r - \text{rang}({}^{\mathbf{t}}(\mathbf{H})) = r - \text{rang}(\mathbf{H}) = r - (n - k)$, ce qui donne bien $\bar{k}$. ■

On peut noter que, si on prend $n > r$, la dimension duale est souvent nulle.

On peut maintenant prouver le théorème :

Preuve du théorème 4. On note $\#V_1 = n_1, \#V_2 = n_2, \#C_1 = r_1, \#C_2 = r_2$. La dimension du code $\mathscr{C}_1 \boxtimes \mathscr{C}_2$ s'exprime

$k =$ nombre de nœuds variables $-$ nombre de nœuds de parité *indépendants*

Le nombre de nœuds variable est $\#(V_1 \times V_2 \cup C_1 \times C_2) = n_1 n_2 + r_1 r_2$, le nombre de nœuds de parité *total* est $\#(C_1 \times V_2 \cup V_1 \times C_2) = r_1 n_2 + n_1 r_2$. Il reste à calculer le nombre de nœuds de parité redondants.

Si on regarde uniquement la partie en X (partie bleue sur la figure 5.1), le nombre de nœuds de parité redondants correspond à la dimension du code *classique* $\tilde{\mathscr{C}_X}$ de graphe de Tanner $\mathscr{G}(\tilde{\mathscr{C}_X}) = (C_1 \times V_2, V, E_X)$, autrement dit la partie X du code, avec les nœuds variables en nœuds de parité et vice-versa.

Ce code correspond à ce qu'on appelle en classique un code produit, qu'on appellera ici *code produit classique* pour éviter les confusions. Plus précisément il est le produit classique de $\mathscr{C}_1 = (C_1, V_1, E_1)$ et de $\mathscr{C}_2$. En effet, il y a une arête $(c_1, v_2) \rightarrowtail (v_1, v_2) \in E_X$ si et seulement si il y a une arête $c_1 \rightarrowtail v_1 \in E_1$, et une arête $(c_1, v_2) \bullet\!\!\rightarrowtail (c_1, c_2) \in E_X$ si et seulement si il y a une arête $v_2 \bullet\!\!\rightarrowtail c_2 \in E_2$. Pour ces codes (voir [MS78] pour plus de détails), on sait que la dimension finale est le produit des dimensions des codes composants, donc ici :

$$\dim(\tilde{\mathscr{C}_X}) = \dim(\bar{\mathscr{C}_1}) \dim(\mathscr{C}_2) = \bar{k}_1 k_2$$

Par symétrie, on obtient le nombre de nœuds redondants de la partie en Z : $k_1 \bar{k}_2$.

On rappelle que, pour un code classique, la dimension duale vérifie $k = n - r + \bar{k}$. La dimension finale est donc :

$$\begin{aligned} k &= n_1 n_2 + r_1 r_2 - r_1 n_2 - n_1 r_2 + \bar{k}_1 k_2 + k_1 \bar{k}_2 \\ &= (n_1 - r_1)(n_2 - r_2) + \bar{k}_1 (n_2 - r_2 + \bar{k}_2) + k_1 (-n_2 + r_2 + k_2) \\ &= (n_1 - r_1)(n_2 - r_2) + (\bar{k}_1 - k_1)(n_2 - r_2) + \bar{k}_1 \bar{k}_2 + k_1 k_2 \\ &= (n_1 - r_1)(n_2 - r_2) + (r_1 - n_1)(n_2 - r_2) + \bar{k}_1 \bar{k}_2 + k_1 k_2 \\ &= \bar{k}_1 \bar{k}_2 + k_1 k_2 \end{aligned}$$

■

Avec cette propriété, si on choisit deux familles de codes classiques de rendements constants R_1 et R_2 et de dimensions duales nulles, le rendement du code produit vaut :

$$R = \frac{k}{n} = \frac{R_1 n_1 R_2 n_2}{n_1 n_2 + (1 - R_1)(1 - R_2) n_1 n_2} = \frac{R_1 R_2}{2 - R_1 - R_2 + R_1 R_2}$$

Ce rendement est donc constant et indépendant des longueurs respectives des codes.

La distance minimale est une caractéristique importante du code, même si une distance élevée ne donne pas forcément des performances au décodage (surtout dans le cas de LDPC quantiques, voir chapitre 4).

On définit tout d'abord la distance minimale duale :

Définition 47 (Distance minimale duale d'un code)**.** *Soit $\mathscr{C}$ un code correcteur et son graphe de Tanner $\mathscr{G}(\mathscr{C}) = (V, C, E)$. Sa* dduale $\bar{d}$ *est la distance minimale du code $\bar{\mathscr{C}}$ de graphe de Tanner $\mathscr{G}\bar{\mathscr{C}} = (C, V, E)$.*

Théorème 5 (Distance minimale du code produit ([TZ09]))**.** *La distance minimale d du code produit $\mathscr{C}_q = \mathscr{C}_1 \boxtimes \mathscr{C}_2$ vérifie :*

$$d \geqslant \min\{d_1, d_2, \bar{d}_1, \bar{d}_2\} \tag{5.1}$$

Les bornes sont atteintes dans les cas suivants :

- *Si les dimensions de $\mathscr{C}_1, \mathscr{C}_2, \bar{\mathscr{C}}_1, \bar{\mathscr{C}}_2$ sont non-nulles :*

$$d = \min\{d_1, d_2, \bar{d}_1, \bar{d}_2\} \tag{5.2}$$

- *Si la dimension de $\mathscr{C}_1$ ou de $\mathscr{C}_2$ est nulle, et les dimensions duales de $\mathscr{C}_1$ et de $\mathscr{C}_2$ sont non-nulles :*

$$d \leqslant \min\{\bar{d}_1, \bar{d}_2\} \tag{5.3}$$

- *Si la dimension duale de de $\mathscr{C}_1$ ou de $\mathscr{C}_2$ est nulle, et les dimensions de $\mathscr{C}_1, \mathscr{C}_2$ sont non-nulles :*

$$d \leqslant \min\{d_1, d_2\} \tag{5.4}$$

où d_1 et d_2 sont les distances minimales de $\mathscr{C}_1$ et $\mathscr{C}_2$, et $\bar{d}_1$ et $\bar{d}_2$ les distances minimales duales de $\mathscr{C}_1$ et $\mathscr{C}_2$.

On peut noter que, dans le cas où $\mathscr{C}_1$ a pour dimension 0 et $\mathscr{C}_2$ dimension duale 0, le théorème ne donne pas de borne supérieure sur la distance minimale. Cela dit, dans ce cas précis, la dimension de $\mathscr{C}_q$ est nulle ($k_1 k_2 + \bar{k}_1 \bar{k}_2 = 0$).

Même s'il est intéressant de connaître les différents cas, dans la pratique on utilisera surtout l'équation 5.1. Comme il facile de construire des codes classiques ayant une distance minimale linéaire en leur longueur ($O(n_1)$ et respectivement $O(n_2)$), en choisissant $n_1 \sim n_2$ on obtient une distance minimale du code quantique en $O(n_1) = O(\sqrt{n})$.

Démonstration. Montrons tout d'abord l'équation 5.1. Le code $\mathscr{C}_q$ étant CSS, on cherche des erreurs composées uniquement de X ou de Z (voir la preuve en section 3.4).

Soit E_Z une erreur composée uniquement de I et de Z, non triviale et non détectée telle que $\mathrm{w}(E_Z) = w < d_1$ et $w < \bar{d}_2$. Pour cette erreur, les nœuds de parité en X sont les seuls nœuds de parité significatifs (ils détectent éventuellement quelque chose), et ceux en Z sont « invisibles ». On cherche à montrer que E_Z appartient au stabilisateur, c'est-à-dire engendré par les stabilisateurs associés aux nœuds de parité en Z.

Soient :

- $V_1' = \{v_1 \in V_1,\ \exists v_2 \in V_2,\ E_Z(v_1, v_2) \neq I\} \subset V_1$
- $C_2' = \{c_2 \in C_2,\ \exists c_1 \in C_1,\ E_Z(c_1, c_2) \neq I\} \subset C_2$

les ensembles de nœuds de variable et de parité restreints aux coordonnées « utiles ». Soient $\mathscr{C}_1'$ et $\mathscr{C}_2'$ les codes classiques restreints associés :

- $\mathscr{C}_1' = (V_1', C_1, \{v_1 \rightarrowtail c_1, v_1 \in V_1', v_1 \rightarrowtail c_1 \in E_1\})$
- $\mathscr{C}_2' = (V_2, C_2', \{v_2 \rightarrowtail c_2, c_2 \in C_2', v_2 \rightarrowtail c_2 \in E_2\})$

On considère le code produit $\mathscr{C}_q' = \mathscr{C}_1' \boxtimes \mathscr{C}_2'$. Ce code a pour dimension :

$$k' = k_1' k_2' + \bar{k_1'} \bar{k_2'}$$

On considère également E_Z' l'erreur E_Z réduite aux nœuds variable de $\mathscr{C}_q'$. Par définition des ensembles V_1' et C_2', E_Z et E_Z' ont le même support.

Puisque $V_1' \subset V_1$ et que $\mathscr{C}_1$ et $\mathscr{C}_1'$ ont le même ensemble de nœuds de parité, $\mathscr{C}_1' \subset \mathscr{C}_1$. La distance minimale de $\mathscr{C}_1'$ est donc au moins égale à d_1. Or, $\#V_1' < d_1$. Autrement dit, $\mathscr{C}_1'$ a moins de nœuds variable que sa distance minimale : ce n'est possible que si $k_1' = 0$.

Par le même raisonnement sur $\bar{\mathscr{C}_2'}$, on obtient $\bar{k_2'} = 0$, et donc $k' = 0$.

Le code $\mathscr{C}_q'$ possède les mêmes nœuds de parité en X que le code d'origine $(C_1 \times V_2)$, E_Z' est donc indétectée dans $\mathscr{C}_q'$. Ce code ayant une dimension nulle, E_Z' appartient donc au stabilisateur de $\mathscr{C}_q'$, ou plus précisément, au stabilisateur en Z de $\mathscr{C}_q'$. Or ce stabilisateur est engendré par les stabilisateurs associés aux nœuds de parité en Z : $\mathcal{S}_Z' = \langle \mathcal{S}(V_1' \times C_2') \rangle \subset \langle \mathcal{S}(V_1 \times C_2) \rangle = S_Z$. Une combinaison de stabilisateurs qui donne E_Z' dans $\mathscr{C}_q'$ donne E_Z dans $\mathscr{C}_q$ si on peut prouver que les $\mathcal{S}(V_1' \times C_2')$ n'ont pas de support sur des coordonnées « enlevées » de $\mathscr{C}_q$. Or, si $(v_1', c_2') \in V_1' \times C_2'$, les nœuds auxquels il est relié dans $\mathscr{C}_q$ sont :

- de la forme $(c_1, c_2') \in C_1 \times C_2'$, ou
- de la forme $(v_1', v_2) \in V_1' \times V_2$

Dans les deux cas, ce sont bien des coordonnées de nœuds de variable de $\mathscr{C}_q'$.

E_Z appartient donc au stabilisateur de $\mathscr{C}_q$, ce qui signifie que $d_Z \geqslant \min\{d_1, \bar{d_2}\}$.

Par un raisonnement symétrique, on obtient également $d_X \geqslant \min\{d_2, \bar{d_1}\}$, et l'équation (5.1).

Pour montrer que cette borne est atteinte, il faut exhiber une erreur du « bon » poids qui soit non détectée, mais qui n'appartienne pas au stabilisateur. Supposons que $\dim(\mathscr{C}_1)$ et $\dim(\mathscr{C}_2)$ soient non nuls. On va montrer que $d \geqslant d_1$.

Soit w_1 un mot de code (binaire) de $\mathscr{C}_1$, de poids d_1. Pour n'importe quel v_2^0 fixé, on peut définir l'erreur E_Z suivante, de poids d_1 :

- $E_Z(v_1, v_2) = Z$ si et seulement si $v_2 = v_2^0$ et si $w_1(v_1) = 1$,
- $E_Z(v_1, v_2) = I$ sur toutes les autres positions de type $V_1 \times V_2$,
- $E_Z(c_1, c_2) = I$ sur toutes les positions de type $C_1 \times C_2$.

Soit w_2 un mot de code quelconque non trivial de $\mathscr{C}_2$. On peut effectuer la con-

struction symétrique : pour n'importe quel v_1^0 fixé, on définit E_X :

- $E_Z(v_1, v_2) = X$ si et seulement si $v_1 = v_1^0$ et si $w_2(v_2) = 1$,
- $E_Z(v_1, v_2) = I$ sur toutes les autres positions de type $V_1 \times V_2$,
- $E_Z(c_1, c_2) = I$ sur toutes les positions de type $C_1 \times C_2$.

Ces deux erreurs ont pour support une colonne et une ligne de la matrice indexée par $V_1 \times V_2$, comme on le voit sur la figure 5.3.

Par construction, E_Z est indétectée : soit un nœud de parité en X (c_1, v_2). Si $v_2 \neq v_2^0$, alors le nœud de parité n'est relié à aucune position de E_Z qui ne soit pas I. Si $v_2 = v_2^0$, il est relié à $V(c_1) = \{(v_1, v_2^0), v_1 \rightarrowtail c_1 \in E_1\}$. Autrement dit on a exactement une copie du nœud de parité $c_1 \in C_1$. Or, par construction, on a affecté une « copie en Z » d'un mot de code de C_1 sur les positions (v_1, v_2^0) : le nombre de positions à Z dans $V(c1)$ est donc nécessairement pair. De la même façon, E_X est indétectée.

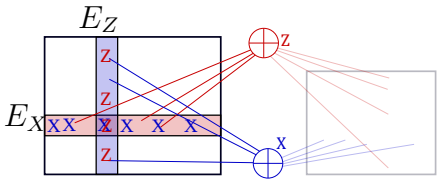


FIGURE 5.3 – Représentation de E_X et E_Z sur la même matrice $V_1 \times V_2$, selon la même convention que la figure 5.2. Ces erreurs n'ont pas de support sur la partie $C_1 \times C_2$, cette section est donc grisée. Changer v_2^0 correspond à faire « glisser » horizontalement la colonne E_Z, et changer v_1^0 correspond à faire « glisser » verticalement la ligne E_X : on peut choisir ces deux constantes de façon à ce que les supports s'intersectent.

On souhaite que ces deux erreurs anti-commutent : pour cela, on choisit $v_2^0 = \min\{i, w_2(i) = 1\}$ et $v_1^0 = \min\{i, w_1(i) = 1\}$. Ainsi, E_X et E_Z ont un et un seul emplacement commun différent de I : (v_1^0, v_2^0), où ils anti-commutent. Cela signifie donc (voir section 3.3.2.3) qu'ils n'appartiennent pas au stabilisateur, et donc qu'E_Z est une erreur atteignant la distance minimale d_1.

Par symétrie on a de la même façon $d \geqslant d_2$, ce qui donne la borne supérieure 5.3. De même, quand les dimensions de $\bar{\mathscr{C}_1}$ et $\bar{\mathscr{C}_2}$ sont non nulles, on a la borne (5.4). Et enfin lorsque ces quatre dimensions sont non nulles, en combinant (5.3) et (5.4) on obtient (5.2).

■

On peut remarquer que cette preuve fournit assez naturellement des candidats pour la base symplectique des erreurs sérieuses (voir section 3.3.2.3) : on peut écrire une base systématique des mots de codes de $\mathscr{C}_1$ et $\mathscr{C}_2$: $x^1, x^2, \ldots x^{k_1}$ et $z^1, \ldots z^{k_2}$

tels que $x_i^j = 0$ si $i \leqslant k_1$, sauf en position j où x_j^j vaut 1 (et de même pour les mots z avec k_2.

Ensuite on construit $k_1 k_2$ couples $\bar{X}_{i,j}, \bar{Z}_{i,j}$, ayant pour support $V_1 \times V_2$ pour $1 \leqslant i \leqslant k_1, 1 \leqslant j \leqslant k_2$, basés sur les E_X et E_Z de la preuve :

- $\bar{Z}_{i,j}(i', j') = Z$ si $x_{i'}^i = 1$ et $j' = j$, et $\bar{Z}_{i,j}(i', j') = I$ ailleurs,
- $\bar{X}_{i,j}(i', j') = X$ si $i' = i$ et $z_{j'}^j = 1$, et $\bar{Z}_{i,j}(i', j') = I$ ailleurs.

La construction est schématisée dans la figure 5.4. De par cette construction, pour $i' \leqslant k_1, j' \leqslant k_2$, $\bar{Z}_{i,j}(i', j')$ vaut I partout sauf en i, j. C'est le cas également pour $\bar{X}_{i,j}$, d'où on peut en conclure

$$\bar{X}_{i,j} \star \bar{Z}_{i',j'} = \delta_{i,i'} \delta_{j,j'}$$

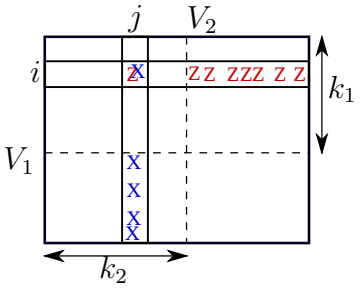


FIGURE 5.4 – Représentation matricielle de $\bar{X}_{i,j}$ et de $\bar{Z}_{i,j}$, pour $1 \leqslant i \leqslant k_1, 1 \leqslant j \leqslant k_2$

On peut bien entendu faire une construction symétrique avec $C_1 \times C_2$ dans le cas où $\bar{k}_1 \bar{k}_2 \neq 0$.

5.2 Application : construction de codes LDPC

5.2.1 Généralités

La technique pour produire un « bon » code produit est relativement simple : il suffit de choisir des « bons » codes classiques pour leur servir de composants.

Si on choisit pour $\mathscr{C}_1$ et $\mathscr{C}_2$ des codes LDPC, le produit obtenu est lui aussi LDPC : pour le graphe de Tanner, le nombre d'entrées non I par ligne et colonne correspond au degré des nœuds variable et de parité.

Or, soit $\mathscr{C}_1$ et $\mathscr{C}_2$ de degré des nœuds variable et de parité borné : $\deg(v_i) \leqslant d_{iV} \forall v_i \in V_i$, $\deg(c_i) \leqslant d_{iC} \forall c_i \in C_i$. Le code obtenu possède alors $n_1 n_2$ nœuds variable de degré au maximum $d_{1V} + d_{2V}$ et $r_1 r_2$ nœuds variable de degré au plus $d_{1C} + d_{2C}$. Du côté des nœuds de parité, il y a $r_1 n_2$ nœuds de degré au plus $d_{1C} + d_{2V}$

et $n_1 r_2$ nœuds de degré au plus $d_{1V} + d_{2C}$. Ces degrés sont résumés sur la figure 5.5, et découlent simplement de la structure du produit.

En effet, soit $(v_1, v_2) \in V_1 \times V_2$. Les arêtes possibles reliées à ce sommet sont d'une part celles de la forme $(v_1, v_2) \rightarrowtail (c_1, v_2)$, pour $v_1 \rightarrowtail c_1 \in E_1$ et celles de la forme $(v_1, v_2) \rightarrowtail (v_1, c_2)$, pour $v_2 \rightarrowtail c_2 \in E_2$. Il y en a donc, au total, autant qu'il y a d'arêtes $v_1 \rightarrowtail c_1 \in E_1$ et d'arêtes $v_2 \rightarrowtail c_2 \in E_2$: cela fait $\deg(v_1) + \deg(v_2)$. Le raisonnement s'adapte facilement aux autres types de nœuds.

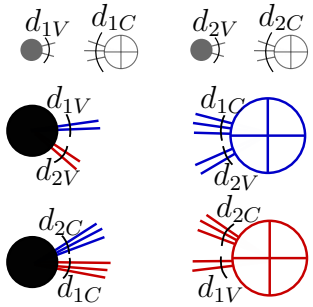


FIGURE 5.5 – Degrés des nœuds du code produit lorsque les codes composants ont des degrés bornés (ou fixes).

5.2.2 Le cas du code torique

Si on choisit pour composants $\mathscr{C}_1$ et $\mathscr{C}_2$ le (même) code à répétition de taille n, on obtient le code torique, présenté dans la section 4.1.3.

Propriété 25 (Code torique de Kitaev vu comme code produit)**.** *Soit $\mathscr{C}_R$ le code à répétition de longueur n :*

$$\begin{aligned} V_R &= \{0, 2, \ldots, 2n-2\} \\ C_R &= \{1, 3, \ldots 2n-1\} \\ E_R &= \{(2i \mod 2n, 2i \pm 1 \mod 2n), 0 \leqslant i < n\} \end{aligned}$$

Le code torique de Kitaev *de paramètre n est le code produit de $\mathscr{C}_R$ avec lui-même : $\mathscr{C} = \mathscr{C}_R \boxtimes \mathscr{C}_R$.*

Démonstration. On applique la définition 45 :

$$\begin{aligned} V &= V_R \times V_R \cup C_R \times C_R \\ &= \{(i,j), i \text{ et } j \text{ pairs}\} \cup \{(i,j), i \text{ et } j \text{ sont impairs}\} \\ C_X &= C_R \times V_R \\ &= \{(i,j), i \text{ impair } j \text{ pair}\} \\ C_Z &= V_R \times C_R \\ &= \{(i,j), i \text{ pair } j \text{ impair}\} \end{aligned}$$

$$\begin{aligned} E_X &= \{(v_1,v_2)\rightarrowtail(c_1,v_2), v_1 \rightarrowtail c_1 \in E_1, v_2 \in V_2\} \\ &\quad \cup \{(c_1,c_2)\rightarrowtail(c_1,v_2), v_2 \rightarrowtail c_2 \in E_2, c_1 \in C_1\} \\ &= \{(i,j)\rightarrowtail(i \pm 1,j), i \text{ pair}, j \text{ pair}\} \\ &\quad \cup \{(i,j)\rightarrowtail(i,j \pm 1), i \text{ impair}, j \text{ impair}\} \\ E_Z &= \{(v_1,v_2)\rightarrowtail(v_1,c_2), v_2 \rightarrowtail c_2 \in E_2, v_1 \in V_1\} \\ &\quad \cup \{(c_1,c_2)\rightarrowtail(v_1,c_2), v_1 \rightarrowtail c_1 \in E_1, c_2 \in C_2\} \\ &= \{(i,j)\rightarrowtail(i,j \pm 1), i \text{ pair}, j \text{ pair}\} \\ &\quad \cup \{(i,j)\rightarrowtail(i \pm 1,j), i \text{ impair}, j \text{ impair}\} \end{aligned}$$

Ce qui correspond bien à la définition 42.

■

Avec cette définition, on peut déduire aisément toutes les caractéristiques du code torique :

Propriété 26. *Le code torique de paramètre n a pour longueur $2n^2$, pour dimension 2 et pour distance minimale n.*

Démonstration. Le code à répétition a pour longueur n, dimension 1, distance minimale duale 1, et distance minimale n. On en déduit ces grandeurs pour le code torique grâce à la définition 45, et aux théorèmes 4 et 5.

■

5.3 Décodage

5.3.1 Présentation

Ces codes ont été testés avec un décodage de type BP CSS simple (voir section 4.2.4). On choisit des codes LDPC classiques de même structure, avec n_1

nœuds variable et r_1 nœuds de parité, et de dimension $k_1 = n_1 - r_1$: ces codes étant générés aléatoirement, ils sont de dimension maximale avec une forte probabilité. On les choisit avec des degrés fixés d_v et d_c, on a donc : $R_1 = 1 - \frac{r_1}{n_1} = \frac{d_c - d_v}{d_c}$. On cherche à obtenir un rendement $\frac{1}{4}$, on veut donc :

$$\begin{aligned} R = \frac{1}{4} &= \frac{R_1^2}{2R_1^2 - 2R_1 + 2} \\ 3R_1^2 + 2R_1 + 2 &= 0 \end{aligned}$$

Ce qui donne $R_1 \simeq 0,55$. On choisit ici $d_v = 4$ et $d_c = 9$, qui donne une bonne approximation du rendement voulu $(0,56)$ et permet d'avoir une structure simple et des degrés peu élevés (donc moins d'arêtes, donc un décodage plus rapide).

On sait que ces codes ont une distance minimale non dégénérée constante, qu'on peut facilement majorer par le poids d'un stabilisateur associé à un nœud de parité : ici $d_v + d_c = 13$ mais on pouvait espérer des performances acceptables.

5.3.2 Performances

Sur le graphique de la figure 5.6, on met en valeur les courbes de décodage des codes suivants (de rendement 1/4 environ) :

Code	Longueur du code composant	d_V	d_C	Longueur finale
Produit A	900	4	9	970 000
Produit B	400	4	9	242 500

On peut constater que, même avec des codes de grande longueur (la longueur étant proportionnelle au carré de celle des codes composants, il est logique d'avoir des codes longs), les performances sont très décevantes.

5.3.3 Explication des performances

Si on considère les erreurs en Z seulement (et donc les nœuds de parité en X), le code est structuré en « couches » (voir figure 5.2). C'est ce qui donne, grossièrement, la distance minimale : il *faut* un mot de code sur une couche complète pour avoir une erreur non triviale. D'un autre côté, pour le décodage, cela veut dire qu'il *suffit* d'un échec sur une de ces couches pour que tout échoue.

Plus précisément, supposons que tout soit bien décodé sauf sur une colonne de $V_1 \times V_2$, autrement dit on a une erreur E telle qu'il existe un indice v_2^0 tel que :

– $\forall c_1, c_2 \in C_1 \times,\ E(c_1, c_2) = I$

– $\forall v_1 \in V_1, v_2 \in v_2 \neq v_2^0,\ E(v_1, v_2) = I$

Si on regarde uniquement les nœuds variable et de parité concernés par le support de l'erreur, il reste $V_1 \times \{v_2^0\}$ d'une part et $C_1 \times \{v_2^0\}$ d'autres part, et

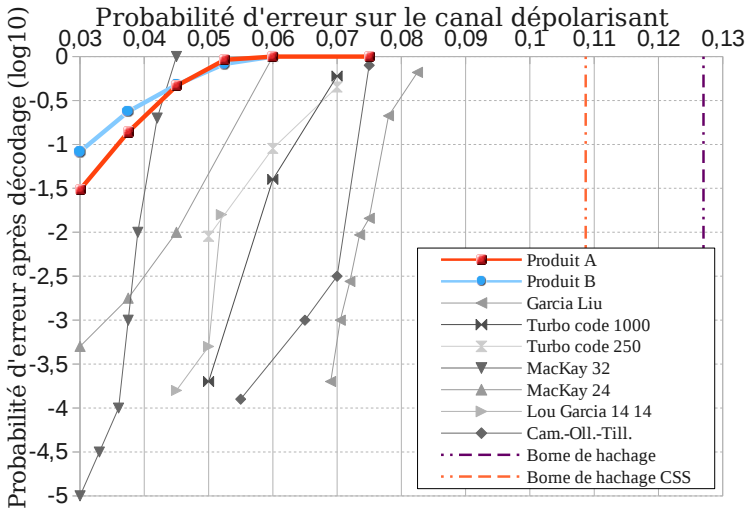


FIGURE 5.6 – Courbes de décodage du code produit comparées à d'autres codes quantiques de même rendement.

les nœuds étant reliés selon les arêtes de $\mathscr{C}_1$, on a exactement une copie de $\mathscr{C}_1$ (comme dans la seconde partie de la preuve du théorème 5). Il est donc possible de décoder, dans ce cas, si et seulement s'il est possible de décoder ce cas d'erreur pour $\mathscr{C}_1$. Connaissant les performances du code $\mathscr{C}_1$, on peut donc donner une borne supérieure sur les performances du code produit :

Propriété 27 (Capacité de correction du code produit)**.** *Soient $\mathscr{C}_1$ et $\mathscr{C}_2$ deux codes classiques. On note $P_{\mathscr{C}_1}(p)$ la probabilité d'erreur résiduelle après un décodage au maximum de vraisemblance du code classique $\mathscr{C}_1$ avec une probabilité d'erreur sur le canal binaire symétrique p. On a alors :*

$$P_{\left[\mathscr{C}_1 \boxtimes \mathscr{C}_2\right]_X}(p) \geqslant 1 - (1 - P_{\mathscr{C}_1}(p))^{n_2} \tag{5.5}$$

On a des résultats symétriques pour $\mathscr{C}_2, \bar{\mathscr{C}_1}, \bar{\mathscr{C}_2}$.

Démonstration. La preuve est en deux parties. Tout d'abord, on montre que si on suppose qu'on n'a aucune erreur sur un certain sous-ensemble des nœuds variable, alors le décodage ne peut être que meilleur : ce résultat, qui semble assez naturel, n'est pas forcément vrai pour tout décodage ; en l'occurence il l'est pour le décodage au maximum de vraisemblance.

Ensuite, grâce à ce résultat, on peut supposer qu'on n'a que des erreurs sur la partie $V_1 \times V_2$: la probabilité d'erreur résiduelle sera alors plus petite que $\mathrm{P}_{\left[\mathscr{C}_1 \boxtimes \mathscr{C}_2\right]_X}(p)$.

On peut donc ôter toute la partie $C_1 \times C_2$ du graphe de Tanner, et puisqu'on considère la section en X, il ne reste qu'un seul type d'arêtes : celles qui relient $V_1 \times V_2$ à $C_1 \times V_2$. Or cette section est composée d'exactement $\#V_2 = n_2$ copies du graphe de $\mathscr{C}_1$: en effet, tous les nœuds de type (v_1, v_2) sont reliés aux nœuds (c_1, v_2) si $v_1 \rightarrowtail c_1$ dans le graphe de $\mathscr{C}_1$, mais ils ne peuvent pas être liés à un (c_1', v_2') si $v_2 \neq v_2'$. De même pour les nœuds de parité.

Ainsi la probabilité de décoder correctement cette section est majorée par la probabilité de décoder *chaque* copie du code $\mathscr{C}_1$, donc la probabilité d'échouer est minorée par celle d'échouer sur au moins une copie du code, d'où (5.5).

Il reste à voir la première partie. Soient X, Y et Z les variable aléatoire correspondant à des mots de longueur n sur la chaîne de canaux suivants :

$$X \longrightarrow Y \longrightarrow Z$$

X est la variable aléatoire correspondant au mot original envoyé, Y correspond au mot reçu après le premier canal appelé $\mathbf{C}_1$ et Z correspond au mot reçu après avoir envoyé Y sur le second canal, $\mathbf{C}_2$. $\mathbf{C}$ est la composée des deux canaux. Les variables aléatoires correspondantes forment ce qu'on appelle une chaîne de Markov :

Définition 48 (Chaîne de Markov)**.** *Les variables aléatoires X, Y et Z forment une* chaîne de Markov *(on note $X \to Y \to Z$) si on peut écrire*

$$p(z \mid y, x) = p(z \mid y) \tag{5.6}$$

Autrement dit, la variable aléatoire Z ne dépend que de Y et non de X.

Propriété 28 (Réversibilité des chaînes de Markov)**.** *Si $X \to Y \to Z$ forment une chaîne de Markov, alors $Z \to Y \to X$ également.*

Démonstration. Si on a $p(z \mid y, x) = p(z \mid y)$, alors on peut écrire :

$$p(x \mid y, z) = \frac{p(x, y, z)}{p(y, z)} = \frac{p(z \mid x, y)p(x, y)}{p(z \mid y)p(y)} = \frac{p(z \mid y)p(x, y)}{p(z \mid y)p(y)} = \frac{p(x, y)}{p(y)} = p(x \mid y)$$

■

On a alors le lemme suivant, qui énonce ce qu'on attend intuitivement :

Lemme 2.

$$P_{\mathscr{C}}(p) \geqslant P'_{\mathscr{C}}(p)$$

où $P_{\mathscr{C}}(p)$ est la probabilité d'erreur résiduelle après décodage de $\mathscr{C}$ au maximum de vraisemblance sur le canal $\mathbf{C}$, et $P'_{\mathscr{C}}(p)$ celle correspondant au canal $\mathbf{C}_1$ (uniquement le premier canal dans la chaîne de Markov).

On applique ensuite ce lemme avec $\mathbf{C}_1$ qui correspond à un canal binaire symétrique sur les bits correspondants aux nœuds $V_1 \times V_2$, et laisse inchangé ceux correspondant aux nœuds $C_1 \times C_2$, et $\mathbf{C}_2$ qui fait l'inverse, c'est-à-dire qui laisse inchangés les bits correspondants aux nœuds de $V_1 \times V_2$ et applique un canal binaire symétrique à ceux correspondant à $C_1 \times C_2$. Ainsi, le canal complet $\mathbf{C}$ est un canal binaire symétrique (sur tous les bits), et le lemme 2 énonce alors que la probabilité d'erreur résultante après décodage est minorée par celle où il n'y a que les bits de $V_1 \times V_2$ qui sont touchés.

■

Preuve du lemme 2. On définit le décodage au maximum de vraisemblance de la façon suivante : on prend f une fonction dite de décodage, qui prend en argument un mot (à la sortie du canal) y, et renvoie un mot $\hat{x}$, qu'on espère être le mot x envoyé. La probabilité d'erreur p_e s'écrit donc :

$$p_e = \mathbb{P}[f(y) \neq x]$$

Pour le décodage au maximum de vraisemblance, on cherche donc à minimiser, sur toutes les fonctions f possibles, cette quantité, ce qui donne :

$$\mathrm{P}_{\mathscr{C}} = \inf_f \{\mathbb{P}[f(Y) \neq X]\}$$

Reprenons notre chaîne de Markov de canaux $X \to Y \to Z$.

La probabilité d'erreur résiduelle après le passage dans les deux canaux (donc après passage dans $\mathbf{C}$) s'exprime donc :

$$\mathrm{P}_{\mathscr{C}} = \inf_f \{\mathbb{P}[f(Z) \neq X]\}$$

Et si on décode après le premier canal seulement, la probabilité d'erreur (avec f' la fonction de décodage sur le premier canal) est :

$$\mathrm{P}'_{\mathscr{C}} = \inf_{f'} \{\mathbb{P}(f'(Y) \neq X)\}$$

Considérons à présent le cas où, pour retrouver x, on utilise une fonction g qui prend en argument y et z. On écrit alors la grandeur correspondante :

$$A = \inf_g \{\mathbb{P}(g(Z, Y) \neq X)\}$$

On constate qu'on a alors les deux inégalités suivantes :

$$\mathrm{P}_{\mathscr{C}} \geqslant A \tag{5.7}$$

$$A = \mathrm{P}'_{\mathscr{C}} \tag{5.8}$$

Pour l'équation (5.7), on peut constater que si on a une fonction f telle que $\hat{x} = f(z)$, on peut construire une fonction g telle que $\hat{x} = g(y, z) = f(z)$, autrement dit, on oublie y et on ne calcule $\hat{x}$ qu'avec z. On a alors $\mathbb{P}[f(Z) = X] = \mathbb{P}[g(Y, Z) = X]$ pour cette fonction g. Ainsi, des deux ensembles décrits dans $\mathrm{P}_{\mathscr{C}}$ et A, le premier est inclus dans le second, ce qui implique que l'infimum du premier est supérieur à celui du second.

Pour l'équation (5.8), grâce à la propriété 28, on a $p(x \mid y) = p(x \mid y, z)$. Cela signifie donc que

on a $\mathbb{P}_{Z,Y}[g(Z, Y) \neq X] = \mathbb{P}_{Z,Y}[g'(Y) \neq X]$, autrement dit retrouver X en sachant Y est Z est équivalent à retrouver X en sachant Y seulement puisque Z est plus bruité que Y. En effet, si $p(x, y, z) = p(x)p(y \mid x)p(z \mid y)$ (par (5.6)), on peut alors écrire également $p(x, y, z) = p(x, y)p(z \mid y)$. On a alors :

$$p(x \mid y) = \frac{p(x, y)}{p(y)} = \frac{p(x, y, z)}{p(z \mid y)} \frac{1}{p(y)} = \frac{p(x, y, z)}{p(y, z)} = p(x \mid y, z)$$

On a donc l'inégalité du lemme.

■

Cela signifie que le code produit hérite des propriétés « dégradées » de décodage de $\mathscr{C}_1$. Cela souligne l'importance d'avoir pour codes composants de bons codes, mais cela n'est pas toujours suffisant. On peut peut même majorer plus précisément $\mathrm{P}_{\left[\mathscr{C}_1 \boxtimes \mathscr{C}_2\right]_X}$.

En effet, on peut noter que lorsque n_1 et n_2 augmentent (le produit étant symétrique, il faut les garder de même ordre de grandeur), $\mathrm{P}_{\mathscr{C}_1}(p)$ tend vers 0 ou 1, autrement dit le décodage tend vers du « tout ou rien », dans lequel la puissance de n_2 devrait avoir moins d'impact : cependant n_2 augmente aussi. En augmentant n_2, on augmente la probabilité qu'une des erreurs tirées au hasard soit sensiblement plus élevée que la moyenne, même si en augmentant n_1, cette probabilité diminue.

En fait, cette probabilité est liée à la probabilité d'obtenir, sur un de ces n_2 vecteurs, une proportion d'erreurs « suffisamment » plus grande que la moyenne :

Propriété 29 (Capacité de correction du code produit - cas général)**.** *Soit $\mathscr{C} = \mathscr{C}_1 \boxtimes \mathscr{C}_2$, de longueur n. Pour p fixé, on a :*

$$P_{\left[\mathscr{C}_1 \boxtimes \mathscr{C}_2\right]_X}(p) \geqslant \alpha n^{1/4} e^{\sqrt{n}\beta(p)} \tag{5.9}$$

avec $\beta(p) < 0$.

Démonstration. On utilise essentiellement le lemme suivant :

Lemme 3. *Soient n un entier positif, et $0 < p \leqslant q < 1$, on supposera p et q proches de 0. On appelle « tirage d'un vecteur » une sucessions de n lancers indépendants de $0-1$, avec probabilité p d'obtenir 1. Si on effectue n tels tirages de vecteurs indépendants, alors lorsque $n \to \infty$, la probabilité d'obtenir au moins un tirage contenant une proportion q de 1 est :*

$$P \sim A(q)\sqrt{n}e^{nB(p,q)} \tag{5.10}$$

Où $A(q) = \frac{1}{\sqrt{2\pi q(1-q)}}$ et $B(p,q) = q\ln(\frac{p}{q}) + (1-q)\ln(\frac{1-p}{1-q}) < 0$ si $p < q$.

On applique le lemme avec $n = n_1 = n_2$, et on choisit q suffisamment grand, de telle sorte que le décodage de $\mathscr{C}_1$ échoue (on peut prendre par exemple q tel que $h(q) > 1 - R$, ou q raisonnablement proche de 1). La longueur totale du code est en $\sqrt{n}$, on obtient alors l'équation 5.9.

■

Preuve du lemme 3. Soit $q' = \frac{\lceil qn \rceil}{n}$. Lorsque $n \to \infty$, $q' \to q$, donc pour simplifier, on travaillera directement avec q', ce qui revient à supposer que qn est un entier. Si on effectue n tirages de $0-1$ indépendants, la probabilité d'obtenir exactement qn fois 1 est :

$$P^n_{qn} = \binom{n}{qn} p^{qn}(1-p)^{n-qn}$$

Donc la probabilité d'obtenir un nombre de 1 supérieur à qn est :

$$P^n_{\geqslant q} = \sum_{k=qn}^{\infty} P^n_{qn} = \sum_{k=qn}^{\infty} \binom{n}{k} p^k (1-p)^{n-k}$$

Dans une telle somme, pour p petit, le terme prépondérant est le terme avec la puissance de p la plus faible : c'est donc ici P^n_{qn}. On utilise la formule de Stirling pour obtenir un équivalent de $\binom{n}{qn}$:

$$\begin{aligned}
\binom{n}{qn} &= \frac{n!}{(qn)!(n-qn)!} \\
&\sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \left(\frac{e}{qn}\right)^{qn} \frac{1}{\sqrt{2\pi qn}} \left(\frac{e}{(1-q)n}\right)^{(1-q)n} \frac{1}{\sqrt{2\pi(1-q)n}} \\
&\sim \frac{n^n}{(qn)^{qn}((1-q)n)^{(1-q)n}} \frac{1}{\sqrt{2\pi q(1-q)n}} \\
&\sim \frac{1}{q^{qn}(1-q)^{(1-q)n}} \frac{A(q)}{\sqrt{n}}
\end{aligned}$$

Avec $A(q)$ tel que défini plus haut. On a donc :

$$\begin{aligned} P_{qn}^n &\sim \frac{A(q)}{\sqrt{n}} \left(\frac{p}{q}\right)^{qn} \left(\frac{1-p}{1-q}\right)^{(1-q)n} \\ &\sim \frac{A(q)}{\sqrt{n}}\, e^{n \ln\left(\left(\frac{p}{q}\right)^q \left(\frac{1-p}{1-q}\right)^{1-q}\right)} \\ &\sim \frac{A(q)}{\sqrt{n}}\, e^{nB(p,q)} \end{aligned}$$

Il reste encore à montrer que $B(p,q) < 0$. Pour cela, on fixe q et on dérive l'expression en p, pour $0 < p \leqslant q$:

$$\frac{dB(p,q)}{dp} = \frac{q}{p} - \frac{1-q}{1-p}$$

Or $\frac{q}{p} > 1$, et $\frac{1-q}{1-p} < 1$, donc l'expression ci-dessus est strictement positive. $B(p,q)$ est donc croissante en p, et vaut 0 pour $p = q$: $B(p,q) < 0$ pour $p < q$.

De là, on déduit la probabilité d'avoir au moins un tel cas sur n tirages de vecteurs indépendants :

$$P = 1 - (1 - P_{qn}^n)^n \sim nP_{qn}^n \sim A(q)\sqrt{n}e^{nB(p,q)}$$

■

On a donc une probabilité d'erreur qui décroît exponentiellement avec n, cependant, le facteur $\alpha n^{1/4}$ est assez élevé et il faut donc augmenter significativement n pour espérer obtenir des performances acceptables (au delà de 100 000), et à ce moment commencent à apparaître les problèmes de distance minimale non dégénérée constante.

Au vu de ces résultats, on peut tout de même se poser la question suivante : et si on tenait compte de la dégénérescence ? En effet, les résultats présentés ci-dessus n'en tiennent absolument pas compte. Cependant, les erreurs qui font échouer le décodage sont des erreurs de type « erreur de $\mathscr{C}_1 \times v_2$ », autrement dit des erreurs qui viennent du code composant $\mathscr{C}_1$ et non de la construction produit : comme les erreurs sérieuses sont de la forme « mot de code de $\mathscr{C}_1 \times v_2$ », il est assez probable que ces erreurs soient problématiques même si on prend en compte la dégénérescence.

Cependant, si on exécutait un décodage qui ne soit pas gêné par les erreurs bénignes, on pourrait alors augmenter suffisamment n pour que le facteur en $\exp(-\beta(p)\sqrt{n})$ devienne prépondérant. Ainsi, il faut travailler sur le décodage et la dégénérescence pour exploiter correctement ce type de codes : cela rappelle fortement (et à juste titre, puisqu'il s'agit d'un cas particulier) le code torique, si performant avec son algorithme spécifique mais très mauvais avec le BP.

5.4 Variante : le code produit tridimensionnel

Du fait de leur structure en « couches » quasiment indépendantes, les codes produits ne sont pas très efficaces. Cependant, ils ont un intérêt théorique : il existe des codes de rendement arbitraire avec une distance minimale en $\sqrt{n}$. Trouver des variantes peut donc être une bonne piste : soit d'un point de vue théorique pour tenter de trouver une distance minimale plus élevée (c'est le cas du code produit tridimensionnel, dans la section suivante), soit d'améliorer les performances pratiques sans perdre cette distance minimale (code produit q-aire, code produit spatialement couplé, dans les sections 6.3, et 7.2).

Ces codes n'ont malheureusement pas apporté les réponses désirées, même si leur étude apporte quelques éléments intéressants.

5.4.1 Définition

On commence par donner une variante du produit de codes, non CSS, représentée sur la figure 5.7 : la structure est exactement la même, sauf que les couleurs (i.e. le type des arêtes, X ou Z) changent.

Définition 49 (Code produit non CSS)**.** *Soient $\mathscr{C}_1$, $\mathscr{C}_2$ deux codes classiques de graphes de Tanner $\mathscr{G}(\mathscr{C}) = (V_1, C_1, E_1)$ et $\mathscr{G}(\mathscr{C}_2) = (V_2, C_2, E_2)$. On définit le code produit non-CSS $\mathscr{C}_q = \mathscr{C}_1 \boxtimes_2 \mathscr{C}_2$ de graphe de Tanner $\mathscr{G}(\mathscr{C}_q) = (V, C, E_X, E_Y, E_Z)$ de la façon suivante :*

- $V = V_1 \times V_2 \cup C_1 \times C_2$
- $C = C_1 \times V_2 \cup V_1 \times C_2$
- $E_X = \{(v_1, v_2) \rightarrowtail (c_1, v_2) | v_1 \rightarrowtail c_1 \in E_1, v_2 \in V_2, \}$
  $\cup\ \{(c_1, c_2) \rightarrowtail (v_1, c_2) | v_1 \rightarrowtail c_1 \in E_1, c_2 \in C_2\}$
- $E_Z = \{(v_1, v_2) \rightarrowtail (v_1, c_2) | v_2 \rightarrowtail c_2 \in E_2, v_1 \in V_1\}$
  $\cup\ \{(c_1, c_2) \rightarrowtail (c_1, v_2) | v_2 \rightarrowtail c_2 \in E_2, c_1 \in C_1\}$
- $E_Y = \emptyset$

On peut voir cette construction de la façon suivante : lorsqu'un nœud variable et un nœud de parité sont reliés, soit la première coordonnée correspond à une arête de $\mathscr{G}(\mathscr{C}_1)$ et la seconde ne bouge pas, auquel cas c'est une arête en X ; soit la première coordonnée est fixe et la seconde correspond à une arête de $\mathscr{G}(\mathscr{C}_2)$, auquel cas c'est une arête en Z.

Cette nouvelle construction, bien que non CSS, possède les mêmes propriétés que l'autre, en terme de distance minimale et dimension, en effet, elles sont identiques à une isométrie près.

Propriété 30 (Isométrie entre le produit CSS et le produit non-CSS)**.** *Soient $\mathscr{C}_1, \mathscr{C}_2$ deux codes classiques, et $\mathscr{C}_q = \mathscr{C}_1 \boxtimes \mathscr{C}_2$, $\mathscr{C}'_q = \mathscr{C}_1 \boxtimes_2 \mathscr{C}_2$, de longueurs n. Alors*

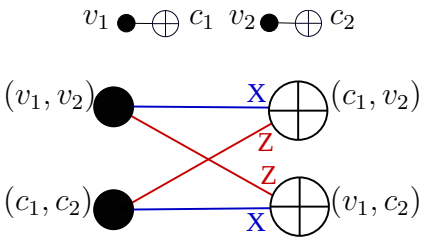


FIGURE 5.7 – Schéma de la construction du produit en version non-CSS : on peut voir que lorsqu'on change la première coordonnée, on emprunte une arête en X, et lorsqu'on change la seconde, on emprunte une arête en Z.

l'opérateur $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}}$ *défini ci-dessous est une isomorphie de* $\mathcal{E}^n$ *dans* $\mathcal{E}^n$*, qui envoie les erreurs sérieuses de* $\mathscr{C}_q$ *sur celles de* $\mathscr{C}'_q$*.*

$$\forall v \in V, \mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(E)(v) = \begin{cases} E(v) & si\ v \leqslant n_1 n_2 \\ E(v) & si\ v > n_1 n_2\ et\ E(v) = I\ ou\ Y \\ X & si\ v > n_1 n_2\ et\ E(v) = Z \\ Z & si\ v > n_1 n_2\ et\ E(v) = X \end{cases} \tag{5.11}$$

Grossièrement, l'opérateur $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}}$ agit sur les positions de type $C_1 \times C_2$, et change les X en Z et inversement.

Démonstration. L'opérateur $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}}$ est un morphisme de $\mathcal{E}_n \to \mathcal{E}_n$ qui préserve le poids : les positions v où $E(v) = I$ vérifient $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(E)(v) = I$, et celles où $E(v) \neq I$ vérifient $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(E)(v) \neq I$. Il est bijectif car $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}} \circ \mathbf{c}_{\mathrm{x}}^{\mathrm{z}} = I_{\mathcal{E}_n}$

Il faut encore vérifier qu'il conserve le produit $\star$: il suffit de vérifier que, sur une coordonnée v, $E_1(v) \star E_2(v) = \mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(E_1)(v) \star \mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(E_2)(v)$:

- si $v \leqslant n_1 n_2$, alors c'est évident puisque $E(v) = \mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(E)(v)$ sur ces positions,
- sinon, on a en position v :
  - un produit de la forme $I \star T$ pour T quelconque est tranformé en un produit de la forme $I \star T'$,
  - un produit de la forme $T \star T$ pour T quelconque est transformé en $T' \star T'$.
  - un produit de la forme $Y \star X$ est transformé en $Y \star Z$,
  - un produit de la forme $X \star Z$ est transformé en $Z \star X$.

Dans chacun des cas, le produit $\star$ n'est pas changé par la transformation.

Enfin, il faut vérifier que chaque élément de S, le stabilisateur de $\mathscr{C}_q$ est envoyé sur un élément de S', le stabilisateur de $\mathscr{C}'_q$: il suffit de le vérifier sur les $\mathcal{S}(c)$, $c \in C_X$ ou C_Z. Soit $c \in C_X$ un tel nœud de parité (avec les conventions de la définition 49), on a $c = (c_1, v_2) \in C_1 \times V_2$. Le support de $\mathcal{S}(c)$ est alors composé de :

– X sur les positions (v_1, v_2) où $v_1 \bullet\!\!\rightarrow c_1 \in E_1$,
– X sur les positions (c_1, c_2) où $v_2 \bullet\!\!\rightarrow c_2 \in E_2$.

Le support de $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(\mathcal{S}(c))$ est donc composé de :

– X sur les positions (v_1, v_2) où $v_1 \bullet\!\!\rightarrow c_1 \in E_1$,
– Z sur les positions (c_1, c_2) où $v_2 \bullet\!\!\rightarrow c_2 \in E_2$.

Or il s'agit exactement du support de $\mathcal{S}(c')$, c' le nœud de parité équivalent à c dans $\mathscr{C}'_q$. On a donc $\mathbf{c}_{\mathrm{x}}^{\mathrm{z}}(\mathcal{S}(c)) = \mathcal{S}(c') \in S'$.

■

Si cette construction n'apporte rien d'intéressant par rapport à sa version CSS, elle permet cependant d'introduire plus facilement sa variante tridimensionnelle : on peut ajouter une troisième coordonnée, et ajouter des arêtes en Y lorsqu'on se déplace selon une arête d'un troisième graphe :

Définition 50 (Code produit tridimensionnel)**.** *Soient $\mathscr{C}_1$, $\mathscr{C}_2$, $\mathscr{C}_3$ trois codes classiques de graphes de Tanner $\mathscr{G}(\mathscr{C}_1) = (V_1, C_1, E_1)$, $\mathscr{G}(\mathscr{C}_2) = (V_2, C_2, E_2)$ et $\mathscr{G}(\mathscr{C}_3) = (V_3, C_3, E_3)$ appelés* codes composants. *On définit le code produit non-CSS $\mathscr{C}_q = \mathscr{C}_1 \boxtimes \mathscr{C}_2 \boxtimes \mathscr{C}_3$ de graphe de Tanner $\mathscr{G}(\mathscr{C}_q) = (V, C, E_X, E_Y, E_Z)$ de la façon suivante :*

– $V = V_1 \times V_2 \times V_3 \cup C_1 \times C_2 \times V_3 \cup C_1 \times V_2 \times C_3 \cup V_1 \times C_2 \times C_3$
– $C = C_1 \times V_2 \times V_3 \cup V_1 \times C_2 \times V_3 \cup V_1 \times V_3 \times C_3 \cup C_1 \times C_2 \times C_3$
– $E_X = \{(v_1, v_2, v_3) \bullet\!\!\rightarrow (\mathbf{c}_1, v_2, v_3) | v_1 \bullet\!\!\rightarrow \mathbf{c}_1 \in E_1, v_2 \in V_2, v_3 \in V_3\}$
  $\cup \{(\mathbf{c}_1, \mathbf{c}_2, v_3) \bullet\!\!\rightarrow (v_1, \mathbf{c}_2, v_3) | v_1 \bullet\!\!\rightarrow \mathbf{c}_1 \in E_1, \mathbf{c}_2 \in C_2, v_3 \in V_3\}$
  $\cup \{(\mathbf{c}_1, v_2, \mathbf{c}_3) \bullet\!\!\rightarrow (v_1, v_2, \mathbf{c}_3) | v_1 \bullet\!\!\rightarrow \mathbf{c}_1 \in E_1, v_2 \in V_2, \mathbf{c}_3 \in C_3\}$
  $\cup \{(v_1, \mathbf{c}_2, \mathbf{c}_3) \bullet\!\!\rightarrow (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3) | v_1 \bullet\!\!\rightarrow \mathbf{c}_1 \in E_1, \mathbf{c}_2 \in C_2, \mathbf{c}_3 \in C_3\}$
– $E_Y = \{(v_1, v_2, v_3) \bullet\!\!\rightarrow (v_1, \mathbf{c}_2, v_3) | v_2 \bullet\!\!\rightarrow \mathbf{c}_2 \in E_2, v_1 \in V_1, v_3 \in V_3\}$
  $\cup \{(\mathbf{c}_1, \mathbf{c}_2, v_3) \bullet\!\!\rightarrow (\mathbf{c}_1, v_2, v_3) | v_2 \bullet\!\!\rightarrow \mathbf{c}_2 \in E_2, \mathbf{c}_1 \in C_1, v_3 \in V_3\}$
  $\cup \{(\mathbf{c}_1, v_2, \mathbf{c}_3) \bullet\!\!\rightarrow (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3) | v_2 \bullet\!\!\rightarrow \mathbf{c}_2 \in E_2, \mathbf{c}_1 \in C_1, \mathbf{c}_3 \in C_3\}$
  $\cup (v_1, \mathbf{c}_2, \mathbf{c}_3) \bullet\!\!\rightarrow (v_1, v_2, \mathbf{c}_3) | v_2 \bullet\!\!\rightarrow \mathbf{c}_2 \in E_2, v_1 \in V_1, \mathbf{c}_3 \in C_3\}$
– $E_Z = \{(v_1, v_2, v_3) \bullet\!\!\rightarrow (v_1, v_2, \mathbf{c}_3) | v_3 \bullet\!\!\rightarrow \mathbf{c}_3 \in E_3, v_1 \in V_1, v_2 \in V_2\}$
  $\cup \{(\mathbf{c}_1, \mathbf{c}_2, v_3) \bullet\!\!\rightarrow (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3) | v_3 \bullet\!\!\rightarrow \mathbf{c}_3 \in E_3, \mathbf{c}_1 \in C_1, \mathbf{c}_2 \in C_2\}$
  $\cup \{(\mathbf{c}_1, v_2, \mathbf{c}_3) \bullet\!\!\rightarrow (\mathbf{c}_1, v_2, v_3), | v_3 \bullet\!\!\rightarrow \mathbf{c}_3 \in E_3, \mathbf{c}_1 \in C_1, v_2 \in V_2\}$
  $\cup \{(v_1, \mathbf{c}_2, \mathbf{c}_3) \bullet\!\!\rightarrow (v_1, \mathbf{c}_2, v_3), | v_3 \bullet\!\!\rightarrow \mathbf{c}_3 \in E_3, v_1 \in V_1, \mathbf{c}_2 \in C_2\}$

La construction est résumée sur la figure 5.8.

Là aussi, il faut vérifier que ce produit est un code quantique valide :

Propriété 31 (Validité du code produit tridimensionnel)**.** *Soient $\mathscr{C}_1$, $\mathscr{C}_2$, $\mathscr{C}_3$ trois codes classiques, alors le produit $\mathscr{C}_q = \mathscr{C}_1 \boxtimes \mathscr{C}_2 \boxtimes \mathscr{C}_3$ est un code quantique valide, autrement dit pour tous $c, c' \in C$, on a*

$$\mathcal{S}(c) \star \mathcal{S}(c') = 0$$

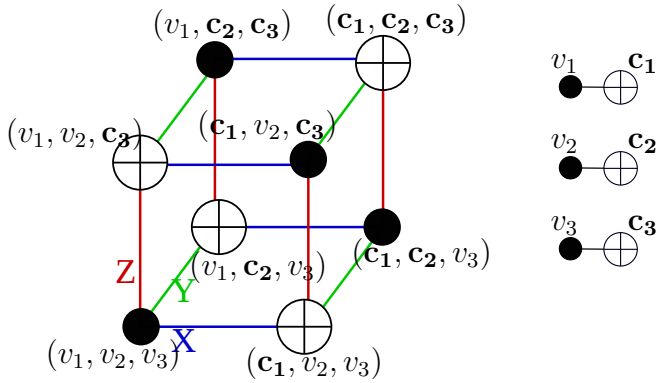


FIGURE 5.8 – Schéma de la construction du produit tridimensionnel : les petits graphes de Tanner à droite représentent 3 arêtes des 3 codes composants, et le schéma en « cube » est le produit de ces arêtes. On peut constater que lorsqu'on emprunte une arête en X, cela revient à changer la première coordonnée selon une arête du premier code, et de même en Y et Z.

Démonstration. Soient $c, c' \in C$. Soient ils sont du même type (appartiennent au même produit cartésien), par exemple $c = (c_1, c_2, c_3), c' = (c'_1, c'_2, c'_3) \in C_1 \times C_2 \times C_3$. Ces nœuds sont reliés :

- aux nœuds de type $V_1 \times C_2 \times C_3$ avec une arête en X,
- aux nœuds de type $C_1 \times V_2 \times C_3$ avec une arête en Y,
- aux nœuds de type $C_1 \times C_2 \times V_3$ avec une arête en Z.

Dans tous les cas, ils sont reliés aux mêmes nœuds variable avec les mêmes types d'arêtes, donc $\mathcal{S}(c)$ et $\mathcal{S}(c')$ commutent.

Supposons alors qu'ils soient de types différent. La construction étant symétrique, on peut supposer $c = (c_1, c_2, c_3) \in C_1 \times C_2 \times C_3$ et $c' = (v'_1, v'_2, c'_3) \in V_1 \times V_2 \times C_3$. Pour qu'ils soient reliés à des mêmes nœuds variable, il faut qu'ils aient au moins une coordonnée en commun, donc $c_3 = c'_3$. Supposons qu'ils soient reliés à un nœud de variable commun, disons de type $C_1 \times V_2 \times C_3$. Il s'agit forcément de $v = (c_1, v'_2, c_3)$, et cela signifie qu'il existe une arête de $\mathscr{G}(\mathscr{C}_1)$ $v'_1 \rightarrowtail c_1$, et une arête de $\mathscr{G}(\mathscr{C}_2)$ $v'_2 \rightarrowtail c_2$. Les deux nœuds de parité sont alors reliés à v avec deux arêtes de type différent (X et Y), mais cela signifie qu'il existe un autre nœud variable (et c'est le seul), $v' = (v'_1, c_2, c_3) \in V_1 \times C_2 \times C_3$, auquel les deux nœuds de parité c et c' sont reliés (voir figure 5.9), et avec des arêtes de type différent. Les deux nœuds de parité sont donc reliés à des nœuds variable par des arêtes de type différent en exactement deux positions, ce qui signifie que $\mathcal{S}(c)$ et $\mathcal{S}(c')$ commutent.

■

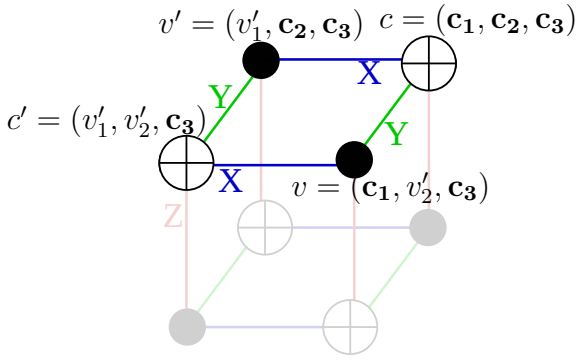


FIGURE 5.9 – Commutation de $\mathcal{S}(c_1, c_2, c_3)$ et $\mathcal{S}(v'_1, v'_2, c_3)$. Les deux nœuds de parité ont exactement deux nœuds variable communs, reliés dans les deux cas avec des arêtes de type différent.

5.4.2 Propriétés

Le produit tridimensionnel ressemble beaucoup à sa version bidimensionnelle, et on pourrait penser qu'il possède de bonnes propriétés équivalentes. Malheureusement, entre autres parce qu'il n'est pas CSS, il est en fait beaucoup plus compliqué, et il est très difficile de donner, de façon générale, la distance minimale et même simplement la dimension d'un produit tridimensionnel. Et quand on peut la donner, elle est loin d'être naturelle. Par exemple, on verra dans la section 5.4.3 que la dimension du code torique dans sa version tridimensionnelle dépend du PGCD des dimensions, ce qui n'était pas le cas dans la version d'origine.

Tout d'abord, concernant la dimension, on peut donner une borne inférieure :

Propriété 32 (Dimension du produit tridimensionnel – borne inférieure)**.** *Si $\mathscr{C}_1$, $\mathscr{C}_2$ et $\mathscr{C}_3$ sont trois codes classiques de graphes de Tanner $\mathscr{G}(\mathscr{C}_1) = (V_1, C_1, E_1)$, $\mathscr{G}(\mathscr{C}_2) = (V_2, C_2, E_2)$ et $\mathscr{G}(\mathscr{C}_3) = (V_3, C_3, E_3)$, alors la dimension k du produit $\mathscr{C}_1 \boxtimes \mathscr{C}_2 \boxtimes \mathscr{C}_3$ vérifie :*

$$k \geqslant k_1k_2k_3 + \bar{k}_1\bar{k}_2k_3 + \bar{k}_1k_2\bar{k}_3 + k_1\bar{k}_2\bar{k}_3 \tag{5.12}$$

où k_i (resp. $\bar{k}_i$) est la dimension (resp. la dimension duale) de $\mathscr{C}_i$.

Démonstration. Comme dans la preuve du théorème 4, on va compter le nombre de nœuds de parité redondants. Pour cela, on va estimer le nombre de nœuds de parité redondants dans chaque produit cartésien.

On commence par considérer $C_1 \times C_2 \times C_3$, et on définit le code classique $\tilde{\mathscr{C}}$ comme le produit *classique* des trois codes donnés par les graphes de Tanner suivants : $\mathscr{G}(\bar{\mathscr{C}_1}) = (C_1, V_1, E_1)$, $\mathscr{G}(\bar{\mathscr{C}_2}) = (C_2, V_2, E_2)$, $\mathscr{G}(\bar{\mathscr{C}_3}) = (C_3, V_3E_3)$ Ce

graphe de Tanner représente le sous-graphe de $\mathscr{G}(\mathscr{C})$ contenant uniquement les nœuds de parité de $C_1 \times C_2 \times C_3$, avec les nœuds de parité et nœuds variable inversés (voir figure 5.10).

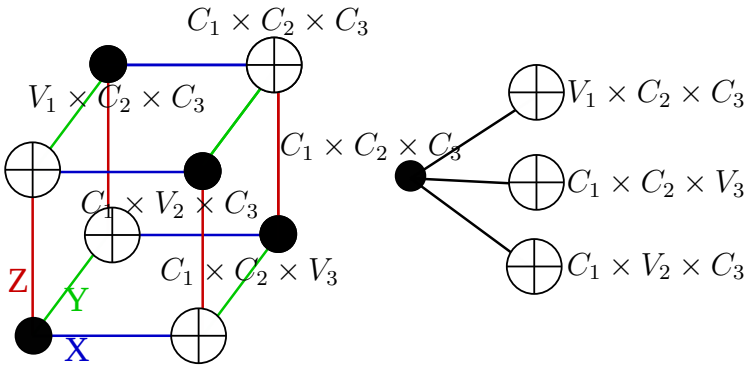


FIGURE 5.10 – Extrait du graphe de Tanner de $\mathscr{C}$ (à droite), en sélectionnant les nœuds de parité de $C_1 \times C_2 \times C_3$ et leurs nœuds variable reliés, puis en inversant nœuds variable et nœuds de parité.

Or la dimension de ce code $\tilde{\mathscr{C}}$ représente le nombre de nœuds de parité redondants dans le produit cartésien $C_1 \times C_2 \times C_3$ de $\mathscr{C}$. En effet, s'il existe un mot de code $w \in \tilde{\mathscr{C}}$ non nul, alors c'est que pour tout nœud de parité, il y a un nombre pair d'arêtes reliées à des nœuds de variable du support de w. Donc, dans $\mathscr{C}$, si on fait la somme de tous les nœuds de parité qui correspondent au support de w, il y a un nombre pair d'arêtes reliées à chaque nœud de parité (il n'y a pas de problème de type d'arêtes différentes, puisque dans le sous-graphe de $\mathscr{C}$ considéré, un nœud de variable n'est relié qu'à un seul type d'arête), et on obtient une combinaison triviale de nœuds de parité : il y a un nœud redondant.

Là aussi le théorème sur la dimension du code produit classique s'applique, et la dimension de $\tilde{\mathscr{C}}$ est le produit des dimensions des codes composants : ici $\bar{k}_1\bar{k}_2\bar{k}_3$.

On peut effectuer le même raisonnement pour les autres produits cartésiens de nœuds de parité, et on obtient $k_1k_2\bar{k}_3$, $k_1\bar{k}_2k_3$ et $\bar{k}_1k_2k_3$ pour respectivement $V_1 \times V_2 \times C_3$, $V_1 \times C_2 \times V_3$, $C_1 \times V_2 \times V_3$. En utilisant la relation $k = n - r + \bar{k}$, on obtient :

$$\begin{aligned}
k &= \text{nombre de nœuds variable} - \text{nombre de nœuds de parité} \\
&\quad + \dim(\text{nœuds de parité redondants}) \\
&\geqslant n_1n_2n_3 + r_1r_2n_3 + r_1n_2r_3 + n_1r_2r_3 \\
&\quad -(r_1r_2r_3 + n_1n_2r_3 + n_1r_2n_3 + r_1n_2n_3) \\
&\quad +\bar{k}_1\bar{k}_2\bar{k}_3 + k_1k_2\bar{k}_3 + k_1\bar{k}_2k_3 + \bar{k}_1k_2k_3 \\
&\geqslant (n_1 - r_1)(n_2 - r_2)(n_3 - r_3) \\
&\quad +\bar{k}_1\bar{k}_2\bar{k}_3 + k_1k_2\bar{k}_3 + k_1\bar{k}_2k_3 + \bar{k}_1k_2k_3 \\
&\geqslant (k_1 - \bar{k}_1)(k_2 - \bar{k}_2)(k_3 - \bar{k}_3) \\
&\quad +\bar{k}_1\bar{k}_2\bar{k}_3 + k_1k_2\bar{k}_3 + k_1\bar{k}_2k_3 + \bar{k}_1k_2k_3 \\
&\geqslant k_1k_2k_3 + \bar{k}_1\bar{k}_2k_3 + \bar{k}_1k_2\bar{k}_3 + k_1\bar{k}_2\bar{k}_3
\end{aligned}$$

■

La preuve ne donne pas d'égalité, cependant, ce résultat permet tout de même de construire des familles de codes de rendement positif quand les longueurs augmentent. En effet, si on choisit pour codes composants des codes de rendement R_1, R_2 et R_3 strictement positifs n'ayant pas de nœuds de parité redondants ($\bar{k}_i = 0$, et $k = r - n$), on obtient :

$$\begin{aligned}
R &\geqslant \frac{k_1k_2k_3}{n_1n_2n_3 + r_1r_2n_3 + r_1n_2r_3 + n_1r_2r_3} \\
&\geqslant \frac{R_1R_2R_3}{1 + (1 - R_1)(1 - R_2) + (1 - R_1)(1 - R_3) + (1 - R_2)(1 - R_3)}
\end{aligned}$$

C'est donc un rendement strictement positif et indépendant des n_i.

Pour obtenir une borne supérieure sur la dimension, il faudrait vérifier qu'il n'y a pas de nœuds de parité redondant « entre » les différents types de nœuds de parité. Ce problème n'apparaissait pas dans le cas du produit bidimensionnel puisque le code était CSS (les stabilisateurs en X ne pouvaient pas engendrer des stabilisateurs en Z et inversement). Or ici ce n'est plus le cas : des stabilisateurs composés de X et de Y peuvent éventuellement engendrer des stabilisateurs contenant des Z.

Par exemple, soit $\mathscr{C}$ un code composant tel que $\mathscr{G}(\mathscr{C}) = (\{1\}, \{1\}, \{1 \rightarrowtail 1\})$, le code le plus simple qu'on puisse imaginer : un nœud variable, un nœud de parité, une arête qui les relie. Alors, le produit tridimensionnel $\mathscr{C} \boxtimes \mathscr{C} \boxtimes \mathscr{C}$ donne exacte-

ment le graphe de Tanner de la figure 5.11. Il a pour matrice de stabilisateurs :

$$\mathbf{H} = \begin{pmatrix} X & Y & Z & I \\ I & Z & Y & X \\ Z & I & X & Y \\ Y & X & I & Z \end{pmatrix}$$

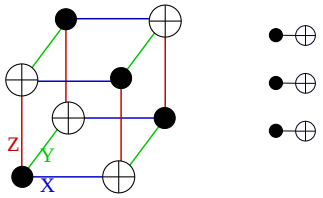


FIGURE 5.11 – Graphe de Tanner du code produit tridimensionnel obtenu à partir des codes composants dont les graphes sont dessinés à droite (chaque graphe est composé d'un nœud variable, d'un nœud de parité et d'une arête).

Or on constate que le produit de chacune des lignes donne $I^{\otimes 4}$, à la phase près (puisque $XYZ = I$), ce qui signifie que ce code a pour dimension au moins 1 (on peut également noter que la dimension est exactement 1). Or la dimension des petits codes composants est clairement 0, et la distance minimale duale est aussi 0. Ainsi, il est possible d'effectuer le produit tridimensionnel de 3 codes de dimension 0 et de distance minimale duale 0, et d'obtenir un code de dimension strictement positive.

De façon générale (on le verra plus en détail dans le cas particulier du code torique 3D dans la section 5.4.3), la dimension peut être très délicate à borner supérieurement.

On peut pourtant, dans certains cas, expliciter une base symplectique (partielle) des erreurs sérieuses. Une telle base, même incomplète, peut aider à donner une borne sur la distance minimale. De plus, sa forme caractéristique est simple à manipuler.

Proposition 1 (Base symplectique des erreurs sérieuses du code tridimensionnel)**.** *Soient $\mathscr{C}_1, \mathscr{C}_2, \mathscr{C}_3$ trois codes classiques. Chaque terme de la somme de l'équation* (5.12) *fait apparaître un produit de dimensions de la forme $k'_1 k'_2, k'_3$, avec $k'_i = \dim(\mathscr{C}_i)$ ou $k'_i = \dim(\overline{\mathscr{C}_i})$, on appelle ce code* code correspondant à k'_i*. Si, pour chaque terme de la somme, un des codes $\mathscr{C}'_i$ correspondants aux k'_i vérifie la propriété suivante, dite d'*orthogonalisation *:*

Définition 51. *Un code $\mathscr{C}$ vérifie la propriété d'*orthogonalisation *si il existe deux bases du code (non nécessairement distinctes) (u_i) et (v_i) telles que :*

$$\forall i, u_i\,{}^{\mathbf{t}}(v_i) = 1 \tag{5.13}$$
$$\forall i \neq j, u_i\,{}^{\mathbf{t}}(v_j) = 0 \tag{5.14}$$

Un code trivial (de dimension 0) est considéré comme vérifiant cette propriété.

Alors il existe une base des erreurs sérieuses $\bar{X}_i, \bar{Z}_i$ qui atteint la borne de l'équation (5.12).

La propriété d'orthogonalisation n'est pas automatique sur un code donné, même s'il suffit qu'un seul sur les trois codes composants ait cette propriété.

En effet, trouver deux bases qui vérifient $u_i\,{}^{\mathbf{t}}(v_j) = 0$ pour $i \neq j$ est toujours possible (par méthode d'orthogonalisation standard), mais la condition $u_i\,{}^{\mathbf{t}}(v_i) = 1$ est plus difficile à obtenir : par exemple si le code est auto-orthogonal, on ne peut pas avoir cette condition.

Il arrive parfois qu'on ait une propriété plus forte : qu'il existe une base *orthonormée* (pour le produit scalaire $(u|v) = u\,{}^{\mathbf{t}}(v)$) pour le code. La propriété d'orthogonalisation est alors vérifiée, en prenant $(u_i) = (v_i) =$ cette base orthonormée. Cependant, il peut exister des codes n'ayant pas de base orthonormée (par exemple, si tous les mots de code sont de poids pair), mais possédant malgré tout la propriété d'orthogonalisation. Par exemple, le code de longueur 3 et de dimension 2 engendré par les mots de code $u = 110$ et $v = 011$: on peut prendre comme bases u, v et v, u et la propriété est vérifiée puisque $v\,{}^{\mathbf{t}}(v) = 0$ et $u\,{}^{\mathbf{t}}(v) = 1$.

Démonstration. On suppose $k_1k_2k_3 \neq 0$, et on construit des éléments de la base ayant pour support $V_1 \times V_2 \times V_3$. Par symétrie de la construction, si les autres produits $\bar{k}_1\bar{k}_2, k_3$, $\bar{k}_1k_2\bar{k}_3$ ou $k_1\bar{k}_2\bar{k}_3$ respectivement sont non nuls, on peut obtenir de la même façon des éléments de la base ayant pour supports respectifs $C_1 \times C_2 \times V_3$, $C_1 \times V_2 \times C_3$ ou $V_1 \times C_2 \times C_3$.

Supposons également que $\mathscr{C}_1$ vérifie la propriété d'orthogonalité (voir définition 51). On écrit alors (u^i) et (v^i) deux telles bases. Pour les deux autres codes, il est toujours possible de construire une base *systématique* des mots de code : ie une base $u^1, \dots u^k$ telle que $u^i(j) = \delta_{i,j}$ pour $1 \leqslant i, j \leqslant k$. On appelle $(y^i)_{1\leqslant i\leqslant k_2}$ et $(z^i)_{1\leqslant i\leqslant k_3}$ les bases systématiques de $\mathscr{C}_2$ et $\mathscr{C}_3$.

Pour $1 \leqslant \alpha \leqslant k_1, 1 \leqslant \beta \leqslant k_2, 1 \leqslant \gamma \leqslant k_3$, on définit $\bar{Y}_{\alpha\beta\gamma}$:

$$\bar{Y}_{\alpha\beta\gamma}(v_1, v_2, v_3) = \begin{array}{l} \forall 1 \leqslant v_1 \leqslant n_1, 1 \leqslant v_2 \leqslant n_2, 1 \leqslant v_3 \leqslant n_3, \\ \left\{ \begin{array}{l} Y \text{ si } u^\alpha(v_1) = 1,\ v_2 = \beta,\ z^\gamma(v_3) = 1 \\ I \text{ sinon} \end{array} \right. \end{array}$$

Autrement dit, on choisit une coordonnée selon v_2 fixe, et sur le « plan » associé, on met un mot de code de $\mathscr{C}_1$ et $\mathscr{C}_3$ dans chaque direction (voir figure 5.12).

De même, on définit $\bar{Z}_{\alpha\beta\gamma}$:

$$\bar{Z}_{\alpha\beta\gamma}(v_1,v_2,v_3) = \begin{array}{l} \forall 1 \leqslant v_1 \leqslant k_1, 1 \leqslant v_2 \leqslant k_2, 1 \leqslant v_3 \leqslant k_3, \\ \left\{ \begin{array}{l} Z \text{ si } v^\alpha(v_1) = 1,\ y^\beta(v_2) = 1,\ v_3 = \gamma \\ I \text{ sinon} \end{array} \right. \end{array}$$

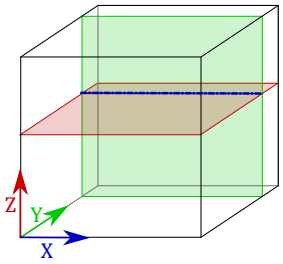


FIGURE 5.12 – Construction des $\bar{Y}_{\alpha\beta\gamma}, \bar{Z}_{\alpha\beta\gamma}$. Les deux plans s'intersectent sur (éventuellement) l'intersection de deux mots de code de $\mathscr{C}_1$

Il faut à présent vérifier plusieurs choses :

1. que les $\bar{Y}_{\alpha\beta\gamma}$ et les $\bar{Z}_{\alpha\beta\gamma}$ sont bien des erreurs indétectées,
2. que $\forall \alpha, \beta, \gamma$, on a bien $\bar{Y}_{\alpha\beta\gamma} \star \bar{Z}_{\alpha\beta\gamma} = 1$,
3. que $\forall (\alpha,\beta,\gamma) \neq (\alpha',\beta',\gamma')$, $\bar{Y}_{\alpha\beta\gamma} \star \bar{Z}_{\alpha'\beta'\gamma'} = 0$.

Pour le premier point, soit $\bar{Y}_{\alpha\beta\gamma}$. Un nœud variable de type $V_1 \times V_2 \times V_3$ est relié à trois sortes de nœuds de parité (voir schéma 5.8) :

- Les nœuds de parité de type $V_1 \times C_2 \times V_3$, par une arête en Y. L'erreur n'étant composée que de Y, elle n'est pas détectée par ces nœuds de parité.
- Ceux du type $C_1 \times V_2 \times V_3$, par une arête en X. Soit un nœud de parité $c = (c_1, v_2, v_3)$. Si $v_2 \neq \beta$ et v_3 n'est pas dans le support de z^γ, alors $\mathcal{S}(c)$ n'a pas de support qui intersecte l'erreur (il y a des X sur les positions (v_1, v_2, v_3) avec les mêmes v_2 et v_3). Sinon, les X sont sur des positions v_2 telles que $v_2 \rightarrowtail c_2$. Or les v_2 correspondants où il y a un Y sont sur le support d'un mot de code de $\mathscr{C}_1$: il y en a un nombre pair.
- Ceux du type $V_1 \times V_2 \times C_3$, par une arête en Z. Le raisonnement est le même, avec un mot de code de $\mathscr{C}_3$ cette fois.

Le raisonnement est symétrique pour les $\bar{Z}_{\alpha,\beta,\gamma}$.

Pour le deuxième point, soient $1 \leqslant \alpha \leqslant k_1, 1 \leqslant \beta \leqslant k_2, 1 \leqslant \gamma \leqslant k_3$.

$$\begin{aligned}
\bar{Y}_{\alpha\beta\gamma} \star \bar{Z}_{\alpha\beta\gamma} &= \sum_{v_1,v_2,v_3} \bar{Y}_{\alpha\beta\gamma}(v_1,v_2,v_3) \star \bar{Z}_{\alpha\beta\gamma}(v_1,v_2,v_3) \mod 2 \\
&= \sum_{v_1} \bar{Y}_{\alpha\beta\gamma}(v_1,\beta,\gamma) \star \bar{Z}_{\alpha\beta\gamma}(v_1,\beta,\gamma) \mod 2 \\
&= \sum_{v_1} u^\alpha(v_1) v^\alpha(v_1) \mod 2 \\
&= u^\alpha \, {}^{\mathbf{t}}(v^\alpha) \\
&= 1
\end{aligned}$$

Pour le troisième point, soient $(\alpha,\beta,\gamma) \neq (\alpha',\beta',\gamma')$.

$$\begin{aligned}
\bar{Y}_{\alpha\beta\gamma} \star \bar{Z}_{\alpha'\beta'\gamma'} &= \sum_{v_1,v_2,v_3} \bar{Y}_{\alpha\beta\gamma}(v_1,v_2,v_3) \star \bar{Z}_{\alpha'\beta'\gamma'}(v_1,v_2,v_3) \mod 2 \\
&= \sum_{v_1} \bar{Y}_{\alpha\beta\gamma}(v_1,\beta,\gamma') \star \bar{Z}_{\alpha'\beta'\gamma'}(v_1,\beta,\gamma') \mod 2
\end{aligned}$$

Or, $\forall v_1$, $\bar{Y}_{\alpha\beta\gamma}(v_1,\beta,\gamma')$ vaut I si γ' n'est pas dans le support du mot de code z^γ. Or comme on a pris une base systématique, si $\gamma \neq \gamma'$, $z^\gamma(\gamma') = 0$. donc $\bar{Y}_{\alpha\beta\gamma} \star \bar{Z}_{\alpha'\beta'\gamma'} = 0$. Pour le cas où $\beta \neq \beta'$, on a le même raisonnement avec la base systématique des mots de code de $\mathscr{C}_2$.

Reste donc le cas où $\beta = \beta', \gamma = \gamma'$ et $\alpha \neq \alpha'$. On a alors

$$\begin{aligned}
\bar{Y}_{\alpha\beta\gamma} \star \bar{Z}_{\alpha\beta\gamma} &= \sum_{v_1} u^\alpha(v_1) v^\alpha(v_1) \mod 2 \\
&= u^\alpha \, {}^{\mathbf{t}}(v^\alpha) \mod 2 \\
&= 0 \mod 2
\end{aligned}$$

par définition des u^α, v^α.

■

Pour la distance minimale, le résultat n'est malheureusement pas complet non plus : on peut obtenir une borne supérieure. Cela n'est pas particulièrement surpenant : la distance minimale et la dimension sont liées, on s'imagine difficilement pouvoir minorer le poids d'une erreur sérieuse si on ignore combien au maximum il y a de telles erreurs. De plus, c'est la formule sur la dimension qui donne la borne inférieure sur la distance minimale dans le cas CSS. Donc sans avoir de borne supérieure sur la dimension, il est difficile d'imaginer obtenir une borne inférieure sur la distance minimale. On peut cependant avoir une borne supérieure :

Proposition 2 (Distance minimale du produit tridimensionnel – borne supérieure). *Soient $\mathscr{C}_1$, $\mathscr{C}_2$ et $\mathscr{C}_3$ trois codes classiques, de distances minimales d_1, d_2 et d_3, et de co-distances minimales $\bar{d}_1$, $\bar{d}_2$ et $\bar{d}_3$. Alors, si $\mathscr{C}_1$ possède un* mot de code minimal non orthogonal *au sens de la définition suivante :*

Définition 52 (Mot de code minimal non orthogonal). *Soit $\mathscr{C}$ un code. $u \in \mathscr{C}$ est un* mot de code minimal non orthogonal *s'il est de poids atteignant la distance minimale et qu'il existe $v \in \mathscr{C}$ tel que :*

$$u.v = 1$$

on a alors la borne supérieure sur la distance minimale :

$$d \leqslant \min\{d_1 d_2, d_1 \bar{d}_2, d_1 d_3, d_1 \bar{d}_3\} \tag{5.15}$$

De même, si $\bar{C}_1$ possède un mot de code minimal non orthogonal, on a le résultat :

$$d \leqslant \min\{\bar{d}_1 d_2, \bar{d}_1 \bar{d}_2, \bar{d}_1 d_3, \bar{d}_1 \bar{d}_3\} \tag{5.16}$$

Et le résultat est valable en permutant les indices 1, 2 *et* 3.

Démonstration. La preuve est assez analogue à celle de la borne supérieure de la distance minimale du code produit (voir le théorème 5), et de la construction de la base symplectique (voir la proposition 1) : on construit une erreur à partir des mots de code des codes composants. Du fait de la symétrie de la construction (on peut inverser nœuds variable et nœuds de parité dans $\mathscr{C}_1$), (5.15) et (5.16) sont équivalentes.

Soit $w_2 \in \mathscr{C}_2$ de poids d_2, w_1 un mot de code de $\mathscr{C}_1$ minimal non orthogonal (on appellera $w_1^\star$ le mot de code de $\mathscr{C}_1$ qui vérifie $w_1\, {}^{\mathbf{t}}(w_1^\star) = 1$), et $w_3 \in \mathscr{C}_3$ quelconque. On appelle v_2^0 la première position telle que $w_3(v_2^0) = 1$ et v_3^0 la première position telle que $w_2(v_3^0) = 1$.

On va construire une erreur sérieuse E_Z de poids $d_1 d_2$, la construction pouvant s'adapter pour $d_1\bar{d}_2$, $d_1 d_3$, $d_1\bar{d}_3$. Cette erreur sera composée de Z uniquement, et aura pour support uniquement les positions de type $V_1 \times V_2 \times V_3$. La construction est très similaire à celle de la base des $\bar{X}_i, \bar{Z}_i$.

$$E_Z(v_1, v_2, v_3) = \begin{cases} Z \text{ si } u_1(v_1) = 1,\ u_2(v_2) = 1,\ v_3 = v_3^0 \\ I \text{ sinon} \end{cases}$$

Ainsi, comme dans le cas précédent, on sait qu'il s'agit d'une erreur indétectée. Pour montrer qu'il s'agit d'une erreur sérieuse, on va construire une erreur indétectée E_Y, ayant pour support les positions $V_1 \times V_2 \times V_3$ et composée de Y telle que $E_Z \star E_Y = 1$.

$$E_Y(v_1, v_2, v_3) = \begin{cases} Y \text{ si } u_1^\star(v_1) = 1,\ v_2 = v_2^0,\ u_3(v_3) = 1 \\ I \text{ sinon} \end{cases}$$

Comme dans le cas précédent (voir figure 5.12), on a $E_Y \star E_Z = u_1 \, {}^{\mathbf{t}}(u_1^\star) = 1$.

■

5.4.3 Un cas particulier : le code torique 3D

Comme dans le cas du produit classique, on peut prendre le code à répétition de taille n pour construire un premier exemple, raisonnablement simple, de code tridimensionnel. Comme dans le cas 2D, ce code a déjà été étudié (dans [BLT11] entre autres), et si ce code est nettement plus simple que le cas général cité plus haut, il n'est encore que partiellement compris.

Déjà, on peut constater un premier problème : alors que dans le cas du code torique habituel il n'y a aucun problème à prendre des dimensions égales en hauteur et largeur ($n_1 = n_2 \sim \sqrt{n}$), dans le cas tridimensionnel, il est nécessaire d'avoir des dimensions différentes (et même idéalement premières entre elles), tant pour la dimension que la distance minimale (mais comme les deux sont liées, ce n'est pas trop surprenant).

Ensuite, cette même distance minimale est très difficile à encadrer : on peut aisément la borner supérieurement par $n^{2/3}$ (en supposant que les trois dimensions sont $\sim n^{1/3}$), et inférieurement, un peu moins facilement, par $n^{1/3}$. Dans [BLT11], les auteurs prouvent, avec une construction d'erreur assez complexe, qu'elle est majorée par $n^{1/2}$. On ne sait pas actuellement quelle est la distance minimale de ce code, pourtant très simple à première vue.

Dans cette partie, on reprendra donc quelques aspects importants de ce code : bornes simples de dimension et de distance minimale ($n^{1/3}$ et $n^{2/3}$ dans les cas où cela s'applique).

Le graphe de Tanner de ce code est représenté partiellement sur la figure 5.13. Il est un peu difficile à voir en deux dimensions seulement, mais il « suffit » d'imaginer un pavage cubique de l'espace, qui boucle dans chacune des dimensions, et d'alterner les nœuds variable et nœuds de parité sur les sommets. Ensuite, pour le type des arêtes, on affecte chaque direction spatiale (verticale, horizontale, transversale) à un type d'arête.

Cependant, la définition à partir du produit tridimensionnel permet de mettre en valeurs différents « types » de nœuds de variable et de parité, et cette distinction peut être utile. Aussi on définira le code de la façon suivante :

Définition 53 (Code torique 3D)**.** *Soient n_1, n_2, n_3 trois entiers. On définit le code torique 3D $\mathscr{C}_{t3}$ de paramètres n_1, n_2 et n_3 par son graphe de Tanner $\mathscr{G}(\mathscr{C}_{t3}) = (V, C, E_X, E_Y, E_Z)$ de la façon suivante, avec les coordonnées de type (x, y, z) dans $\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \mathbb{Z}/n_3\mathbb{Z}$:*

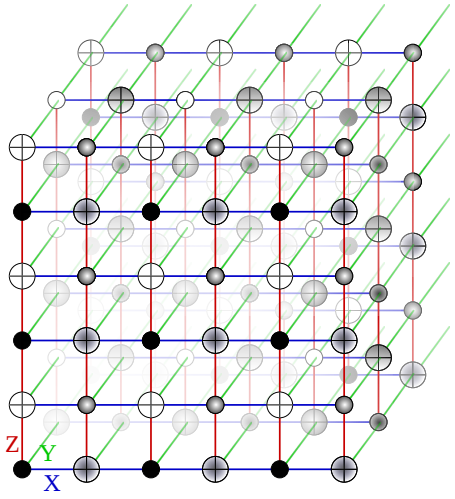


FIGURE 5.13 – Représentation graphique d'une partie du code torique en 3D. Les nœuds variable et de parité sont volontairement de différents gris pour mettre en valeur les différents produits cartésiens de nœuds ($V_1 \times V_2 \times V_3$, $V_1 \times C_2 \times C_3$, etc).

- $V = V_A \cup V_B \cup V_C \cup V_D$ *où*

$$\begin{aligned} V_A &= \{(x,y,z), x, y, z \textit{ pairs}\} \\ V_B &= \{(x,y,z), x, y \textit{ impairs}, z \textit{ pair}\} \\ V_C &= \{(x,y,z), x, z \textit{ impairs}, y \textit{ pair}\} \\ V_D &= \{(x,y,z), y, z \textit{ impairs}, x \textit{ pair}\} \end{aligned}$$

- $C = C_A \cup C_B \cup C_C \cup C_D$ *où*

$$\begin{aligned} C_A &= \{(x,y,z), x, y, z \textit{ impairs}\} \\ C_B &= \{(x,y,z), x, y \textit{ pairs}, z \textit{ impair}\} \\ C_C &= \{(x,y,z), x, z \textit{ pairs}, y \textit{ impair}\} \\ C_D &= \{(x,y,z), y, z \textit{ pairs}, x \textit{ impair}\} \end{aligned}$$

- $E_X = \{(x,y,z) \rightarrowtail (x \pm 1, y, z), (x,y,z) \in V\}$
- $E_Y = \{(x,y,z) \rightarrowtail (x, y \pm 1, z), (x,y,z) \in V\}$
- $E_Z = \{(x,y,z) \rightarrowtail (x, y, z \pm 1), (x,y,z) \in V\}$

On peut remarquer que cette définition est peut-être un peu lourde, et on pouvait se contenter de noter que $V = \{(x,y,z), x+y+z \text{ pair}\}$ et $C = \{(x,y,z), x+$

$y+z$ impair}. On le verra un peu plus loin, le fait de distinguer les différents types de nœuds est utile.

On se placera par la suite dans un cas où n_1, n_2, n_3 sont premiers entre eux deux à deux (et donc entre autres, deux de ces nombres sont impairs), car c'est dans ce cas que la distance minimale a des chances d'être la meilleure (voir plus bas), et donc c'est dans ce cas qu'on obtient les résultats les plus intéressants.

Voyons tout d'abord la dimension. Par la propriété 32, on sait qu'elle est d'au moins 4 (chaque code composant est de dimension 1 et de distance minimale duale 1). On peut prouver que, dans ce cas précis, la dimension est d'exactement 4 [1] :

Propriété 33 (Dimension du code torique 3D)**.** *Si n_1 et n_2 sont premiers entre eux, alors le code torique 3D de paramètres n_1, n_2, n_3 a pour dimension* 4*.*

Avant de faire cette preuve, on peut remarquer qu'il s'agit exactement de la borne donnée par la propriété 32. Cette remarque, bien qu'évidente, signifie une chose : les nœuds de parité redondants le sont *au sein d'un même type* de nœuds de parité (voir la preuve du théorème en question, on y traite de façon séparée les différents types de nœuds de parité). On peut d'ailleurs constater que, dans chaque ensemble de nœuds de parité, la somme de tous les stabilisateurs associés est nulle. En effet, considérons $C_A = \{(x, y, z), x, y, z \text{ impairs}\}$. Pour chaque nœud variable (x, y, z) de type V_B, il y a exactement deux nœuds de parité de C_A reliés : $(x, y, z \pm 1)$. Ainsi, si on fait le produit des stabilisateurs associés aux nœuds de parité de C_A, en chaque position de ce type, il y a un produit de deux Z. On a le même raisonnement sur les autres types de nœuds variable V_C, V_D avec Y et X, et on peut constater que ceux de type V_A ne sont pas du tout reliés à ces nœuds de parité-là. Ainsi, le produit de tous les stabilisateurs associés aux C_A donne l'identité.

La propriété 33 montre que ce sont les seules relations de redondance qu'on peut y trouver.

Démonstration. Pour montrer que la dimension est au plus 4, on commence par enlever les nœuds de parité redondants : on retire un nœud quelconque dans chaque produit cartésien.

Ensuite on va construire, pour chaque nœud de parité restant c, des erreurs E_c telles que $E_c \star \mathcal{S}(c) = 1$, et $\mathcal{E}_c \star \mathcal{S}(c') = 0$ pour tout autre nœud de parité $c' \neq c$. Cela montre bien qu'il n'y a pas d'autres redondances : si jamais $\mathcal{S}(c) = \prod_i \mathcal{S}(c_i)$, en utilisant E_c telle que décrite plus haut, alors on aurait $E_c \star \mathcal{S}(c) = \sum_i E_c \star \mathcal{S}(c_i) = 0$, ce qui n'est pas possible.

1. Il est montré de fait dans [BLT11] que cette dimension est en fait de 4 fois le PGCD de n_1, n_2, n_3

Pour cela, on va montrer qu'on peut construire, pour toute paire de nœuds de parité appartenant au même produit cartésien c et c', une erreur qui n'ait que ces deux nœuds insatisfaits : on parlera de « relier » ces nœuds. Il suffira alors de relier n'importe quel nœud de parité c à celui qui a été enlevé dans le même produit cartésien pour obtenir E_c recherchée.

On définit donc la famille d'erreurs suivantes, composées de Z uniquement : soient u_1 et u_2 deux entiers tels que $u_1 n_1 + u_2 n_2 = -1$.

$$\forall\ 0 \leqslant a \leqslant 2n_1 - 1, 0 \leqslant b \leqslant 2n_2 - 1, 0 \leqslant c \leqslant 2n_3 - 1,$$
$$a + b + c \text{ pair}, \quad 0 \leqslant i \leqslant 2u_1 n_1$$
$$\Delta^Z_{a,b,c} = Z \text{ sur les positions de type } (a + i \mod 2n_1, b + i \mod 2n_2, c)$$

Il s'agit d'une erreur « en diagonale », sur un plan d'altitude c, comme le montre la figure 5.14. Sa forme est telle que seuls quatre nœuds de parité sont insatisfaits : ceux aux extrémités de la chaîne, c'est à dire ceux de coordonnées $(a-1, b, c), (a, b-1, c)$ pour le début de la chaîne, et ceux de coordonnées $(a+i+1, b+i, c), (a+i, b+i+1, c)$ si la chaîne est de longueur i.

Comme on a choisi une chaîne de longueur $2u_1n_1$, ces coordonnées sont $(a + 2u_1n_1+1, b+2u_1n_1, c), (a+2u_1n_1, b+2u_1n_1+1, c)$. Or comme la première coordonnée est comptée modulo $2n_1$, et la seconde modulo $2n_2$, et puisque $u_1n_1 = -1 \mod n_2$, ces coordonnées sont donc : $(a+1, b-2, c), (a, b-1, c)$, ce qui signifie que le nœud de parité en $(a, b-1, c)$ est désormais satisfait (voir figure 5.14).

L'erreur $\Delta^Z_{a,b,c}$ permet donc de « relier » deux nœuds de parité, en positions $(a-1, b, c)$ et $(a+1, b-2, c)$. En combinant deux telles erreurs $\Delta^Z_{a,b,c}$ et $\Delta^Z_{a+2,b-2,c}$, on peut alors relier $(a-1, b, c)$ et $(a+3, b-4, c)$. En en combinant un nombre i, on peut relier $(a-1, b, c)$ et $(a+2i+1, b-2i-2, c)$. Comme n_1 et n_2 sont premiers entre eux, les coordonnées de la forme $(a+2i+1 \mod 2n_1, b-2i-2 \mod 2n_2, c)$ peuvent parcourir tous les nœuds de parité de la forme $(a + 2i, b + 2j, c)$. Or ces nœuds de parité sont tous ceux qui appartiennent au même produit cartésien (voir la définition 53) dans un même plan d'altitude c.

On peut donc relier deux nœuds de parité appartenant au même produit cartésien, pour peu qu'ils soient à la même altitude (troisième coordonnée fixe). Il reste à montrer qu'on peut également changer d'altitude. Pour cela, si on cherche à relier les nœuds (a, b, c) et $(d, e, c+2)$, on effectue une construction analogue à la figure 5.15, c'est-à dire qu'on introduit une erreur Y en position $(d, e, c+1)$, ce qui donne quatre nœuds de parité insatisfaits, en positions $(d+1, e, c+1), (d-1, e, c+1), (d, e, c), (d, e, c+2)$. Les deux premiers nœuds sont de même altitude et du même produit cartésien : on peut donc les relier. Puis on peut alors relier (d, e, c) à (a, b, c) puisqu'ils sont dans le même produit cartésien.

On peut donc, en adaptant cette construction, relier deux nœuds du même produit cartésien peu importe leur position, ce qui prouve la propriété 33.

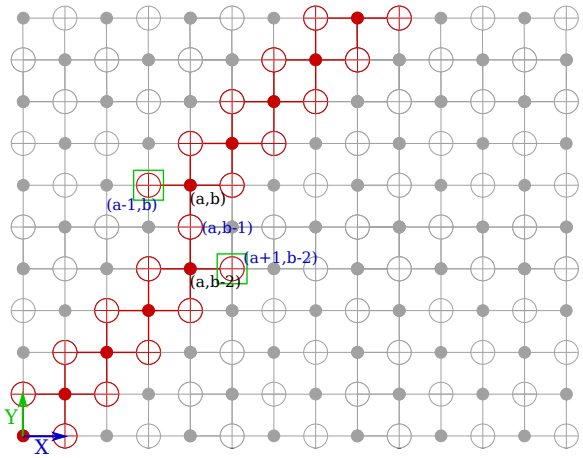


FIGURE 5.14 – Erreur dite « diagonale », sur un plan d'altitude fixée c (pour alléger le schéma, on n'a pas écrit la troisième coordonnée qui est constante). Les erreurs sont toutes des erreurs en Z, et on peut constater qu'à l'endroit où les deux diagonales se recoupent, il n'y a que deux nœuds de parité non satisfaits (carrés verts).

■

De plus, les codes composants $\mathscr{C}_i$, ainsi que les $\bar{\mathscr{C}}_i$ sont tous les deux le code à répétition de longueur n_i : ils n'ont qu'un mot de code, de poids n_i : pour peu que l'un des n_i soit impair (ce qui est le cas avec l'hypothèse qu'ils sont premiers entre eux), on a avec la proposition 1 une base symplectique des erreurs sérieuses, qu'on réécrit ici :

Proposition 3 (Base symplectique du code torique 3D)**.** *Soient n_1, n_2, n_3 premiers entre eux deux à deux, avec n_2 impair. La base suivante est une base symplectique du code torique 3D : $\bar{X}_{00}, \bar{X}_{01}, \bar{X}_{10}, \bar{X}_{11}$ et $\bar{Z}_{00}, \bar{Z}_{01}, \bar{Z}_{10}, \bar{Z}_{11}$ tels que :*

$$\bar{X}_{b_1 b_2}(x,y,z) = \begin{cases} X \text{ si } x = b_1 + b_2, \ y = b_1 \mod 2, \ z = b_2 \mod 2 \\ I \text{ sinon} \end{cases}$$

$$\bar{Z}_{b_1 b_2}(x,y,z) = \begin{cases} Z \text{ si } x = b_2 \mod 2, \ y = b_1 \mod 2, \ z = b_1 + b_2 \\ I \text{ sinon} \end{cases}$$

Démonstration. La preuve découle de la proposition 1.

■

La distance minimale, grâce à la propriété précédente, est bornée supérieurement par $\min\{n_1 n_2, n_2 n_3, n_1 n_3\}$ (ce qui donne, avec des n_i du même ordre de grandeur, une distance minimale en $n^{2/3}$ au maximum).

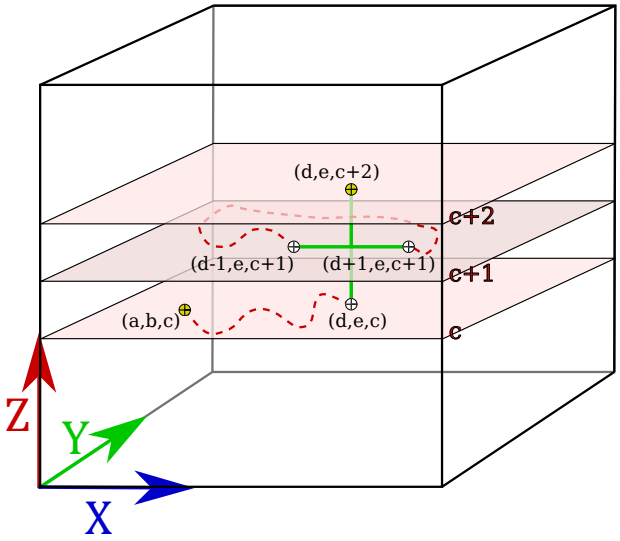


FIGURE 5.15 – Une façon de relier deux nœuds de parité appartenant au même produit cartésien (a, b, c) et $(d, e, c + 2)$: on commence par placer une erreur Y en $(a, b, c + 1)$, puis on relie les nœuds de parité (a, b, c) à (d, e, c), et également $(a-1, b, c+1)$ et $(a+1, b, c+1)$: les seuls nœuds de parité encore insatisfaits sont alors en (a, b, c) et $(d, e, c + 2)$ comme désiré.

On peut également remarquer que, si on choisit $n_1 = n_2$, alors la distance minimale est majorée par $2n_1$. En effet, on peut alors construire une erreur E_{diag}, composée uniquement de Z, avec pour support $\{(i, i, 0), i = 0, \dots n_1 - 1\}$. Cette erreur est indétectée, et de poids $2n_1$, il n'y a plus qu'à montrer qu'elle n'est pas dans le stabilisateur. Pour cela, on construit l'erreur $\bar{Y}$, composée uniquement de Y, et ayant pour support $\{(i, 0, k), i, k \text{ pairs}\}$. Cette erreur est basée sur le principe des $\bar{X}$ et $\bar{Z}$ de la base symplectique, et si on n'a pas de preuve qu'elle est bien dans $\mathrm{C}(\mathcal{S})\backslash\mathcal{S}$ (car la dimension n'est pas, a priori, 4), cette erreur est bien indétectée. Or on peut constater que $E_{\text{diag}} \star \bar{Y} = 1$ car ils ont une seule intersection commune (voir figure 5.16) : ces deux erreurs ne sont donc pas dans le stabilisateur, la distance minimale est donc d'au plus $2n_1$. On comprend alors l'intérêt de prendre des longueurs premières entre elles si on veut obtenir une distance minimale potentiellement intéressante.

On peut, de plus, borner inférieurement cette distance minimale par le minimum de n_1, n_2, et n_3 (donc une distance en $n^{1/3}$ au moins).

Proposition 4 (Borne inférieure sur la distance minimale du code torique 3D)**.** *Soient n_1, n_2, n_3 premiers entre eux deux à deux, avec n_2 impair. Alors la distance minimale du code produit est minorée par* $\min n_i$.

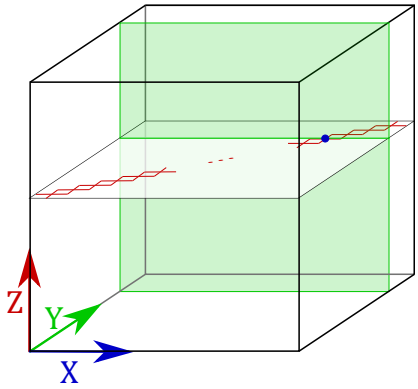


FIGURE 5.16 – Une erreur diagonale en Z (dans le cas où $n_1 = n_2$) qui intersecte une erreur en Y : l'erreur diagonale n'est pas dans le stabilisateur, on a donc une erreur non détectée qui n'est pas dans le stabilisateur, de poids $2n_1$.

Démonstration. Soit E une erreur sérieuse de poids minimal. Elle anti-commute avec au moins l'un des éléments de la base symplectique de la proposition 3. Supposons que $E \star \bar{X}_{00} = 1$, on va montrer qu'elle est au moins de poids n_1.

On construit alors n_1 représentations de $\bar{X}_{00}$:

$$\bar{X}_{00i} = \begin{cases} X \text{ si } x = 2i,\ y = 0 \mod 2,\ z = 0 \mod 2 \\ I \text{ sinon} \end{cases}$$

pour $0 \leqslant i \leqslant (n_1 - 1)$. Ces représentations sont distinctes, et sont toutes égales, à un élément du stabilisateur près, à $\bar{X}_{000} = \bar{X}_{00}$ (il suffit pour cela de vérifier les produits $\star$ avec les différents éléments de la base symplectique).

On constate entre autres que $\bar{X}_{00i} \star E = 1$ pour tout i, ce qui implique que E et $\bar{X}_{00i}$ ont au moins une intersection commune. Or, les $\bar{X}_{00i}$ sont de support disjoints : l'erreur E a donc au moins n_1 positions qui ne sont pas I (voir figure 5.17).

■

On peut noter que cette preuve fournit une méthode plus générale pour obtenir la distance minimale d'un code quantique :

Proposition 5 (Borne inférieure sur la distance minimale d'un code stabilisateur)**.** *Soit $\mathscr{C}$ stabilisé par $\mathcal{S}$ de dimension k, dont la base symplectique des erreurs sérieuses s'écrit $(\bar{X}_i)_i, (\bar{Z}_i)_i$. Supposons qu'il existe un m tel que pour tout i, il existe $\bar{X}_{i,1}, \bar{X}_{i,m}$ et $\bar{Z}_{i,1}, \bar{Z}_{i,m}$ tels que :*

- *$\bar{X}_{i,j}$ est égal à $\bar{X}_i$ modulo le stabilisateur, de même pour les $\bar{Z}_{i,j}$,*
- *$\bar{X}_{i,j}$ et $\bar{X}_{i,j'}$ sont à support disjoint si $j \neq j'$, de même pour les $\bar{Z}_{i,j}$.*

Alors la distance minimale du code quantique est d'au moins m.

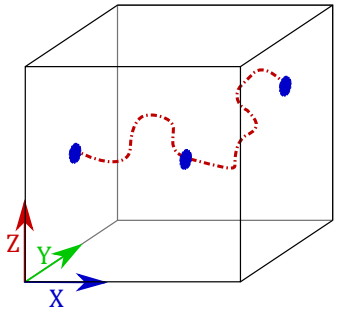


FIGURE 5.17 – Représentation d'une erreur sérieuse éventuelle (en pointillés rouge) et de plusieurs représentations de $\bar{X}_{00}$: chaque telle représentation doit intersecter au moins une fois l'erreur sérieuse, et on peut construire n_1 telles représentations, ce qui donne au moins n_1 positions non triviales pour l'erreur sérieuse.

Avec cette propriété, il n'est pas possible d'obtenir une borne supérieure à $\sqrt{n}$, où n est la longueur du code : en effet, supposons que la distance minimale soit d. Alors, si on fixe un i, chaque $\bar{X}_{i,j}$ est de poids supérieur ou égal à d. Or comme ils sont tous à support disjoints, on doit avoir une longueur d'au moins d^2.

Démonstration. La preuve est une généralisation de la preuve de la proposition 4. Soit E une erreur indétectée, n'appartenant pas au stabilisateur. Il existe donc un des $\bar{X}_i$ ou des $\bar{Z}_i$ avec lequel E anti-commute, on peut supposer qu'il s'agit de $\bar{X}_1$ sans perte de généralité.

Comme $\bar{X}_1 \star E = 1$, alors pour tout $1 \leqslant i \leqslant m$, $\bar{X}_{1,i} \star E = 1$. Chaque $\bar{X}_{1,i}$ a donc au moins une intersection commune avec E. Or les $\bar{X}_{1,i}$ sont à support disjoints : l'erreur E est donc au moins de poids m.

■

5.5 Autres variantes

Deux autres variantes du produit de codes sont étudiées ici, et développées dans d'autres chapitres : le code produit q-aire (voir 6.3, page 160), ainsi que le code produit spatialement couplé (voir 7.2, page 170).

Chapitre 6

Codes quantiques q-aires

On étudie dans ce chapitre une technique existante dans le monde classique, à savoir la construction, à partir d'un code binaire, d'un code sur une *extension* de $\mathbb{F}_2 : \mathbb{F}_q$, avec $q = 2^m$. On appelle ce nouveau code un *code q-aire*.

Ces codes sont connus pour avoir de meilleurs performances que les codes utilisés au départ (voir [DM98, Hu02, HEA05]), au moins dans le cas de codes 2-réguliers, c'est-à-dire qui possèdent exactement deux 1 par colonne dans la matrice de parité, ou encore dont le degré des nœuds variable est 2 dans le graphe de Tanner. De plus, il existe également des algorithmes de décodage très efficaces sur $\mathbb{F}_q$ (voir [BD03, VDV+10] entre autres).

La première adaptation de cette idée aux codes quantiques vient de [KHIK11], où les auteurs construisent des codes q-aires CSS à partir d'une famille de codes CSS bien particuliers, des codes quasi-cycliques 2-réguliers. La construction n'est pas si simple de façon générale : effectuer une construction q-aire à partir de chacun des codes formant un code CSS ne suffit pas, il faut encore garantir l'orthogonalité de ces deux nouveaux codes (voir la section 3.4.2). En s'inspirant de cette méthode, on la généralise à n'importe quel code CSS 2-régulier. De plus, en l'appliquant au code torique (voir section 4.1.3), on obtient des performances de décodage très bonnes, même avec l'algorithme de propagation de croyances standard, du moins sa variante q-aire (alors qu'elles sont très mauvaises sur le code torique usuel), et ce, sans perdre les bonnes propriétés de celui-ci.

6.1 Principe

6.1.1 Construction classique

La construction consiste à partir d'un code binaire existant, et d'en faire un code q-aire. Pour cela, il faut d'abord définir un code q-aire : la définition est exactement la même qu'en binaire (voir section 1.2.1), au corps près.

Définition 54 (Code correcteur q-aire, ou code sur $\mathbb{F}_q$)**.** *Soit le corps $\mathbb{F}_q$, avec $q = 2^m$, et une matrice H de taille $r \times n$ d'éléments de $\mathbb{F}_q$. Le code correcteur $\mathscr{C}$ ayant pour matrice de parité $\mathbf{H}$ est défini par :*

$$\mathscr{C} = \{x \in {\mathbb{F}_q}^n, \mathbf{H}\,{}^{\mathbf{t}}(x) = 0\}$$

où la multiplication matricielle est celle induite de la multiplication dans le corps $\mathbb{F}_q$. Ce code a pour longueur n, et pour dimension celle de l'espace défini par $\mathscr{C}$ (en tant que $\mathbb{F}_q$-espace vectoriel).

Pour construire un code q-aire $\mathscr{C}^q$ (avec matrice de parité $\mathbf{H}^q$) à partir d'un code binaire $\mathscr{C}$ (avec matrice de parité $\mathbf{H}$), on procède de la sorte : pour tous $1 \leqslant i \leqslant r, 1 \leqslant j \leqslant n$,

- si $\mathbf{H}_{i,j} = 0$, alors ${\mathbf{H}^q}_{i,j} = 0 \in \mathbb{F}_q$,
- si $\mathbf{H}_{i,j} = 1$, alors ${\mathbf{H}^q}_{i,j}$ est choisi aléatoirement parmi les éléments *non nuls* de $\mathbb{F}_q$.

Autrement dit on remplace simplement tous les 1 de la matrice $\mathbf{H}$ par des éléments aléatoires non-nuls de $\mathbb{F}_q$. Cette condition permet de garantir le caractère LDPC de $\mathscr{C}^q$, pour peu que $\mathscr{C}$ le soit.

Comme on manipulera essentiellement des codes LDPC, il est utile de définir le graphe de Tanner d'un code q-aire :

Définition 55 (Graphe de Tanner d'un code q-aire)**.** *Soit $\mathscr{C}$ un code q-aire de matrice de parité $\mathbf{H}$ de taille $r \times n$. On définit le graphe de Tanner de $\mathscr{C}$ $\mathscr{G}_\mathscr{C} = (V, C, E)$, où*

- *V est l'ensemble des* nœuds variable*, de taille n,*
- *C est l'ensemble des* nœuds de parité*, de taille r,*
- *$E = \{v \overset{e}{\multimap} c \mid v \in V, c \in C, \mathbf{H}_{c,v} = e, e \neq 0\}$ est l'ensemble des arêtes étiquetées. $l \in {\mathbb{F}_q}^*$ est appelée* l'étiquette *de e.*

Graphiquement, un graphe de Tanner d'un code q-aire est un graphe de Tanner ordinaire, avec des étiquettes (non nulles) sur chaque arête. Le passage d'un code binaire à un code q-aire consiste alors à étiqueter chaque arête d'un élément non nul de $\mathbb{F}_q$.

Dans la pratique, on a généralement affaire à un canal qui agit sur des bits et non sur des éléments de $\mathbb{F}_q$. Or, comme un élément de $\mathbb{F}_q$ s'identifie à un vecteur de m bits via le choix d'une base (cela revient à identifier $\mathbb{F}_q$ à l'espace vectoriel $(\mathbb{F}_2)^m$), on peut travailler sur un canal q-aire équivalent : si on travaille, par exemple, sur le canal binaire symétrique de paramètre p, la probabilité de passer de $(b_1, \ldots, b_m)$ à $(b'_1, \ldots, b'_m)$ est égale à

$$\prod_{i=1}^{m} \begin{cases} 1-p \text{ si } b'_i = b_i \\ p \text{ sinon} \end{cases}$$

On peut alors utiliser des variantes de l'algorithme de propagations de croyances pour les codes q-aires, comme celui de [BD03].

Une autre possibilité, équivalente dans le principe mais plus complexe en pratique, consiste à transformer le code q-aire en code binaire, de longueur augmentée de m.

Pour cela, on prend A un isomorphisme d'anneau : $\mathrm{A} : \mathbb{F}_q \to \mathrm{Im(A)} \subset \mathcal{M}_m(\mathbb{F}_2)$ l'ensemble des matrices carrées $m \times m$ sur $\mathbb{F}_2$. A est construit de la façon suivante :

On prend tout d'abord $(\alpha_1, \alpha_2, \ldots, \alpha_m)$ une base de $\mathbb{F}_q$, vu comme un $\mathbb{F}_2$-espace vectoriel (généralement, on choisira une base de la forme $(1, \alpha, \alpha^2, \ldots, \alpha^{m-1})$). On définit ensuite :

$$\begin{array}{rccl} \mathrm{A}: & \mathbb{F}_q & \to & \mathbb{F}_2^{m^2} \\ & x & \to & \mathrm{A}(x) = \left(\begin{array}{c|c|c|c} \alpha_1 x & \alpha_2 x & \ldots & \alpha_m x \end{array} \right) \end{array}$$

où les $\alpha_i x$ sont écrits en colonne dans la base $(\alpha_1, \ldots, \alpha_m)$. Autrement dit, $\mathrm{A}(x)$ est la matrice de multiplication par x dans la base donnée. Ainsi, l'application A est un isomorphisme :

- $\mathrm{A}(x+y)$ est la matrice de multiplication par $x+y$, donc est $\mathrm{A}(x) + \mathrm{A}(y)$,
- $\mathrm{A}(xy)$ est la matrice de multiplication par xy, donc est la composée des deux multiplications (par x et par y), c'est donc $\mathrm{A}(x)\,\mathrm{A}(y)$,
- $\mathrm{A}(x) = 0$ signifie que la multiplication par x est toujours égale à 0 : cela signifie que $x = 0$.

Avec ce morphisme A, on construit la matrice $\mathbf{H}'$ telle que

$$\begin{gathered} \forall \quad 1 \leqslant i \leqslant r,\ 1 \leqslant j \leqslant n,\ 1 \leqslant i' \leqslant m,\ 1 \leqslant j' \leqslant m, \\ \mathbf{H}'_{m(i-1)+i', m(j-1)+j'} = \mathrm{A}(\mathbf{H}^q{}_{i,j})_{i',j'} \end{gathered}$$

Autrement dit, on remplace chaque élément de $\mathbf{H}^q$ par son image par A, et on obtient la nouvelle matrice (de dimensions $rm \times nm$).

Le code binaire équivalent est alors « le même » que le code q-aire, au sens où, si on prend x^q un mot de code q-aire, il vérifie :

$$\mathbf{H}^q \, {}^{\mathbf{t}}(x^q) = 0 \text{ dans } \mathbb{F}_q \iff \mathbf{H}' \, {}^{\mathbf{t}}(x') = 0$$

où x' est le vecteur ligne de longueur mn, obtenu en écrivant chaque entrée x_i^q de x^q comme un petit vecteur binaire de longueur m.

On verra plus bas que cet isomorphisme est en fait très utile pour les codes quantiques.

6.1.2 Construction quantique

En quantique, on peut appliquer cette construction à un code CSS, qui est composé de deux codes classiques binaires. Cependant, comme dans toute construction CSS, il ne suffit pas de contruire une paire de codes, il faut encore garantir l'orthogonalité des matrices de parité.

La méthode de base est assez similaire : on part d'une paire de matrices binaires $\mathbf{H}_X, \mathbf{H}_Z$, qui vérifient

$$\mathbf{H}_X \, {}^{\mathbf{t}}(\mathbf{H}_Z) = 0 \tag{6.1}$$

Puis on plonge ces matrices dans $\mathbb{F}_q$ et on obtient $\mathbf{H}_X^q, \mathbf{H}_Z^q$, en effectuant la même manipulation que dans le cas classique, *mais en gardant la condition*

$$\mathbf{H}^q{}_X \, {}^{\mathbf{t}}(\mathbf{H}^q{}_Z) = 0 \tag{6.2}$$

Enfin, on transforme ces matrices en matrices binaires agrandies, de façon à ce que les matrices finales $\mathbf{H}'_X$ et $\mathbf{H}'_Z$ vérifient

$$\mathbf{H}'_X \, {}^{\mathbf{t}}(\mathbf{H}'_Z) = 0 \tag{6.3}$$

L'isomorphisme d'anneau $\mathbb{F}_q \to \mathcal{M}_m(\mathbb{F}_2)$ décrit plus haut prend alors son sens : c'est avec cet isomorphisme qu'on effectue la dernière étape, et il permet d'obtenir automatiquement la condition (6.3) à partir de la condition (6.2).

L'étape $\mathbb{F}_q \to \mathbb{F}_2$ étant simple, la principale difficulté consiste donc à choisir dans $\mathbb{F}_q$ les valeurs non-nulles des matrices $\mathbf{H}_X^q$ et $\mathbf{H}_Z^q$ de telle sorte qu'on ait (6.2).

6.2 Codes quantiques q-aires de cycles

On étudie ici le cas particulier de codes dits *de cycles*. Ce sont des codes LDPC dont la matrice de parité possède exactement deux 1 par colonne, ou encore dont le degré des nœuds variable du graphe de Tanner est 2. On les nomme ainsi car, dans le cas de codes binaires classiques, les mots de codes correspondent alors des cycles (ou des unions de cycles) du graphe de Tanner.

Dans le cas quantique CSS, ce qu'on appelle les mots de code (et qui sont les erreurs sérieuses) sont des cycles *modulo* un ensemble de cycles (qui correspond au stabilisateur). Par exemple, dans le cas du code torique (présenté en premier lieu

en section 4.1.3), les erreurs sérieuses sont les cycles du graphe, modulo les petits cycles « évidents » de longueur 8, ce sont donc des cycles qui font au moins le tour du tore.

Les codes q-aires classiques obtenus à partir de codes de cycles ont alors de bonnes propriétés, notamment leur distance minimale ne diminue pas (bien qu'elle soit logarithmique en général), et on peut utiliser des algorithmes qui effectuent le décodage directement sur $\mathbb{F}_q$, qui sont très efficaces (voir [Hu02, HEA05, BD03, VDV$^+$10] entre autres). Tout porte à croire qu'une version quantique serait très efficace elle aussi, pour peu qu'on arrive à obtenir la condition (6.2).

Cette construction, présentée dans [AMT12a] est en deux parties. Tout d'abord, on étudie une construction générale, inspirée de [KHIK11]. Dans ce dernier résultat, les auteurs construisent des codes q-aires quantiques à partir d'une famille très particulière de codes, des codes quasi-cycliques ; ici la méthode fonctionne pour tous les codes CSS de cycles. Cependant, on ne connaît que partiellement les propriétés de ces codes q-aires construits. Ensuite, on étudie cette construction dans le cas du code torique. Le code ainsi obtenu obtient des performances en pratique très bonnes sous l'algorithme de propagations de croyances (alors qu'elles sont très mauvaises dans le cas binaire), et on peut prouver que non seulement sa dimension est augmentée (multipliée par m), mais sa distance minimale n'est pas diminuée.

6.2.1 Construction générale

On part de deux matrices $\mathbf{H}_X = (c_{i,j})_{i,j}$ et $\mathbf{H}_Z = (d_{i,j})_{i,j}$, de dimensions $r_X \times n$ et $r_Z \times n$ avec $c_{i,j}, d_{i,j} \in \mathbb{F}_2$, et on cherche à construire $\mathbf{H}_X^q = (\gamma_{i,j})_{i,j}, \mathbf{H}_Z^q = (\delta_{i,j})_{i,j}$ avec $\gamma_{i,j}, \delta_{i,j} \in \mathbb{F}_q$ telles que :

- Si $c_{i,j} = 0$, alors $\gamma_{i,j} = 0 \in \mathbb{F}_q$, et de même pour $d_{i,j}$ et $\delta_{i,j}$,
- Si $c_{i,j} = 1$, alors $\gamma_{i,j} \neq 0 \in \mathbb{F}_q$, et de même pour $d_{i,j}$ et $\delta_{i,j}$,

On cherche à assigner les étiquettes de sorte que $\mathbf{H}_X^q\ {}^{\mathbf{t}}(\mathbf{H}_Z^q) = 0$. Soit $1 \leqslant k \leqslant r_X$ un entier, et $\Delta(k)$ la k-ième ligne de $\mathbf{H}_Z^q$: $\Delta(k) = (\delta_{k,1}, \ldots, \delta_{k,n})$. On doit alors, pour tout tel k, vérifier :

$$\mathbf{H}_X^q\ {}^{\mathbf{t}}(\Delta(k)) = 0$$

Ce système est équivalent au système restreint suivant :

$$\widetilde{\mathbf{H}_X^q}\ {}^{\mathbf{t}}\Big(\widetilde{\Delta(k)}\Big) = 0$$

où le vecteur $\widetilde{\Delta(k)}$ est le vecteur $\Delta(k)$ privé de ses entrées nulles : $\widetilde{\Delta(k)} = (\delta_{k,i}, \delta_{k,i} \neq 0 \in \mathbb{F}_q)$, et $\widetilde{\mathbf{H}_X^q}$ est la matrice $\mathbf{H}_X^q$ à laquelle on n'a gardé que les colonnes dont les indices sont gardés dans $\widetilde{\Delta(k)}$, et ensuite les lignes contenant au moins

une entrée non-nulle. Informellement, cela signifie qu'on a enlevé toutes les lignes et les colonnes qui n'interviennent pas dans le système.

Il faut donc trouver une affectation dans $\mathbb{F}_q$ des termes non-nuls pour que ces sytèmes soient vérifiés. Comme chaque ligne $\Delta(k)$ est différente, il suffit de faire en sorte que, pour tout k, chacun de ces systèmes ait une solution non-triviale, et que cette solution soit non-nulle partout.

Or on a le fait suivant :

Lemme 4. *Pour tout k, dans la matrice $\widetilde{\mathbf{H}_X^q}$ correspondante, il y a un nombre pair de termes non-nuls par ligne, et exactement deux termes non-nuls par colonne.*

Démonstration. Pour le premier point, il suffit de reprendre le système

$$\widetilde{\mathbf{H}_X^q}\,{}^{\mathbf{t}}\Big(\widetilde{\Delta(k)}\Big) = 0$$

dans sa version binaire, qui est vérifié (puisque $\mathbf{H}_X\,{}^{\mathbf{t}}(\mathbf{H}_Z) = 0$) : le vecteur colonne réduit contient uniquement des 1, donc le nombre de 1 dans chaque ligne de la matrice $\mathbf{H}_X$ réduite doit être pair, d'où le nombre pair de termes non-nuls dans une ligne de $\widetilde{\mathbf{H}_X^q}$.

Pour le second point, lors de la réduction de la matrice $\mathbf{H}_X^q$, on a ôté uniquement les lignes vides : il y avait au départ exactement deux 1 par colonne, c'est toujours le cas.

■

On peut alors construire le graphe de Tanner associé à la matrice de parité réduite $\widetilde{\mathbf{H}_X^q}$: c'est un graphe dont le degré des nœuds variable est 2, et le degré des nœuds de parité est pair : il existe donc, pour chaque composante connexe, un chemin eulérien. Autrement dit, le graphe se décompose en un ensemble disjoint de cycles, ce qui veut dire qu'on peut réécrire $\widetilde{\mathbf{H}_X^q}$ de la façon suivante :

$$\begin{pmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_m \end{pmatrix}$$

avec C_i des matrices dont les graphes de Tanner correspondants sont des cycles. On supposera ici que les cycles ne se recoupent pas, c'est-à-dire que toutes les matrices C_i sont de la forme :

$$C_i = \begin{pmatrix} \gamma_1 & \gamma_2 & & & \\ & \gamma_3 & \gamma_4 & & \\ & & \ddots & \ddots & \\ \gamma_{2l} & & & & \gamma_{2l-1} \end{pmatrix}$$

Ainsi, le système se réécrit :

$$\begin{pmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_m \end{pmatrix} \begin{pmatrix} \widetilde{v_1} \\ \widetilde{v_2} \\ \vdots \\ \widetilde{v_m} \end{pmatrix} = 0$$

Si les cycles se recoupent, alors on écrira quand même la matrice sous cette forme, quitte à y rajouter des lignes (voir figure 6.1). Cela revient à transformer, par exemple, une équation du type :

$$\gamma_1\delta_1 + \gamma_2\delta_2 + \gamma_3\delta_3 + \gamma_4\delta_4 = 0$$

en :

$$\gamma_1\delta_1 + \gamma_2\delta_2 = 0$$
$$\gamma_3\delta_3 + \gamma_4\delta_4 = 0$$

La condition qu'on cherche sera alors une condition suffisante et pas forcément nécessaire (elle ne le serait que si tous ces cycles sont des cycles simples).

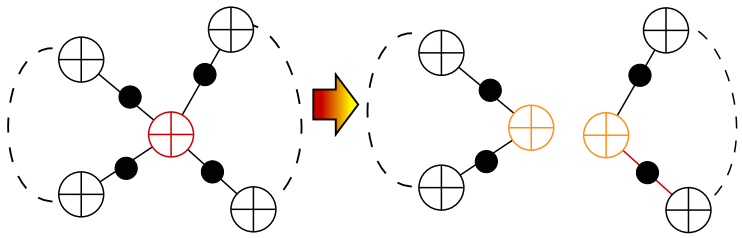


FIGURE 6.1 – Un cas où on a affaire à non pas un simple cycle, mais un cycle qui se recoupe : quitte à ajouter des nœuds de parité (ce qui correspond à des lignes de la matrice de parité), on se ramène au cas où on n'a que des cycles simples.

Le système est alors équivalent à m sous-systèmes $C_i\,{}^{\mathbf{t}}(v_i) = 0$, et on cherche à affecter des valeurs aux entrées non-nulles des C_i de façon à ce que tous les v_i soient non-nuls, et plus précisément que *toutes les entrées* des v_i soient non-nulles. Cela est assuré par la propriété suivante :

Propriété 34. *Le système suivant :*

$$C\,{}^{\mathbf{t}}(v) = 0$$

avec

$$C = \begin{pmatrix} \gamma_1 & \gamma_2 & & & \\ & \gamma_3 & \gamma_4 & & \\ & & \ddots & \ddots & \\ \gamma_{2l} & & & & \gamma_{2l-1} \end{pmatrix}$$

où les γ_i sont non-nuls possède une solution non triviale si et seulement si on a :

$$\gamma_1\gamma_2^{-1}\gamma_3\gamma_4^{-1}\dots\gamma_{2l-1}\gamma_{2l}^{-1} = 1 \tag{6.4}$$

Toutes les entrées du vecteur v sont alors non-nulles.

Démonstration. Pour avoir une solution non-triviale, il faut et il suffit que la matrice C ait un déterminant nul. On a donc :

$$\begin{aligned} & \det C = 0 \\ \Leftrightarrow \quad & \gamma_1\gamma_3\dots\gamma_{2l-1} + \gamma_2\gamma_4\dots\gamma_{2l} = 0 \\ \Leftrightarrow \quad & \gamma_1\gamma_2^{-1}\gamma_3\gamma_4^{-1}\dots\gamma_{2l-1}\gamma_{2l}^{-1} = 1 \end{aligned}$$

Dans le cas où cette condition est vérifiée, il existe donc une solution non-triviale v, reste à montrer que toutes ses entrées sont non-nulles. Supposons qu'il existe un indice i_0 tel que $v_{i_0} = 0$. Quitte à réorganiser la matrice, on peut supposer qu'il s'agit de l'indice 1.

La deuxième ligne du système s'écrit alors $\gamma_1 v_1 + \gamma_2 v_2 = 0$, ce qui implique donc, puisque les γ_i sont non-nuls, que $v_2 = 0$. Ensuite, la troisième ligne s'écrit : $\gamma_3 v_2 + \gamma_4 v_3 = 0$, ce qui implique $v_3 = 0$, et ainsi de suite : tous les v_i sont nuls, ce qui contredit l'hypothèse de départ.

■

On peut remarquer que, sur le graphe de Tanner associé à la matrice C de la propriété, le produit apparaissant dans l'equation (6.4) est le produit des étiquettes des arêtes le long d'un cycle ; on appellera ce produit *produit sur un cycle* :

Définition 56 (Produit sur un cycle)**.** *Soit un cycle $C = v_1, c_1, v_2, c_2, \dots c_l, v_1$ sur un graphe de Tanner dans $\mathbb{F}_q$. Le produit sur ce cycle est le produit des étiquettes des arêtes qui apparaissent dans le cycle, avec une puissance 1 si l'arête apparaît dans le sens nœud variable vers nœud de parité, et avec une puissance -1 sinon (voir figure 6.2).*

La condition suffisante pour avoir des entrées non-nulles des matrices $\mathbf{H}_X^q$ et $\mathbf{H}_Z^q$ serait donc que, pour tout k, pour tous les cycles intervenant dans le graphe de Tanner de $\widetilde{\mathbf{H}_X^q}$ correspondant, le produit sur ces cycles soit égal à 1.

On impose donc une condition un peu plus forte et plus simple à remplir : que cette condition soit vraie pour *tous* les cycles du graphe de $\mathbf{H}_X^q$.

Propriété 35. *Si on assigne les valeurs non-nulles à $\mathbf{H}_X^q$ de façon à ce que le produit sur tous les cycles soit égal à 1, on peut trouver une assignation de $\mathbf{H}_Z^q$ de façon à ce que les entrées désirées non-nulles le soient.*

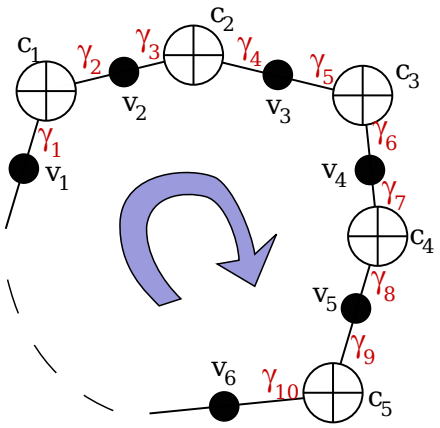


FIGURE 6.2 – Pour calculer le produit sur un cycle à partir du graphe, on fait le « tour » du cycle en comptant avec une puissance -1 une arête sur deux : ici le produit vaut $\gamma_1\gamma_2^{-1}\gamma_3\gamma_4^{-1}\ldots$ etc.

Démonstration. Si le produit de tous les cycles est égal à 1 dans le graphe de Tanner de $\mathbf{H}_X^q$, alors c'est le cas pour chacun des sous-graphes représentés par les sous-matrices $\widetilde{\mathbf{H}_X^q}$. Ainsi, on peut résoudre tous les petits systèmes du type $C_i\,{}^{\mathbf{t}}(v_i) = 0$ avec des entrées toutes non-nulles.

■

Reste à présent à assigner les coefficients de la matrice $\mathbf{H}_X^q$ de façon à vérifier cette condition de cycles.

6.2.2 Construction d'un graphe vérifiant la condition de cycles

Il n'est pas évident, a priori, de trouver une assignation des arêtes non triviale qui vérifie cette condition. Pour ce faire, il faut étudier d'un point de vue plus général les cycles d'un graphe et ses étiquetages.

Tout d'abord, le groupe multiplicatif $\mathbb{F}_q{}^*$ est cyclique ; il est donc isomorphe au groupe additif $\mathbb{Z}/r\mathbb{Z}$ avec $r = q - 1$. La condition devient alors « la somme de toutes les étiquettes sur les arêtes est égale à 0 », avec un « $-$ » lorsque l'arête est prise dans le sens nœud de parité vers nœud variable, et un « $+$ » sinon.

Il se trouve qu'on aura parfois besoin de la multiplication dans $\mathbb{Z}/r\mathbb{Z}$, ainsi on se placera dans l'*anneau* $\mathbb{Z}/r\mathbb{Z}$, même si la multiplication correspondante dans $\mathbb{F}_q{}^*$ n'est pas très naturelle ; elle est utile pour comprendre le fonctionnement des étiquetages.

Dans cette section, on présente le formalisme qui permet de comprendre comment ces étiquetages fonctionnent : on définit tout d'abord un graphe orienté, puis un cycle, puis un étiquetage dans $\mathbb{Z}/r\mathbb{Z}$, pour enfin comprendre ce que signifie la condition de cycles dans ce formalisme, et trouver une méthode pour construire *tous* les étiquetages vérifiant cette condition. Dans notre cas, on pourra alors construire un étiquetage aléatoire.

Cette construction est assez classique, et est traitée dans [Gib] dans les anneaux $\mathbb{Z}/2\mathbb{Z}$ (qui se trouve être un corps) et $\mathbb{Z}$. Elle se généralise facilement au cas de l'anneau $\mathbb{Z}/r\mathbb{Z}$[1], et on choisit de la refaire ici, entre autres par commodité de lecture.

6.2.2.1 Graphe et chaînes

Définition 57 (Graphe orienté)**.** *Un graphe orienté $\mathscr{G}$ est la donnée de (V, E) avec :*

- *$V = \{v_1, \dots v_n\}$ est l'ensemble des sommets,*
- *$E = \{u \rightarrowtail v | u, v \in V, e \in \mathscr{G}\}$ l'ensemble des* arêtes orientées.

vérifiant les conditions suivante :

$$\begin{aligned} \forall u, v \in V, \qquad & u \rightarrowtail v \Rightarrow v \rightarrowtail u \notin E \\ \forall u \in V, \qquad & u \rightarrowtail u \notin E \end{aligned}$$

Autrement dit, il s'agit d'un graphe général (notamment, il n'est pas forcément biparti comme c'est le cas des graphes de Tanner), sans arête reliant un sommet à lui-même, pour lequel on choisit une orientation des arêtes.

Par exemple, le graphe de la figure 6.3 correspond à $V = \{a, b, c, d, e, f\}$ et $E = \{a \rightarrowtail c, a \rightarrowtail f, b \rightarrowtail a, d \rightarrowtail a, b \rightarrowtail c, b \rightarrowtail d, e \rightarrowtail b, c \rightarrowtail e, f \rightarrowtail d, e \rightarrowtail f\}$. Dans le cas des graphes de Tanner, on a une orientation naturelle, du fait que le graphe est biparti : celle nœud de variable vers nœud de parité.

On définit à présent les *0-chaînes* (respectivement les *1-chaînes*), qui sont des combinaisons linéaires (dans $\mathbb{Z}/r\mathbb{Z}$) de sommets (respectivement d'arêtes) d'un graphe orienté.

Définition 58 (0-chaîne)**.** *Soit $\mathscr{G} = (V, E)$ un graphe orienté. Une 0-chaîne dans $\mathscr{G}$ est une somme formelle de la forme :*

$$\alpha_1 v_1 + \dots \alpha_k v_k$$

avec les $\alpha_i \in \mathbb{Z}/r\mathbb{Z}$, $v_i \in V$.

1. En fait, la construction décrite par la suite peut se généraliser à n'importe quel anneau unitaire.

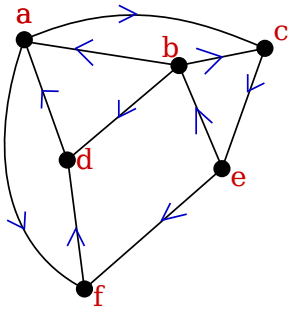


FIGURE 6.3 – Un exemple de graphe orienté.

On munit cet ensemble de la somme naturelle $(\alpha_1 v_1 + \ldots + \alpha_k v_k) + (\beta_1 w_1 + \ldots + \beta_l w_l) = \alpha_1 v_1 + \ldots \alpha_k v_k + \beta_1 w_1 + \ldots + \beta_l w_l$, *et on pose* $\alpha v + \beta v = (\alpha + \beta)v$. *On pose également* $\beta(\sum \alpha_i v_i) = \sum(\beta\alpha_i)v_i$. *On note* 0 *la chaîne triviale, et* $v = 1v$.

Définition 59 (1-chaîne)**.** *Soit* $\mathscr{G} = (V, E)$ *un graphe orienté. Une 1-chaîne dans* $\mathscr{G}$ *est une somme formelle de la forme :*

$$\alpha_1 e_1 + \ldots + \alpha_k e_k$$

avec les $\alpha_i \in \mathbb{Z}/r\mathbb{Z}$, $e_i \in E$. *On écrira* $(u \rightarrowtail v) = -1\,(v \rightarrowtail u)$, *ainsi la somme peut s'étendre aux arêtes de* $\bar{E} = \{u \rightarrowtail v, v \rightarrowtail u \in E\}$.

On munit cet ensemble des lois équivalentes à celles des 0-chaînes.

Aucune condition n'est imposée sur ces chaînes, notamment dans le cas des 1-chaînes, il n'y a aucune restriction sur le fait que les arêtes doivent se « toucher » ou non. Par exemple, dans le graphe de la figure 6.3, $3\,(a \rightarrowtail b) - (b \rightarrowtail d) + 1\,(e \rightarrowtail f)$ est une 1-chaîne dans $\mathbb{Z}/7\mathbb{Z}$. On peut également remarquer que si on se place dans $\mathbb{Z}/2\mathbb{Z}$, alors cela revient à ne pas tenir compte de l'orientation des arêtes, choisie, puisque $-1 = +1$ dans cet anneau.

Les lois définies avec ces chaînes (addition, multiplication par un scalaire de $\mathbb{Z}/r\mathbb{Z}$) font de ces ensembles des $\mathbb{Z}/r\mathbb{Z}$-*modules*, qui sont des équivalents d'espaces vectoriels, avec un anneau au lieu d'un corps pour l'ensemble des scalaires :

Définition 60 (Module sur un anneau A)**.** *Soit un anneau unitaire* $(A, +_A, \cdot_A)$ *commutatif.* M *est un module sur l'anneau* A *si :*

- M *est muni d'une loi interne* $+_M$ *telle que* $(M, +_M)$ *soit un groupe commutatif,*

– *M est muni d'une loi externe $\cdot$ de $A \times M \to M$ vérifiant*

$$\forall \alpha, \beta \in A,\ x \in M, \quad (\alpha +_A \beta) \cdot x = \alpha \cdot x +_M \beta \cdot x \tag{6.5}$$
$$\forall \alpha \in A,\ x, y \in M, \quad \alpha \cdot (x +_M y) = \alpha \cdot x +_M \alpha \cdot y \tag{6.6}$$
$$\forall \alpha, \beta \in A,\ x \in M, \quad \alpha \cdot (\beta \cdot x) = (\alpha \cdot_A \beta) \cdot x \tag{6.7}$$
$$\forall x \in M, \quad 1 \cdot x = x \tag{6.8}$$

Dans cette définition, les indices servent à bien différencier les lois des différentes structures : la loi $+_A$ de l'anneau A n'est pas la même que la loi $+_M$ du module, mais par la suite on s'abstrait de ces précisions pour plus de lisibilité (tant qu'il n'y a pas d'ambiguïté).

Cette structure de module permet d'utiliser tous les outils mathématiques qui y sont liés. Notamment, l'équivalent dans les modules des bases :

Définition 61 (Base finie d'un module)**.** *Soit M un A-module. Une famille finie $\{v_1, \ldots, v_n\}$ est une* base *de M si :*

- *cette famille est* génératrice*, c'est-à-dire que tout élément peut s'écrire comme une combinaison linéaire des v_i,*
- *cette famille est* libre*, c'est-à-dire que $\sum_i \alpha_i v_i = 0$ implique que tous les α_i sont nuls.*

La définition de la base est la même que pour les espaces vectoriels, et est équivalente à dire que l'écriture dans la base existe et est unique. De plus, si on a une base de cardinal n, M est isomorphe à A^n (toutes les bases ont donc le même cardinal). Par contre, le caractère libre d'une famille n'est plus équivalent au fait de n'avoir aucun élément qui puisse s'exprimer en fonction des autres.

On peut noter, parmi les résultats importants sur les modules, que le théorème de la base incomplète n'est plus valable : il faut en général chercher une autre méthode pour construire une base, si celle-ci existe car ce n'est pas toujours le cas. De plus, ce n'est pas parce qu'on possède une base d'un module qu'un sous-module (partie stable par combinaison linéaire) possède également une base, et si on en a une, elle n'est pas forcément de cardinal inférieur.

Ici, l'ensemble des 0-chaînes possède la famille $(v)_{v \in V}$ comme base, et l'ensemble des 1-chaînes la famille $(e)_{e \in E}$.

6.2.2.2 Cycles

On définit à présent un 1-cycle, qui est une généralisation de ce qu'on appellerait un cycle naturellement.

Définition 62 (Bord d'une chaîne). *Soit $s = \sum_i \alpha_i e_i$ une 1-chaîne. On appelle* bord *de s et on note $\delta(s)$ la 0-chaîne :*

$$\delta(s) = \sum_i \alpha_i \delta(e_i)$$

où $\delta(u \rightarrowtail v) = v - u$.

Définition 63 (1-cycle). *Une 1-chaîne s est appelée* 1-cycle *si $\delta(s) = 0$.*

D'un point de vue mathématique, l'opérateur bord est un morphisme de modules, et l'ensemble des 1-cycles est son noyau. C'est donc un sous-module de l'ensemble des 1-chaînes.

Par exemple, si on se place sur le graphe de la figure 6.3, $s_1 = 3\,(a \rightarrowtail b) + 3\,(b \rightarrowtail c) - 4\,(c \rightarrowtail a)$ est un 1-cycle dans $\mathbb{Z}/7\mathbb{Z}$: $\delta(s_1) = 3(b-a)+3(c-b)-4(a-c) = b(3-3)+a(-3-4)+c(3+4) = 0$. Par contre $s_2 = 3\,(a \rightarrowtail b) + 3\,(b \rightarrowtail c) + 2\,(c \rightarrowtail a)$ n'en est pas un : $\delta(s_2) = -a + c \neq 0$.

Il faut ensuite faire le lien avec ce qu'on appellerait intuitivement un cycle :

Définition 64 (Cycle). *On appelle cycle une suite de sommets $u_1, u_2 \ldots u_k, u_1$ (non nécessairement distincts) tels que $u_i \rightarrowtail u_{i+1} \in E \cup \bar{E}$ pour $\forall 1 \leqslant i \leqslant k-1$ et $u_k \rightarrowtail u_1 \in E \cup \bar{E}$.*

Sa 1-chaîne associée est $(u_1 \rightarrowtail u_2) + \ldots + (u_{k-1} \rightarrowtail u_k) + (u_k \rightarrowtail u_1)$.

Propriété 36 (1-cycles et cycles). *Une 1-chaîne est un 1-cycle si et seulement si elle peut s'écrire comme une combinaison linéaire de 1-chaînes associées à des cycles.*

Démonstration. On peut noter que si $u_1, \ldots, u_k, u_1$ est un cycle, alors sa 1-chaîne associée s vérifie $\delta(s) = u_2 - u_1 + u_3 - u_2 + \ldots + u_k - u_{k-1} + u_1 - u_k = 0$. Une combinaison linéaire de 1-chaînes associées à des cycles vérifie donc également cette propriété, la difficulté est bien entendu de montrer la réciproque.

On va montrer ce résultat par récurrence sur la longueur de la chaîne, qu'on définit comme le nombre de termes de la somme lorsqu'on l'écrit de façon « réduite » :

Définition 65. *Soit $s = \sum_i \alpha_i e_i$ avec $e_i \neq \pm e_j$ pour $i \neq j$. La longueur de la chaîne s est le nombre de α_i non-nuls (dans $\mathbb{Z}/r\mathbb{Z}$).*

Si s est de longueur 0, alors le résultat est évident : s est la 1-chaîne triviale, et s'écrit bien comme une combinaison linéaire de 1-chaînes associées à des cycles (une combinaison linéaire triviale en l'occurence).

Si la chaîne est de longueur n, soit $u_1 \rightarrowtail u_2$ une arête qui apparaît dans s, avec un coefficient α_1. On cherche une arête de la forme $u_2 \rightarrowtail v$ dans s. Il en existe

forcément puisque $\delta(s) = 0$, donc comme $\alpha_1 (u_1 \rightarrowtail u_2)$ fait apparaître $\alpha_1 u_2$ dans $\delta(s)$, il existe une ou plusieurs arêtes de la forme $u_2 \rightarrowtail v'$ ou $v' \rightarrowtail u_2$, de façon à annuler le terme $\alpha_1 u_2$. Soit $u_2 \rightarrowtail u_3$ une telle arête, et α_2 son coefficient (quitte à prendre $-\alpha_2$ si il s'agit de l'arête $u_3 \rightarrowtail u_2$).

On cherche ensuite une arête $u_3 \rightarrowtail u_4$ (avec $u_4 \neq u_2$, qui existe puisque la somme fait apparaître un terme en $\alpha_3 u_3$ qui doit être annulé par un autre terme en u_3) et son coefficient α_3, et on continue le processus, jusqu'à trouver une arête $u_i \rightarrowtail u_{i+1}$ (et son coefficient α_i) tel que u_{i+1} soit égal à un des sommets précédemment trouvés, u_k. Un tel i existe forcément puisqu'il y a un nombre fini de termes dans la somme s : on finit forcément par retomber sur un sommet déjà trouvé.

On a alors le cycle $u_k, u_{k+1}, \ldots u_i, u_k$, non-trivial, composé d'arêtes qui apparaissent dans s. On écrit la 1-chaîne associée au cycle, multipliée par α_k :

$$c = \alpha_k (u_k \rightarrowtail u_{k+1}) + \alpha_k (u_{k+1} \rightarrowtail u_{k+2}) + \ldots + \alpha_k (u_i \rightarrowtail u_k)$$

On écrit alors $t = s - c$.

On constate alors que t est de longueur au plus $n - 1$. En effet, le terme $\alpha_k (u_k \rightarrowtail u_{k+1})$ a disparu, les autres termes correspondant aux arêtes du chemin existaient déjà dans s, donc au mieux sont annulés, au pire sont toujours là. On a également

$$\delta(t) = \delta(s) - \delta(c) = 0$$

Ainsi, on peut appliquer l'hypothèse de récurrence, et t s'écrit comme une combinaison linéaire de 1-chaînes associées à des cycles, donc s également.

■

L'intérêt de travailler avec des 1-cycles plutôt qu'avec les cycles est d'avoir une structure de module (de sous-module des 1-chaînes en l'occurence), qu'on n'a pas avec les cycles normaux. En fait, on verra plus bas qu'on peut construire une base des 1-cycles.

6.2.2.3 Étiquetage et cocycle

On a défini précisément ce qu'est un cycle et un 1-cycle. Or, on rappelle qu'on cherche une condition à vérifier pour un certain étiquetage des arêtes, il faut donc définir un étiquetage :

Définition 66 (Étiquetage)**.** *Soit $\mathscr{G} = (V, E)$ un graphe orienté. Un étiquetage des arêtes est la donnée de $\lambda : E \to \mathbb{Z}/r\mathbb{Z}$. On étend λ aux arêtes de $\bar{E}$ en définissant $\lambda(v \rightarrowtail u) = -\lambda(u \rightarrowtail v)$ si $u \rightarrowtail v \in E$.*

De même, on étend λ aux 1-chaînes : si $s = \sum_i \alpha_i e_i$, on écrira $\lambda(s) = \sum_i \alpha_i \lambda(e_i)$.

La figure 6.4 montre un graphe étiqueté orienté, avec les étiquettes dans $\mathbb{Z}/7\mathbb{Z}$.

On peut vérifier assez rapidement que l'ensemble des étiquetages forme un $\mathbb{Z}/r\mathbb{Z}$-*module*. On peut même remarquer qu'il s'agit de l'ensemble des applications linéaires du module des 1-chaînes vers l'anneau $\mathbb{Z}/r\mathbb{Z}$: c'est le dual algébrique du module des 1-chaînes.

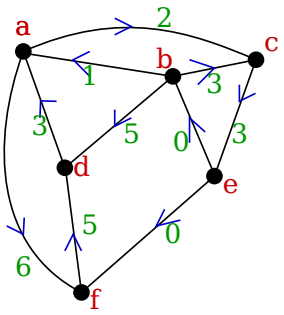


FIGURE 6.4 – Un graphe orienté avec des étiquettes dans le groupe additif $\mathbb{Z}/7\mathbb{Z}$. On peut constater que l'étiquetage ici est choisi de façon à ce que la somme sur chaque cycle soit nulle : cet étiquetage est un cocycle.

On cherche à présent des étiquetages qui vérifient que la somme sur n'importe quel cycle est nulle. Or un étiquetage qui vérifie cette propriété la vérifie également sur les 1-cycles, qui sont des combinaisons linéaires de cycles (voir la propriété 36). Les 1-cycles sont plus aisés à manipuler que les cycles, on définit donc un *cocycle*, qui est un étiquetage qui vérifiant cette condition :

Définition 67 (Cocycle)**.** *Soit $\mathscr{G} = (V, E)$ un graphe orienté. Un* cocycle *de $\mathscr{G}$ est un étiquetage λ qui vérifie, pour tout 1-cycle s,*

$$\lambda(s) = 0$$

Il existe des cocycles très simples, par exemple l'étiquetage nul est un cocycle. De même, si on choisit un sommet v quelconque, qu'on choisit un élément $a \in \mathbb{Z}/r\mathbb{Z}$, non nul, on peut choisir $\lambda(u \multimap v) = a \ \forall u \multimap v \in E \cup \bar{E}$, (c'est-à-dire a sur toutes les arêtes *partant* de v, donc on met $-a$ si l'arête est prise en sens inverse), et 0 sur toutes les autres arêtes : on peut alors vérifier que dans tout cycle qui passe par v, on se retrouve à additionner a et $-a$: la somme est donc nulle. On peut voir la figure 6.5 pour ce cas particulier, ainsi que la figure 6.4 pour un cocycle plus général.

On peut vérifier rapidement que, l'ensemble des cocycles forme un *sous-module* du module des étiquetages (on peut additionner et multiplier par des éléments de $\mathbb{Z}/r\mathbb{Z}$, et le résultat reste un cocycle). Si on trouve une base de ce sous-module (rappelons que trouver une base dans un module est généralement plus difficile que

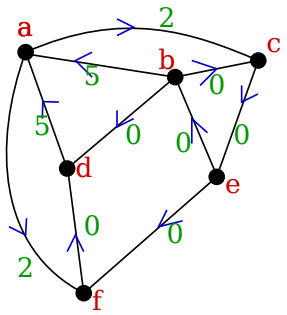


FIGURE 6.5 – Un étiquetage simple dans $\mathbb{Z}/7\mathbb{Z}$. On a choisi un sommet, et étiqueté les arêtes adjacentes à ce sommet par 2 ou $5 = -2 \mod 7$ selon l'orientation des arêtes : la somme sur tous les cycles est bien nulle, on a donc un cocycle.

dans un espace vectoriel), on saura exactement combien d'étiquetages conviennent, et comment les construire. C'est ce qui est traité dans les sections suivantes.

6.2.2.4 Base des 1-cycles

Pour construire la base des cocycles, on a besoin de construire une base des 1-cycles. Non seulement cela permet de vérifier plus simplement qu'un élément donné est un cocycle (en testant les éléments de la base et non tous les 1-cycles possibles), mais la base des cocycles se construit de manière assez similaire à celle des 1-cycles.

La construction est basée sur l'arbre couvrant maximal.

Définition 68 (Arbre couvrant maximal d'un graphe orienté)**.** *Soit $\mathscr{G} = (V, E)$. Un arbre couvrant maximal de $\mathscr{G}$ est un sous-graphe $\mathscr{A} = (V, E')$ tel que :*
- *Il n'existe pas de 1-cycle non trivial dans $\mathscr{A}$ (on dit que $\mathscr{A}$ est un arbre),*
- *pour toute arête $e \in E \backslash E'$, $(V, E' \cup \{e\})$ contient un 1-cycle non-trivial.*

On peut noter que, pour un graphe donné, il n'y a pas qu'un seul arbre couvrant maximal. Un exemple de graphe et d'un arbre couvrant maximal pour ce graphe sont montrés sur la figure 6.6.

On souhaite caractériser l'arbre couvrant maximal d'une autre manière, et pour cela, il faut définir une notion de connexité :

Définition 69 (Connexité)**.** *Soit $\mathscr{G} = (V, E)$ un graphe, et $u, v \in V$. On dit que u est relié à v dans $\mathscr{G}$, et on note $u \bullet\!\!-\!\!\bullet v$ s'il existe dans $\mathscr{G}$ une 1-chaîne s telle que $\delta(s) = v - u$.*

On dit que $\mathscr{G} = (V, E)$ est connexe *si pour tous $u, v \in V$, $u \bullet\!\!-\!\!\bullet v$.*

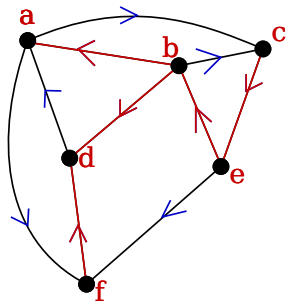


FIGURE 6.6 – Un graphe orienté (en transparence), avec un arbre couvrant maximal. On constate que tous les sommets sont reliés à (au moins) une arête de l'arbre, ainsi l'arbre « couvre » tous les sommets du graphe, d'où son nom. On peut voir aussi qu'il n'y a pas de cycle, et si on ajoute n'importe quelle arête, on crée un cycle.

La relation $\cdot\!\sim\!\cdot$ est une *relation d'équivalence* : elle est symétrique, puisque si s relie u à v, $-s$ relie v à u. Elle est également réflexive, puisque la chaîne triviale relie u à u. De plus, pour la transitivité, si s relie u à v et t relie v à w, $s+t$ relie u à w.

On peut également montrer que $\cdot\!\sim\!\cdot$ peut être caractérisée autrement :

Propriété 37. *$u \sim v$ si et seulement si il existe une 1-chaîne s qui vérifie $\delta(s) = \alpha v - \alpha u$ avec $\alpha \neq 0$.*

Il n'est pas évident que cette caractérisation soit équivalente à la première, car $\mathbb{Z}/r\mathbb{Z}$ est un anneau et pas (forcément) un corps, donc on ne peut pas forcément « diviser » une 1-chaîne par α.

Démonstration. Tout d'abord, si $u \rightarrowtail v \in E$, alors le résultat est évident : la 1-chaîne $(u \rightarrowtail v)$ relie bien u et v.

Sinon, on considère $\tilde{\mathscr{G}} = (V, \tilde{E})$, où $\tilde{E} = E \cup \{u \rightarrowtail v\}$. Dans $\tilde{\mathscr{G}}$, on peut écrire la 1-chaîne $\tilde{s} = s - \alpha\,(u \rightarrowtail v)$. On a $\delta(\tilde{s}) = \alpha v - \alpha u - \alpha v + \alpha u = 0$. C'est donc un 1-cycle, et on peut l'écrire comme une combinaison linéaire de 1-chaînes associées à des cycles. Au moins un de ces cycles contient l'arête $u \rightarrowtail v$, on peut l'écrire $u, u_2, \ldots u_k, v, u$. Quitte à écrire un cycle plus court, on peut supposer que v n'apparaît pas dans les u_i. Alors chacune des arêtes $u \rightarrowtail u_2, u_2 \rightarrowtail u_3, \ldots, u_k \rightarrowtail v$ est dans E, on peut donc écrire dans $\mathscr{G}$ (sans l'arête ajoutée) la 1-chaîne $s' = (u \rightarrowtail u_2) + \ldots + (u_k \rightarrowtail v)$, elle vérifie bien $\delta(s') = v - u$.

■

Par la suite, sauf indication contraire, on supposera que $\mathscr{G}$ est connexe, et on

utilisera l'une ou l'autres des définitions de $\bullet\!\!-\!\!\bullet$. On possède à présent l'outil pour donner une autre caractérisation de l'arbre couvrant maximal :

Propriété 38. *Soit $\mathscr{G} = (V, E)$ un graphe orienté connexe, et $\mathscr{A} = (V, E')$ un sous-graphe de $\mathscr{G}$. $\mathscr{A}$ est un arbre couvrant maximal si et seulement si, pour tous $u, v \in V$, il existe une* unique *1-chaîne s dans $\mathscr{A}$ qui vérifie $\delta(s) = v - u$.*

Cette propriété montre que $\mathscr{A}$ est connexe, et que les « chemins » qu'on y emprunte pour aller d'un sommet à un autre sont uniques.

Démonstration. Soit $\mathscr{A}$ un arbre couvrant maximal, et soient u, v deux sommets de $\mathscr{A}$. S'il existe deux 1-chaînes différentes s et s' qui vérifient $\delta(s) = \delta(s') = v - u$, alors la chaîne $s - s'$ est un 1-cycle, non trivial : ce n'est pas possible. Il faut encore montrer qu'il existe bien un tel chemin. Si ce n'est pas le cas, alors $\mathscr{A}$ est non-connexe. On appelle alors $U = \{u', \exists s, \delta(s) = u' - u\}$ et $V = \{v', \exists s, \delta(s) = v' - v\}$ ces deux composantes connexes de $\mathscr{A}$ (il y en a potentiellement d'autres). Comme la relation est transitive, il n'existe donc pas non plus de 1-chaîne qui relie les u' aux v' dans $\mathscr{A}$.

Or $\mathscr{G}$ est connexe, il existe donc au moins dans $\mathscr{G}$ une arête $u' \bullet\!\!-\!\!\bullet v'$ avec $u' \in U, v' \in V$. En ajoutant cette arête, $\mathscr{A}$ ne possède toujours pas de 1-cycle non trivial : s'il en existait un c, alors il contient $u' \bullet\!\!-\!\!\bullet v'$ (avec un coefficient α), sinon ce 1-cycle pourrait exister sans cette arête. On aurait donc $c - \alpha\,(u' \bullet\!\!-\!\!\bullet v')$ qui est dans $\mathscr{A}$, et permet de relier u' à v', ce qui n'est pas possible. Comme on a pu ajouter à $\mathscr{A}$ une arête de façon à ce qu'il n'y ait toujours pas de 1-cycle non trivial, c'est que $\mathscr{A}$ n'était pas l'arbre couvrant maximal.

Voyons maintenant le sens inverse : soit $\mathscr{A}$ un sous-graphe vérifiant cette propriété. Alors $\mathscr{A}$ ne contient pas de 1-cycle non trivial : s'il en existait un, c, contenant une arete $u \bullet\!\!-\!\!\bullet v$ avec un coefficient α, alors $\delta(c - \alpha\,(u \bullet\!\!-\!\!\bullet v)) = \alpha u - \alpha v$: $-c + \alpha\,(u \bullet\!\!-\!\!\bullet v)$ est une 1-chaîne qui est différente de $\alpha\,(u \bullet\!\!-\!\!\bullet v)$, et qui relie u à v : contradiction. Soit maintenant $\mathscr{A}$ muni d'une nouvelle arête $e \in E \backslash E', e = u \bullet\!\!-\!\!\bullet v$. Or il existe, dans $\mathscr{A}$, une 1-chaîne s qui vérifie $\delta(s) = u - v$. On a donc $s' = s + (u \bullet\!\!-\!\!\bullet v)$ qui est un 1-cycle, et non trivial car il est composé d'au moins une arête : ce n'est pas possible non plus.

■

On peut alors construire une base des 1-cycles de la façon suivante : si on ajoute une seule arête à l'arbre couvrant maximal, il contient un 1-cycle : ce 1-cycle devient un élément de la base, et on peut faire de même pour chaque arête hors de l'arbre couvrant maximal.

Définition 70 (Base des 1-cycles)**.** *Soit $\mathscr{G} = (V, E)$ un graphe orienté, et $\mathscr{A} = (V, E')$ un arbre couvrant maximal de $\mathscr{G}$. Pour tout $e \in E \backslash E'$, on définit c_e, tel*

que c_e soit l'unique 1-cycle de $(V, E' \cup \{e\})$ contenant l'arête e avec un coefficient 1 devant e.

L'existence de ce 1-cycle vient du fait que si $e = u \rightarrowtail v$, il existe dans $\mathscr{A}$ une 1-chaîne s qui vérifie $\delta(s) = u - v$, en ajoutant $(u \rightarrowtail v)$ à s, on a bien le 1-cycle cherché.

Il faut aussi vérifier que ce 1-cycle est bien unique : soient deux 1-cycles c et c' qui contiendraient tous les deux e. Alors $c = e + \sum_i \alpha_i e_i$, $c' = e + \sum_i \beta_i e'_i$, avec les e_i et e'_i dans $E' \cup \bar{E}'$. Le 1-cycle $c - c'$ est donc $\sum_i (\alpha_i e_i - \beta_i e'_i)$: c'est un 1-cycle (puisque c'est une somme de 1-cycles) dans $\mathscr{A}$, il ne peut être que trivial, d'où $c = c'$.

Propriété 39. *La famille $(c_e), e \in E \backslash E'$ forme une base des 1-cycles.*

Démonstration. Montrons d'abord que la famille (c_e) est libre. Soient des éléments α_e de $\mathbb{Z}/r\mathbb{Z}$ qui vérifient $\sum_e \alpha_e c_e = 0$. Une arête $e \in E \backslash E'$ donnée apparaît à un seul endroit : dans $\alpha_e c_e$ (puisque que chaque c_e fait apparaître des arêtes de E' et e, mais pas d'autres arêtes de $E \backslash E'$). Ainsi, pour obtenir 0 dans la somme, il faut imposer $\alpha_e = 0$ pour toute arête $e \in E \backslash E'$.

Ensuite, montrons qu'elle est génératrice, c'est-à-dire que tout 1-cycle s peut s'écrire $\sum_e \alpha_e c_e$, avec $\alpha_e \in \mathbb{Z}/r\mathbb{Z}$. Si on se donne s, on peut trouver les α_e selon si l'arête e apparaît dans s, et avec quel coefficient, (si elle apparaît inversée, on prend l'opposé du coefficient), ou si elle n'apparaît pas ($\alpha_e = 0$). On écrit donc $s' = s - \sum_e \alpha_e c_e$, et on cherche à montrer que cette somme est triviale (ainsi on aura bien $s = \sum_e \alpha_e c_e$).

Pour cela, on prend une arête $e \in E \backslash E'$. Du fait du choix de α_e, e n'apparaît plus dans s'. Comme c'est vrai pour tout $e \in E \backslash E'$, s' n'a pour support que des arêtes de $\mathscr{A}$. Or s' est toujours un 1-cycle (combinaison linéaire de 1-cycles) : il est nécessairement trivial, ce qu'on voulait montrer.

■

6.2.2.5 Base des cocycles

L'arbre couvrant maximal $\mathscr{A}$ sert aussi à définir une base des cocycles.

Définition 71 (base des cocycles)**.** *Soit $\mathscr{G} = (V, E)$ un graphe orienté, et $\mathscr{A} = (V, E')$ un arbre couvrant maximal de $\mathscr{G}$. On définit, pour toute arête $e \in E'$, l'étiquetage λ_e de la façon suivante. Pour $e = u \rightarrowtail v$, on définit $V_u = \{u' \in V, u' \bullet\!\!-\!\!\bullet u$ dans $\mathscr{A}\}$ et $V_v = V \backslash V_u$.*

On définit alors λ_e tel que :

– *$\lambda_e(u_1, u_2) = 0$ si $u_1, u_2 \in V_u$,*

- $\lambda_e(v_1, v_2) = 0$ *si* $v_1, v_2 \in V_v$,
- $\lambda_e(u_1, v_1) = 1$ *si* $u_1 \in V_u$ *et* $v_1 \in V_v$.

Visuellement, V_u représente tous les sommets qui sont du « côté » de u dans l'arbre couvrant. On peut voir assez rapidement que, comme tous les sommets sont reliés par une 1-chaîne dans $\mathscr{A}$, il y a ceux qui sont du « côté » de u ; et ceux qui ne le sont pas sont forcément du « côté » de v, ainsi on peut écrire $V_v = \{v' \in V, v' \bullet\!\!\cdots\!\!\bullet v \text{ dans } \mathscr{A}\}$. On peut voir sur la figure 6.7 un tel cocycle.

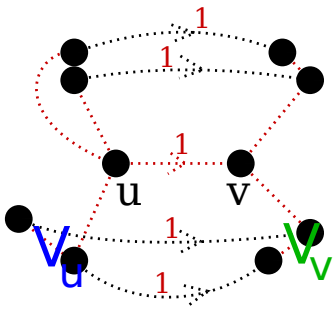


FIGURE 6.7 – Un exemple de construction de cocycle à partir de l'arbre couvrant maximal : pour une arête donnée $u \bullet\!\!\rightarrow\!\!\bullet v$, on sépare les sommets en deux lots, ceux du côté de u et ceux du côté de v. On étiquette alors 1 sur chaque arête entre les deux lots (si les arêtes sont orientées dans le sens inverse, on met -1), et 0 sur les arêtes intérieures à chaque lot.

On a alors la propriété espérée :

Propriété 40. *La famille des cocycles* (λ_e), $e \in E'$ *définie ci-dessus est une base des cocycles.*

Démonstration. Il faut d'abord montrer que les λ_e sont bien des cocycles, donc que pour tout 1-cycle c, on a $\lambda_e(c) = 0$. Puisqu'on a construit une base des 1-cycles, il suffit de vérifier cette propriété sur ces 1-cycles particuliers.

Soit donc $e' = u' \bullet\!\!\rightarrow\!\!\bullet v' \in E', e = u \bullet\!\!\rightarrow\!\!\bullet v \in E \backslash E'$. c_e s'écrit donc $(e) + s$, avec s ayant pour support les arêtes de $\mathscr{A}$ uniquement.

- Soit $\lambda_{e'}(e) = 1$. Cela signifie donc que $u \in V_u$ et $v \in V_v$. Donc u et u' sont dans V_u : il existe une 1-chaîne s_u, dans $\mathscr{A}$, qui vérifie $\delta(s_u) = u - u'$. De même pour v et v', on a s_v telle que $\delta(s_v) = v - v'$. On a donc $s = s_u - (u' \bullet\!\!\rightarrow\!\!\bullet v') + s_v$ car cette écriture est unique dans $\mathscr{A}$ (par la propriété 38), et donc $c_e = e + s_v - (e') + s_u$.
  Or $\lambda_{e'}(s_u) = 0$ car toutes les arêtes ont des sommets dans $\mathscr{A}$. De même pour s_v. On a donc $\lambda_{e'}(c_e) = \lambda_{e'}(e) + \lambda_{e'}(v' \bullet\!\!\rightarrow\!\!\bullet u') = 1 - 1 = 0$.
- Soit $\lambda_{e'}(e) = -1$. On a un résultat symétrique.

– Soit $\lambda_{e'}(e) = 0$. On a donc u et v dans le même ensemble, disons V_u. La 1-chaîne s s'écrit donc comme une somme (unique) d'arêtes de $\mathscr{A}$ dont les sommets sont dans V_u : $\lambda_{e'}(c_e) = \lambda_{e'}(s) + \lambda_{e'}(e) = 0$.

Pour montrer que la famille est libre, c'est assez simple : chaque λ_e a un support non nul sur une arête e sur laquelle les autres $\lambda_{e'}$ sont nuls : par un raisonnement analogue à celui de la preuve de la propriété 39, on a le résultat.

Il reste à montrer que la famille est génératrice, autrement dit que tout cocycle λ s'écrit $\sum_e \alpha_e \lambda_e$. Pour un cocycle donné λ, on peut regarder la valeur de $\lambda(e)$ pour obtenir α_e. Si on regarde $\lambda' = \lambda - \sum_e \alpha_e \lambda_e$, on constate que $\lambda'(e) = 0$ pour tout $e \in E'$. Il faut encore montrer que λ' est nul sur les autres arêtes. Soit une arête $e \in E \backslash E'$. Comme λ' est un cocycle, $\lambda'(c_e) = 0$ (où c_e est un 1-cycle de base, voir définition 70). Or $c_e = s + e$ avec s ayant des arêtes uniquement dans E' : $\lambda(s) = 0$. On a alors $\lambda'(c_e) = 0 + \lambda'(e) = 0$, donc $\lambda'(e) = 0$.

On a donc $\lambda' = 0$, donc $\lambda = \sum_e \alpha_e \lambda_e$, ce qu'on voulait démontrer.

■

On a donc obtenu la base recherchée, et donc tout étiquetage vérifiant la condition de cycle s'écrit de manière unique $\sum_e \alpha_e \lambda_e$.

6.2.2.6 Construction de l'étiquetage avec une famille génératrice plus simple

Si on revient au problème de départ, la méthode pour obtenir les étiquettes du graphe est donc la suivante :

1. On construit un arbre couvrant maximal du graphe de Tanner,
2. Pour chaque arête $e \in E'$ de cet arbre, on construit λ_e,
3. On choisit autant de valeurs $\alpha_e \in \mathbb{Z}/r\mathbb{Z}$,
4. On calcule l'étiquetage $\lambda = \sum_e \alpha_e \lambda_e$
5. On considère l'étiquetage associé dans le groupe multiplicatif $\mathbb{F}_q{}^*$.

Cependant, construire une telle base peut être un peu long, et il existe des familles génératrices plus simples, dont la suivante (déjà évoquée dans les exemples plus haut, notamment dans la figure 6.5) :

Définition 72. *Soit $\mathscr{G} = (V, E)$ un graphe orienté. On définit la famille d'étiquetages $(\mu_v)_{v \in V}$ de la façon suivante :*
Si $v \rightarrowtail u \in E \cup \bar{E}$, $\mu_e(v \rightarrowtail u) = 1$, et $\mu_e(e') = 0$ ailleurs.

Si on montre que cette famille est génératrice, alors l'algorithme devient plus simple :

1. Pour chaque sommet v du graphe, on tire α_v un élément de $\mathbb{F}_q{}^*$,
2. Pour chaque arête $u \bullet\!\!-\!\!\bullet v$, on associe l'étiquette $\alpha_u \alpha_v^{-1}$.

Notamment, il n'y a pas besoin de calculer d'arbre couvrant maximal. Encore mieux, comme dans le cas des graphes de Tanner on a affaire à un graphe biparti dont on a choisi l'orientation des arêtes dans le sens nœud de variable vers nœud de parité, chaque arête $v \bullet\!\!-\!\!\bullet c$ aura pour étiquette $\alpha_v \beta_c^{-1}$: or, puisque les β_c sont tirés au hasard dans $\mathbb{F}_q{}^*$, il n'est même pas nécessaire de les inverser. Ainsi on affectera à $v \bullet\!\!-\!\!\bullet c$ l'étiquette $\alpha_v \beta_c$ directement.

Quel est alors l'intérêt de la base construite plus haut ? Précisément à montrer qu'une telle famille est génératrice.

Propriété 41. *La famille (μ_v) est une famille génératrice des cocycles.*

Démonstration. On va montrer que tout cocycle de la base des λ_e s'écrit comme une combinaison linéaire des μ_v, et on a besoin de l'arbre couvrant maximal pour cela. Soit $\mathscr{A} = (V, E')$ un arbre couvrant maximal, et soit λ_e, avec $e = u \bullet\!\!-\!\!\bullet v$ un cocycle de la base associée. On va construire une combinaison linéaire de μ_v qui est égale à λ_e sur les arêtes de E'. Par le même raisonnement que dans la preuve de la propriété 40, on aura l'égalité sur les autres arêtes. On se place donc sur l'arbre $\mathscr{A}$, au niveau de l'arete $u \bullet\!\!-\!\!\bullet v$. On définit alors $\mu(u \bullet\!\!-\!\!\bullet v)$ récursivement de la façon suivante :

- Si u est une feuille (c'est-à-dire si $u \bullet\!\!-\!\!\bullet v$) est l'unique arête faisant intervenir v dans $\mathscr{A}$), alors on définit $\mu = \mu_u$
- Sinon, pour toutes les arêtes de la forme $e' = u' \bullet\!\!-\!\!\bullet u$ ($u' \neq v$), on construit récursivement $\mu^{e'}$ tel que $\mu^{e'} = \lambda_{e'}$. On écrit ensuite $\mu = \mu_u + \sum_{e'} \mu^{e'}$.

On peut vérifier rapidement que cet algorithme termine, au pire en un nombre d'étapes égal au nombre de sommets du graphe. On peut vérifier également que μ ainsi obtenu est bien une combinaison linéaire des μ_v. Il n'y a plus qu'à vérifier que $\mu = \lambda_e$. Lorsque u est une feuille, l'ensemble V_u des sommets du côté de u dans $\mathscr{A}$ est restreint à $\{u\}$. On a donc une seule arête de $\mathscr{A}$ qui soit de V_u vers V_v, c'est $e = u \bullet\!\!-\!\!\bullet v$. Donc $\lambda_e(e') = 0$ partout ailleurs que sur l'arête e, et $\lambda_e(e) = 1$. Or $\mu_u(u \bullet\!\!-\!\!\bullet v) = 1$, et comme il n'y a pas d'autre arête de la forme $u \bullet\!\!-\!\!\bullet v'$, $\mu_u(e') = 0$ ailleurs. On a donc bien $\lambda_{u \bullet\!\!-\!\!\bullet v} = \mu_v$.

Sinon, on suppose par hypothèse de récurrence qu'on a su construire $\mu^{e'}$ pour toute arête de la forme $e' = u' \bullet\!\!-\!\!\bullet u$ vérifiant $\mu^{e'} = \lambda_{e'}$. Avec $\mu = \mu_u + \sum_{e'} \lambda_{e'}$, on constate que $\mu(e) = \mu_u(e) = 1$. Pour une arête de la forme $u' \bullet\!\!-\!\!\bullet u$, on a $\mu(u' \bullet\!\!-\!\!\bullet u) = \mu_u(u' \bullet\!\!-\!\!\bullet u) + \lambda_{u' \bullet\!\!-\!\!\bullet u}(u' \bullet\!\!-\!\!\bullet u) = -1 + 1 = 0$. Pour les autres arêtes de $\mathscr{A}$, μ y est clairement nul, car elles n'apparaissent pas dans le support des $\lambda_{e'}$ ou de μ_u.

■

Ainsi, on dispose d'une famille génératrice simple à construire, et ainsi on peut facilement étiqueter le graphe de façon à obtenir le produit sur les cycles égal à 1. Cette construction était celle présentée dans [AMT12a], jusqu'ici il n'était pas prouvé qu'elle générait bien tous les étiquetages possibles.

6.2.3 Application : le code torique étendu

Le code torique de paramètre n est présenté dans la section 4.1.3, page 65. Il s'agit d'un code CSS, dont le graphe de Tanner est celui de la figure 6.8. La matrice de parité en X (ainsi que celle en Z) possède exactement deux 1 par colonne, et quatre 1 par ligne. Il est de dimension 2 (la somme de toutes les lignes de chaque matrice étant égale à 0), de longueur $2n^2$ et de distance minimale n.

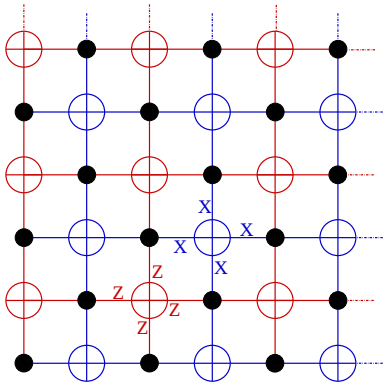


FIGURE 6.8 – Graphe de Tanner du code torique, présenté en section 4.1.3. Si on considère uniquement la partie en X (bleue), ou la partie en Z, on a bien un degré des nœuds variable (nombre de 1 par colonne) de 2, et un degré des nœuds de parité (nombre de 1 par ligne) de 4.

Définition 73 (Code torique q-aire, code torique étendu)**.** *On appelle* code torique q-aire *de paramètre n la version q-aire du code torique de paramètre n, obtenue par la méthode de la section 6.2.1.*
On appelle code torique étendu *de paramètres n et m le code binaire obtenu à partir du code torique q-aire, avec $q = 2^m$, par la méthode décrite dans la section 6.1.2.*

Dans toute cette section, on notera $\mathscr{C}$ le code torique, $\mathscr{C}_X$ et $\mathscr{C}_Z$ les parties en X et en Z, et $\mathbf{H}_X$ et $\mathbf{H}_Z$ leurs matrices de parité respectives, et $\mathscr{G}$, $\mathscr{G}_X$ et $\mathscr{G}_Z$ le graphe de Tanner de $\mathscr{C}$, $\mathscr{C}_X$ et $\mathscr{C}_Z$ respectivement. On notera avec un ^{q} toutes les versions q-aires : $\mathscr{C}^q$, $\mathscr{C}_X^q$, $\mathbf{H}_X^q$, $\mathscr{G}_X^q$, etc ; et avec un $'$ pour les version binaires étendues : $\mathscr{C}'$, $\mathscr{C}_X^{q\,\prime}$, etc.

Le code torique étendu de paramètres n et m a donc pour longueur $2mn^2$. On a en outre deux résultats intéressants, concernant la dimension et la distance minimale :

Théorème 6 (Dimension du code torique étendu)**.** *Le code torique étendu de paramètres n et m a pour dimension $2m$.*

Théorème 7 (Distance minimale du code torique étendu)**.** *Le code torique étendu de paramètres n et m a une distance minmale comprise entre n et mn.*

Pour cela, on va montrer les résultats suivants, sur le code q-aire :

Propriété 42. *Le code torique q-aire de paramètre n a pour dimension 2, où la dimension est définie par :*

$$2n^2 - \mathrm{rang}(\mathbf{H}_X^q) - \mathrm{rang}(\mathbf{H}_Z^q)$$

Propriété 43. *Le code torique q-aire de paramètre n a pour distance minimale n, où la distance minimale d est définie par :*

$$d = \min\{d_X, d_Z\}$$

$$d_X = \min\{\mathrm{w_q}(x), x \in {\mathbb{F}_q}^{2n^2}, \mathbf{H}_Z^q\ {}^{\mathbf{t}}(x) = 0, x \notin \mathrm{VectLi}(\mathbf{H}_X^q)\}$$

$$d_Z = \min\{\mathrm{w_q}(x), x \in {\mathbb{F}_q}^{2n^2}, \mathbf{H}_X^q\ {}^{\mathbf{t}}(x) = 0, x \notin \mathrm{VectLi}(\mathbf{H}_Z^q)\}$$

où $\mathrm{w_q}(x)$ *est le poids q-aire d'un mot, autrement dit :*

$$\mathrm{w_q}(x_1 x_2 \dots x_k) = \#\{1 \leqslant i \leqslant k, x_i \neq 0 \in \mathbb{F}_q\}$$

et $\mathrm{VectLi}(A)$ *désigne ici l'espace vectoriel sur $\mathbb{F}_q$ engendré par les lignes de la matrice A.*

Il est nécessaire de redéfinir la notion de dimension et de distance minimale, car jusqu'ici on n'a pas parlé de codes quantiques q-aires en tant que tels, et ces notions n'ont donc pas forcément de sens dans l'absolu. Cependant, elles sont très utiles pour comprendre le code torique étendu, et pour cela, on les définit de façon analogue au cas binaires.

Une fois qu'on a ces propriétés 42 et 43, la preuve du théorème 6 est évidente : comme à un élément de $\mathbb{F}_q$ est associé m bits, une dimension 2 q-aire correspond à une dimension $2m$ binaire. Pour le théorème 7, on sait qu'à un élément non-nul de $\mathbb{F}_q$ est associé m bits, dont au moins un n'est pas nul : ainsi, un mot de poids n sur $\mathbb{F}_q$ correspond à un mot binaire de poids minimum n et de poids maximum mn.

On peut noter que ces propriétés ne sont pas a priori évidentes : pour la dimension, comme il y a exactement n^2 lignes dans chaque matrice, il faut faire en sorte qu'elles soient liées (sinon on aurait une dimension de $2n^2 - \text{rang}(\mathbf{H}_X^q) - \text{rang}(\mathbf{H}_Z^q) = 0$), et ce n'est pas parce qu'elles le sont dans $\mathbb{F}_2$ qu'elles le sont forcément dans $\mathbb{F}_q$. Dans le cas général, on n'a pas forcément cette préoccupation (on peut se permettre de perdre un peu en dimension), ici il faut garantir une dimension positive.

De plus, pour la distance minimale, comme on a un degré 2 au niveau des nœuds variable, les erreurs indétectées sont des cycles dans le graphe de Tanner. Dans notre cas, la taille minimale d'un cycle dans $\mathbb{F}_q$ est bien sûr minorée par celle d'un cycle dans $\mathbb{F}_2$. Or, les erreurs sérieuses sont des cycles *modulo* un ensemble de cycles : un cycle correspondant à une erreur sérieuse sur $\mathbb{F}_q$ est toujours une erreur indétectée sur $\mathbb{F}_2$, mais rien ne garantit qu'elle n'est pas dans le stabilisateur.

Preuve de la propriété 42. La démonstration se base sur deux lemmes :

Lemme 5. *Si $\mathscr{G}_X^q$ vérifie la condition de cycles, alors $\mathscr{G}_Z^q$ aussi.*

Lemme 6. *Si $\mathscr{G}_X^q$ vérifie la condition de cycles, alors $\mathscr{C}_X^q$ est de dimension n^2+1.*

À partir de ces deux lemmes, on déduit le rang q-aire des deux matrices $\mathbf{H}_X^q$ et $\mathbf{H}_Z^q$, qui est de n^2-1, et donc la dimension du code : $k = 2n^2 - \text{rang}(\mathbf{H}_X^q) - \text{rang}(\mathbf{H}_Z^q) = 2n^2 - 2(n^2-1) = 2$.

■

Preuve du lemme 5 . Considérons dans un premier temps un morceau du graphe $\mathscr{G}^q$, composé de deux nœuds variable qu'on renommera 0 et 1 pour simplifier, et de deux nœuds de parité A et B (voir figure 6.9). Si on appelle x_{A0}, x_{A1} et z_{B0}, z_{B1} les étiquettes des arêtes correspondantes[2], la condition d'orthogonalité entre A et B s'exprime alors :

$$x_{A0}z_{B0} + x_{A1}z_{B1} = 0$$

autrement dit,

$$x_{A0}{x_{A1}}^{-1}z_{B0}{z_{B1}}^{-1} = 1 \tag{6.9}$$

Cela correspond à dire que le produit sur ce petit cycle élémentaire est égal à 1, en généralisant la définition de produit sur un cycle au cas du graphe de Tanner d'un code CSS :

Définition 74 (Produit sur un chemin, cas CSS)**.** *Soit un chemin $c = c_1c_2\dots c_k$ sur le graphe de Tanner d'un code CSS q-aire. On définit le* produit sur ce chemin *$\pi(c)$ dans le graphe de Tanner par le produit des étiquettes des arêtes $c_i \leftrightarrow c_{i+1}$ le long de ce chemin, avec une puissance :*

2. les couleurs ici n'ont pas de signification propre, elles sont là uniquement pour améliorer la lisibilité des formules, en gardant la convention rouge pour Z, bleu pour X.

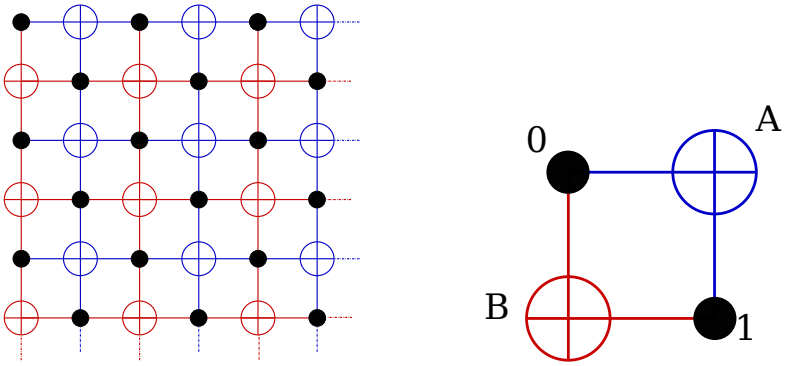


FIGURE 6.9 – Extrait du graphe de Tanner de $\mathscr{C}^q$: on considère un cycle élémentaire composé de deux nœuds variable et deux nœuds de parité.

- $+1$ *si l'arête apparaît dans le sens nœud variable vers nœud de parité sur le graphe* $\mathscr{G}_X^q$,
- -1 *si l'arête apparaît dans le sens nœud variable vers nœud de parité sur le graphe* $\mathscr{G}_Z^q$,
- -1 *si l'arête apparaît dans le sens nœud de parité vers nœud variable sur le graphe* $\mathscr{G}_X^q$
- $+1$ *si l'arête apparaît dans le sens nœud de parité vers nœud variable sur le graphe* $\mathscr{G}_Z^q$.

On parlera de produit sur un cycle *lorsque le chemin est un cycle.*

On peut voir assez rapidement que si $c = c_1 c_2 \dots c_k$, alors $\pi(c) = \pi(c_1, \dots c_i)$ $\pi(c_i, c_{i+1}, \dots c_k)$ pour $1 \leqslant i \leqslant k$ (avec le produit trivial, c'est-à-dire le produit sur un unique sommet, égal à 1). On a également $\pi(c_1, \dots, c_k) = \pi(c_k, \dots, c_1)^{-1}$.

On observe également que le graphe $\mathscr{G}_X^q$ est très similaire à celui de $\mathscr{G}_Z^q$: on peut donc associer un nœud de $\mathscr{G}_X^q$ à un nœud de $\mathscr{G}_Z^q$ par une translation :

Définition 75. *Soit (i, j) un nœud de $\mathscr{G}_Z^q$. On définit son translaté $\overrightarrow{T}(i, j) = (i + 1, j + 1)$ et son translaté partiel $\overrightarrow{T}_p(i, j) = (i + 1, j)$, où les coordonnées sont prises modulo $2n$.*
On peut alors définir le translaté (respectivement le translaté partiel) d'une arête $v \rightarrowtail c$ par l'arête $\overrightarrow{T}v \rightarrowtail \overrightarrow{T}c$ (respectivement $\overrightarrow{T}_p v \rightarrowtail \overrightarrow{T}_p c$), et le translaté (respectivement le translaté partiel) d'un chemin $c_1, \dots, c_n$ par $\overrightarrow{T}c_1, \dots, \overrightarrow{T}c_n$ (respectivement $\overrightarrow{T}_p c_1, \dots, \overrightarrow{T}_p c_n$).

Si (i, j) est un nœud variable, alors $\overrightarrow{T}(i, j)$ est aussi un nœud variable (pour des raisons de parité), et si (i, j) est un nœud de parité en Z, alors $\overrightarrow{T}(i, j)$ est un nœud de parité en X. La structure très régulière du code torique permet d'assurer

que l'arête $\overrightarrow{T}v \rightarrowtail \overrightarrow{T}c$ existe toujours si $v \rightarrowtail c$ existe, et donc que le translaté d'un chemin est bien un chemin.

On peut voir sur la figure 6.10 un exemple de chemin, avec son translaté et son translaté partiel.

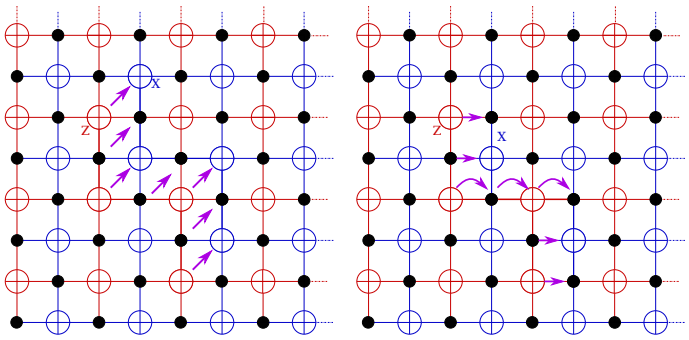


FIGURE 6.10 – Un chemin dans $\mathscr{G}_Z^q$ et son translaté (à droite, son translaté partiel) : il s'agit du même chemin, décalé en haut à droite (ou simplement à droite). On peut remarquer que le translaté d'un chemin dans $\mathscr{G}_Z^q$ est dans $\mathscr{G}_X^q$, mais ce n'est pas le cas du translaté partiel qui peut être sur $\mathscr{G}_X^q$ et $\mathscr{G}_Z^q$.

On a alors le lemme suivant :

Lemme 7. *Si $C_k = c_1, c_2, \dots c_k$ est un chemin dans $\mathscr{G}_Z^q$*[3]*, alors on a :*

$$\pi(c_1, c_2, \dots c_k)\pi(c_k, \overrightarrow{T}_p c_k, \overrightarrow{T}c_k)\pi(\overrightarrow{T}c_1, \overrightarrow{T}c_2, \dots, \overrightarrow{T}c_k)^{-1}\pi(c_1, \overrightarrow{T}_p c_1, \overrightarrow{T}c_1)^{-1} = 1$$

Ce lemme correspond à prendre le produit sur un cycle composé d'un chemin, de son translaté et des extrémités nécessaires pour « fermer » le cycle (voir figure 6.11).

Démonstration. On va montrer le résultat par récurrence sur k (la longueur du chemin).

Si $k = 1$, alors la relation cherchée s'écrit

$$\pi(C_1) = \pi(c_1)\pi(c_1, \overrightarrow{T}_p c_1, \overrightarrow{T}c_1)\pi(\overrightarrow{T}c_1)^{-1}\pi(c_1, \overrightarrow{T}_p c_1, \overrightarrow{T}c_1)^{-1} = 1$$

Or cette relation est évidente puisque si un chemin possède un seul sommet, le produit sur ce chemin est égal à 1. Supposons que la relation soit vraie pour un

3. Là encore, la couleur n'a que pour but de faciliter la lecture

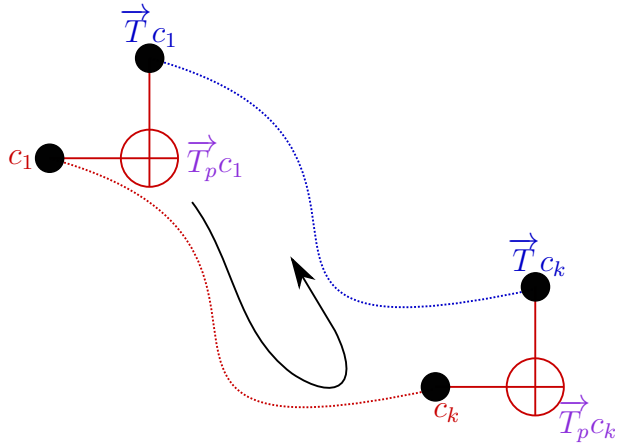


FIGURE 6.11 – Un cycle composé d'un chemin dans $\mathscr{G}_Z^q$, de son translaté, et des nœuds (obtenus par translaté partiel) qui permettent de fermer le cycle.

chemin de longueur $k-1 \geqslant 1$. On peut écrire :

$$\begin{aligned}
\pi(C_k) &= \pi(c_1,\dots,c_k)\pi(c_k,\overrightarrow{T}_p c_k,\overrightarrow{T}c_k)\pi(\overrightarrow{T}c_1,\dots\overrightarrow{T}c_k)^{-1}\pi(c_1,\overrightarrow{T}_p c_1,\overrightarrow{T}c_1)^{-1}\\
&= \pi(c_1,\dots,c_{k-1})\pi(c_{k-1},c_k)\pi(c_k,\overrightarrow{T}_p c_k,\overrightarrow{T}c_k)\pi(\overrightarrow{T}c_1,\dots,\overrightarrow{T}c_{k-1})^{-1}\\
&\quad \pi(\overrightarrow{T}c_{k-1},\overrightarrow{T}c_k)^{-1}\pi(c_1,\overrightarrow{T}_p c_1,\overrightarrow{T}c_1)^{-1}\\
&= \pi(C_{k-1})\pi(c_{k-1},\overrightarrow{T}_p c_{k-1},\overrightarrow{T}c_{k-1})^{-1}\pi(c_{k-1},c_k)\\
&\quad \pi(c_k,\overrightarrow{T}_p c_k,\overrightarrow{T}c_k)\pi(\overrightarrow{T}c_{k-1},\overrightarrow{T}c_k)^{-1}
\end{aligned}$$

Cela revient à découper le chemin de longueur k en un chemin de longueur $k-1$, puis la dernière arête. En utilisant l'hypothèse de récurrence, $\pi(C_{k-1})=1$; il reste donc à calculer

$$\begin{aligned}
p &= \pi(c_{k-1},\overrightarrow{T}_p c_{k-1},\overrightarrow{T}c_{k-1})^{-1}\pi(c_{k-1},c_k)\pi(c_k,\overrightarrow{T}_p c_k,\overrightarrow{T}c_k)\pi(\overrightarrow{T}c_{k-1},\overrightarrow{T}c_k)^{-1}\\
&= \pi(c_{k-1},c_k,\overrightarrow{T}_p c_k,\overrightarrow{T}c_k)\pi(c_{k-1},\overrightarrow{T}_p c_{k-1},\overrightarrow{T}c_{k-1},\overrightarrow{T}c_k)^{-1}\\
&= \pi(c_{k-1},c_k,\overrightarrow{T}_p c_k,\overrightarrow{T}c_k,\overrightarrow{T}c_{k-1},\overrightarrow{T}_p c_{k-1},c_{k-1}).
\end{aligned}$$

Or on rappelle que c_{k-1} de coordonnées (i',j') est relié à c_k de coordonnées (i,j). On a donc quatre possibilités : $(i',j')=(i\pm1,j)$, ou $(i',j')=(i,j\pm1)$.

- Si $(i',j')=(i-1,j)$. Alors $p=\pi((i-1,j),(i,j),(i+1,j),(i+1,j+1),(i,j+1),(i,j),(i-1,j)) = \pi((i,j),(i+1,j),(i+1,j+1),(i,j+1),(i,j))$ Or il s'agit d'un cycle élémentaire, donc le produit y est égal à 1.
- Si $(i',j')=(i+1,j)$, alors $p=\pi((i+1,j),(i,j),(i+1,j),(i+1,j+1),(i+2,j+1),(i+2,j),(i+1,j))=\pi((i+1,j),(i+1,j+1),(i+2,j+1),(i+2,j),(i+1,j))$ qui est aussi un cycle élémentaire.

- Si $(i',j') = (i,j-1)$, alors $p = \pi((i,j-1),(i,j),(i+1,j),(i+1,j+1),(i+1,j),(i+1,j-1),(i,j-1)) = \pi((i,j-1),(i,j),(i+1,j),(i+1,j-1),(i,j-1))$ qui est aussi un cycle élémentaire.
- Si enfin $(i',j') = (i,j+1)$, alors $p = \pi((i,j+1),(i,j),(i+1,j),(i+1,j+1),(i+1,j+2),(i+1,j+1),(i,j+1)) = \pi((i,j+1),(i,j),(i+1,j),(i+1,j+1),(i,j+1))$ qui est encore un cycle élémentaire.

On a donc dans tous les cas $p = 1$, donc $\pi(C_k) = 1$.

■

Considérons à présent un cycle dans le graphe de Tanner de $\mathscr{C}_Z^q$. Il est de la forme $C_Z = c_1, \ldots, c_k, c_1$. Alors, le chemin $C_X = \overrightarrow{T}c_1, \ldots, \overrightarrow{T}c_k, \overrightarrow{T}c_1$ est un cycle du graphe de $\mathscr{C}_X^q$.

On applique alors le lemme 7, et on obtient : $\pi(c_1, \ldots, c_k, c_1)\pi(c_1, \overrightarrow{T_p}c_1, \overrightarrow{T}c_1)$ $\pi(\overrightarrow{T}c_1, \ldots, \overrightarrow{T}c_k, \overrightarrow{T}c_1)^{-1}\pi(c_1, \overrightarrow{T_p}c_1, \overrightarrow{T}c_1)^{-1} = 1$, c'est-à-dire $\pi(C_Z)\pi(C_X) = 1$. Or comme $\pi(C_X) = 1$, on a bien $\pi(C_Z) = 1$, ce qu'il fallait démontrer.

■

Il est intéressant de remarquer que la preuve fonctionne grâce à deux principaux ingrédients : la relation de produit sur les cycles élémentaires, et l'opérateur de translation. La première relation peut être vérifiée pour d'autres codes de cycles, il suffit pour cela d'avoir au plus deux nœuds variable reliés à deux nœuds de parité donnés, et la condition d'orthogonalité donne la relation voulue. Pour trouver un équivalent de $\overrightarrow{T}$ sur un autre code, il faut trouver un opérateur qui transforme un cycle de $\mathscr{C}_Z^q$ en un cycle de $\mathscr{C}_X^q$, et pour lequel on puisse prouver un équivalent du lemme 7. La structure particulière du code torique (notamment le fait que les graphes $\mathscr{G}_X^q$ et $\mathscr{G}_Z^q$ soient les mêmes) est donc ici très importante, et ce lemme peut difficilement se généraliser à d'autres codes.

Preuve du lemme 6. Pour montrer que $\mathscr{C}_X^q$ ainsi obtenu est de dimension n^2+1, on va montrer dans un premier temps que le code est de dimension *au moins* n^2+1, en montrant que chaque mot de code binaire possède un équivalent q-aire de même support : ainsi, comme il existe n^2+1 mots binaires indépendants, ce sera le cas pour le code q-aire. Ensuite, il restera à montrer que la dimension est *au plus* n^2+1, en montrant que la matrice de parité possède n^2-1 lignes indépendantes.

Pour le premier point, on utilisera le lemme suivant :

Lemme 8. *Si sur $\mathscr{G}_X^q$ le produit sur les cycles est égal à 1, tout cycle sur $\mathscr{G}_X^q$ peut être le support d'un mot de code de $\mathscr{C}_X^q$.*

Ce lemme prouve immédiatement que la dimension est supérieure à n^2+1, car un mot de code binaire est forcément un cycle ou une union de cycles.

Démonstration. Soit un cycle sur $\mathscr{G}_X^q$, de longueur m, et un mot de code w ayant pour support ce cycle.

Le cycle s'écrit $v_1, c_1, v_2, c_2 \ldots, v_m, c_m, v_1$. Pour simplifier les notations, on écrira w_i pour la valeur de w en position v_i. De même, on notera les coefficients sur l'arête $v_i \multimap c_i$ x_i et les coefficients sur $v_{i+1} \multimap c_i$ x'_i. Ces notations sont résumées sur la figure 6.12.

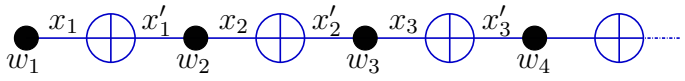


FIGURE 6.12 – Notations utilisées pour prouver le lemme 8 : on considère un cycle du graphe de Tanner, et on construit un mot de code $w_1 \ldots w_m$ ayant pour support ce cycle.

Trouver les w_i revient à résoudre le système suivant :

$$\begin{aligned} w_1x_1 + w_2x'_1 &= 0 \\ w_2x_2 + w_3x'_2 &= 0 \\ &\ldots \\ w_ix_i + w_{i+1}x'_i &= 0 \\ &\ldots \\ w_mx_m + w_0x'_m &= 0 \end{aligned}$$

Pour trouver des w_i non-nuls, il faut et il suffit que le déterminant de ce système soit égal à 0, autrement dit que :

$$\begin{aligned} & x_1x_2 \ldots x_m + x'_1x'_2 \ldots x'_m = 0 \\ \iff \quad & x_1{x'_1}^{-1}x_2{x'_2}^{-1} \ldots x_m{x'_m}^{-1} = 1 \end{aligned}$$

Ce qui est équivalent à dire que le produit sur ce cycle est égal à 1. De plus, les w_i sont bien *tous* non-nuls : si w_0 est non-nul, alors la première équation impose w_1 non-nul, la seconde w_2 non-nul, et ainsi de suite. Le support de ce mot de code est donc bien le cycle entier.

■

Contrairement au cas précédent, ce lemme est tout à fait applicable à un code de cycles quelconques : on peut donc garantir une dimension pour $\mathscr{C}_X^q$ qui est au moins celle de $\mathscr{C}_X$.

Pour montrer que la dimension est au plus n^2+1, c'est-à-dire qu'il y a au moins $n^2 - 1$ nœuds de parité indépendants, on utilise une technique assez similaire à celle de la propriété 33 pour la dimension du code torique tridimensionnel : on ôte

un nœud de parité dans $\mathscr{G}_X^q$, et pour chaque nœud de parité restant, on trouve une erreur qui a un syndrome nul sauf en ce nœud de parité où il vaut 1.

Or si on enlève un nœud de parité en position $(2i_0+1, 2j_0)$, on peut trouver un chemin sur $\mathscr{G}_X^q$ qui le relie à n'importe quel nœud de parité $(2i+1, 2j)$: $(2i+1, 2j), (2i, 2j), (2i-1, 2j), \ldots (2i_0+1, 2j), (2i_0+1, 2j-1), \ldots (2i_0+1, 2j_0)$, voir la figure 6.13 pour un exemple.

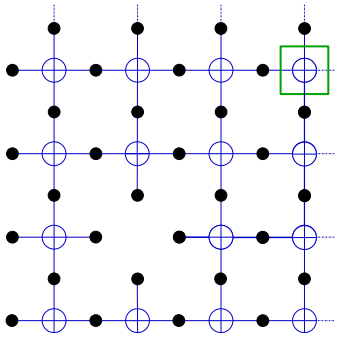


FIGURE 6.13 – Un exemple de chemin reliant un nœud de parité enlevé à un nœud de parité quelconque (ici encadré en vert) sur $\mathscr{G}_X^q$.

Il reste à affecter à ce chemin un mot de code : en binaire, c'est simple, il suffit de mettre un 1 sur chaque nœud variable du chemin. En q-aire, il faut chercher des éléments non-nuls. On renomme le chemin $c_0, v_1, c_1, v_2, \ldots c_m, v_m, c$, où c_0 est le nœud de parité enlevé et c un nœud de parité quelconque. On note w_i la valeur à donner au nœud v_i, x_i l'étiquette de l'arête $v_i \rightarrowtail c_i$ et x'_i l'étiquette de l'arête $v_{i+1} \rightarrowtail c_i$ (les notations sont analogues à celles de la preuve du lemme 8). La condition sur le syndrome égal à 1 sur le nœud de parité c impose $v_m = {x'_m}^{-1}$. Ensuite, pour avoir un syndrome nul sur le nœud de parité c_m, il faut et il suffit d'avoir $v_{m-1} = v_m x'_m x_m^{-1}$, et ainsi de suite jusqu'à v_1. Comme le dernier nœud de parité, c_0, a été supprimé, il n'y a pas de contrainte supplémentaire, et on a bien une erreur qui a un syndrome égal à 1 sur c et nul ailleurs.

■

Ce lemme peut également s'adapter au cas d'un code de cycles quelconque : pour chaque nœud de parité redondant dans le cas binaire, on effectue la construction de l'erreur spécifique, et on en cherche une version q-aire de même support, d'une manière équivalente au cas du code torique.

Il faut à présent montrer la propriété 43, sur la distance minimale.

Preuve de la propriété 43. On va utiliser ici la proposition 5 de la section 5.4.3 : pour chaque erreur sérieuse d'une base symplectique, il faut trouver n représentants de cette erreur qui soient de supports disjoints. La propriété est vraie pour

des erreurs binaires, mais on peut voir rapidement que la preuve s'adapte immédiatement au cas q-aire, puisque le seul point important est de savoir si il y a un élément nul ou non-nul sur une position donnée.

Pour cela, commençons par trouver la base symplectique des erreurs sérieuses : la dimension étant 2, on cherche des mots q-aires $\bar{x}_1, \bar{x}_2, \bar{z}_1, \bar{z}_2$ qui correspondent à des erreurs indétectées, donc $\bar{x}_i$ est un mot de code de $\mathscr{C}_X^q$ et $\bar{z}_i$ un mot de code de $\mathscr{C}_Z^q$, et qui vérifient

$$\bar{x}_i \, {}^{\mathbf{t}}(\bar{z}_j) = \delta_{i,j}$$

Pour les trouver, on prend des mots de même support que leurs équivalents binaires : on sait dans ce cas que de tels mots sont des grands cycles sur le graphe de Tanner de $\mathscr{C}_X^q$ et $\mathscr{C}_Z^q$ (voir figure 6.14).

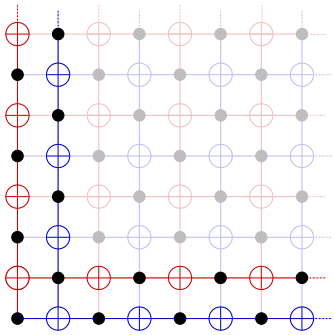


FIGURE 6.14 – Les erreurs sérieuses sur le code torique (binaire) : un grand cycle horizontal en X ($\bar{x}_1$), un grand cycle vertical en X ($\bar{x}_2$), un grand cycle vertical en Z ($\bar{z}_1$) et un grand cycle horizontal en Z ($\bar{z}_2$). On constate que $\bar{x}_1$ et $\bar{z}_1$ ont un nœud d'intersection, mais $\bar{x}_1$ et $\bar{z}_2$ n'en ont pas, etc.

On prend donc pour support de $\bar{x}_1$ un grand cycle horizontal, de $\bar{x}_2$ un grand cycle vertical sur le graphe de Tanner en X, et pour support de $\bar{z}_1$ un grand cycle vertical et de $\bar{z}_2$ un grand cycle horizontal sur le graphe de Tanner en Z (voir figure 6.14). Or grâce au lemme 8, pour chacun de ces cycles, on peut trouver un mot de code q-aire ayant exactement ce support.

Une fois trouvés ces mots de codes $\bar{x}_i$, $\bar{z}_i$, on a $\bar{x}_1 \, {}^{\mathbf{t}}(\bar{z}_1) = \alpha \neq 0$ puisqu'ils s'intersectent en une seule position : quitte à multiplier $\bar{x}_1$ par α^{-1}, on peut supposer que $\bar{x}_1 \, {}^{\mathbf{t}}(\bar{z}_1) = 1$, et de même pour $\bar{x}_2 \, {}^{\mathbf{t}}(\bar{z}_2)$. On a $\bar{x}_1 \, {}^{\mathbf{t}}(\bar{z}_2) = 0$ et $\bar{x}_2 \, {}^{\mathbf{t}}(\bar{z}_1) = 0$ car ils sont de supports disjoints (voir figure 6.14).

Pour trouver n représentants à supports disjoints de $\bar{x}_1$, on peut prendre les n grands cycles horizontaux parallèles (voir figure 6.15), plus précisément, $\bar{x}_{1,i}$ est un cycle de support $(0, 2i), (2, 2i), \ldots (2n-2, 2i)$ pour $i = 0, 1, \ldots n-1$. On peut construire de tels mots de code q-aires (ils ont pour support des cycles), et ils sont

bien tous des représentants (modulo le stabilisateur) de $\bar{x}_1$ car ils vérifient :

$$\bar{x}_{1,i}\,{}^{\mathbf{t}}(\bar{z}_1) = 1 \tag{6.10}$$
$$\bar{x}_{1,i}\,{}^{\mathbf{t}}(\bar{z}_2) = 0 \tag{6.11}$$

L'équation (6.11) est aisée à voir car les supports des $\bar{x}_{1,i}$ et de $\bar{z}_2$ sont disjoints. Les équations (6.10) donnent $\bar{x}_{1,i}\,{}^{\mathbf{t}}(\bar{z}_2) = \alpha_i \neq 0$, car les supports des deux mots coïncident en une seule position (en l'occurence, $(0, 2i)$). On peut donc multiplier les $\bar{x}_{1,i}$ par α_i^{-1} pour avoir les équations (6.10).

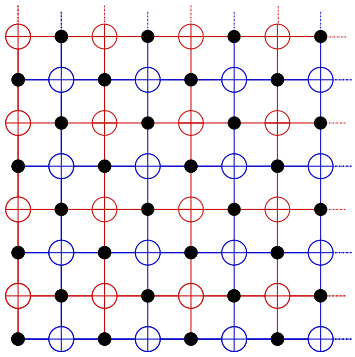


FIGURE 6.15 – Les différents représentants de $\bar{x}_1$ modulo le stabilisateurs : ce sont tous les grands cycles horizontaux en X.

On effectue cette construction pour $\bar{x}_2$ (tous les grands cycles verticaux en X), $\bar{z}_1$ (tous les grands cycles verticaux en Z) et $\bar{z}_2$ (tous les grands cycles horizontaux en Z), et on a les représentants voulus, ce qui suffit pour montrer que la distance minimale est n.

■

Adapter ce résultat à un code de cycles quelconque est non trivial. Pour une borne supérieure sur la distance minimale, il « suffit » d'expliciter une erreur sérieuse de poids adéquat. Or si on part d'une erreur sérieuse binaire de poids d, qu'on possède une autre erreur sérieuse qui n'y est pas orthogonale, quand bien même on arriverait à construire un équivalent q-aire de ces erreurs (il faudrait un équivalent du lemme 8 pour les deux parties, en X et en Z), il faudrait encore prouver qu'elles ne sont pas devenues orthogonales. Si elles s'intersectent en une seule position, comme c'était le cas ici entre les $\bar{x}_i$ et les $\bar{z}_i$, cela fonctionne, sinon le résultat est plus délicat à montrer.

De même pour une borne inférieure, il n'y a pas de méthode générale ; si on veut utiliser une méthode analogue au lemme et qu'on construit un certain nombre de représentants q-aires de $\bar{x}_1$, il n'est pas évident qu'ils soient tous non-orthogonaux

à $\bar{z}_1$, et tous orthogonaux à $\bar{z}_i$ pour $i \neq 1$. Dans le cas du code torique tout cela fonctionnait, car le premier cas était assuré par un seul point d'intersection, et le second par une intersection vide, mais pour un code de cycles quelconque, cela n'est pas clair.

6.2.4 Décodage et performances

On rappelle que le code torique étendu de paramètres n et m a pour longueur $2mn^2$. Comme il est intéressant de comparer des codes de même longueur finale, on peut fixer une longueur N qui soit un double d'un carré : ici $N = 1152 = 2 \times 24^2$. Cela permet d'avoir comme paramètres :

Paramètres	Code A	Code B	Code C
m	1	4	9
$q = 2^m$	2	16	512
n	24	12	8
$2n^2$	1152	288	128

Pour le décodage, on utilise l'algorithme de décodage BP CSS simple, c'est-à-dire qu'on décode les parties en X et en Z de façon indépendante. Un canal dépolarisant de paramètre p est donc vu comme deux canaux binaires symétriques de paramètre $2p/3$. On utilise un algorithme qui décode directement sur $\mathbb{F}_q$, avec le canal adapté, et qui est l'équivalent du BP sur le code binaire final.

Le code torique est connu pour avoir de très mauvaises performances sous l'algorithme BP, notamment à cause de distance minimale non dénégérée, qui est de 4 (les petits cycles de taille 8 sur le graphe de Tanner correspondent à des erreurs bénignes, mais le décodeur ne le sait pas). S'il existe des algorithmes spécifiques avec de bonnes performances, ils ne sont pas forcément adaptés à sa version étendue. L'intérêt est donc de comparer, pour le BP, la version d'origine à ses versions étendues.

On peut donc voir sur la figure 6.16 les courbes de décodage correspondantes. La différence est nette : sous un algorithme a priori simple, le code torique étendu montre de très bonnes performances, sans compter que pour $\mathbb{F}_{16}$ et $\mathbb{F}_{512}$, la dimension est plus grande.

6.3 Codes q-aires par produit

Lorsqu'on n'a pas de codes qui soient 2-réguliers, on ne peut pas appliquer la même méthode pour construire des matrices $\mathbf{H}_X^q$ et $\mathbf{H}_Z^q$ qui soient orthogonales dans $\mathbb{F}_q$. Cependant, si on parvient à ce résultat, la méthode pour obtenir les codes binaires étendus à partir des codes q-aires fonctionne encore. Si on peut trouver

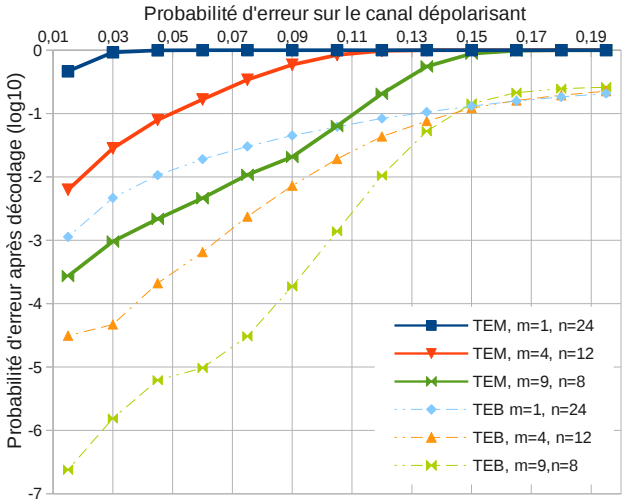


FIGURE 6.16 – Courbes de décodage des différentes variantes du code torique étendu. Le « TEM » est le Taux d'Erreurs par Mot, autrement dit le taux d'erreur résiduel, et le « TEB » est le Taux d'Erreur par Bit, qui permet de visualiser combien de bits erronés il peut rester même si le décodage échoue.

une autre méthode permettant d'avoir l'orthogonalité de $\mathbf{H}_X^q$ et $\mathbf{H}_Z^q$ à partir de matrices données $\mathbf{H}_X$ et $\mathbf{H}_Z$, alors on peut construire un code étendu.

Ce problème est difficile dans le cas général, mais lorsque le code a une structure particulière, on peut trouver une construction spécifique, et c'est justement le cas des codes produit (voir le chapitre 5 pour plus de détails sur la construction produit).

La construction est schématisée sur la figure 6.17 : on cherche à affecter des étiquettes aux arêtes, de façon à avoir la condition d'orthogonalité. Or il y a quatre types d'arêtes : celles qui relient les sommets de $V_1 \times V_2$ et ceux de $C_1 \times V_2$, etc. On définit donc quatre fonctions d'étiquetage x_1, x_2, z_1, z_2 (prenant en argument des arêtes, et à valeurs dans $\mathbb{F}_q$), pour chacune de ces catégories :

- $x_1((v_1, v_2) \rightarrowtail (c_1, v_2))$ pour les arêtes (en X) de $V_1 \times V_2$ vers $C_1 \times V_2$,
- $x_2((c_1, c_2) \rightarrowtail (c_1, v_2))$ pour les arêtes (en X) de $C_1 \times C_2$ vers $C_1 \times V_2$,
- $z_1((v_1, v_2) \rightarrowtail (v_1, c_2))$ pour les arêtes (en Z) de $V_1 \times V_2$ vers $V_1 \times C_2$,
- $z_2((c_1, c_2) \rightarrowtail (v_1, c_2))$ pour les arêtes (en Z) de $C_1 \times C_2$ vers $V_1 \times C_2$.

Quelle est la condition qui doit être vérifiée ? Si on prend deux nœuds de parité (c_1, v_2) en X et (v_1, c_2) en Z, ils sont reliés à des nœuds de variable communs si et seulement si $v_1 \rightarrowtail c_1$ et $v_2 \rightarrowtail c_2$ existent dans les graphes composants, et dans ce cas ils y a exactement deux nœuds de variable concernés : (v_1, v_2) et (c_1, c_2).

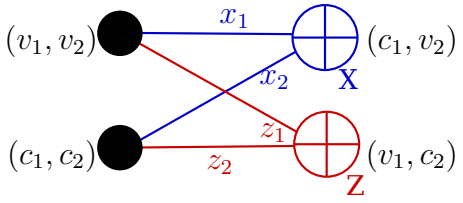


FIGURE 6.17 – Schéma de construction du code produit q-aire, avec les étiquettes sur les arêtes. Pour être valable, les étiquettes doivent vérifier $x_1 z_1 + x_2 z_2 = 0$

On en déduit alors que la condition d'orthogonalité s'écrit dans ce cas :

$$x_1(v_1, v_2 \rightarrowtail c_1, v_2) z_1(v_1, v_2 \rightarrowtail v_1, c_2) + x_2(c_1, c_2 \rightarrowtail c_1, v_2) z_2(c_1, c_2 \rightarrowtail v_1, c_2) = 0 \quad (6.12)$$

Il existe potentiellement plusieurs façons d'étiqueter les arêtes, une façon de le garantir est d'appliquer la construction suivante. On commence par définir quatre fonctions :

$$\begin{aligned} \alpha_1 : &\quad E_1 \to {\mathbb{F}_q}^* \\ \alpha_2 : &\quad E_2 \to {\mathbb{F}_q}^* \\ \gamma_X : &\quad C_1 \times V_2 \to {\mathbb{F}_q}^* \\ \gamma_Z : &\quad V_1 \times C_2 \to {\mathbb{F}_q}^* \end{aligned}$$

On donne alors les étiquettes :

$$\begin{aligned} x_1(v_1, v_2 \rightarrowtail c_1, v_2) = &\quad \alpha_1(v_1 \rightarrowtail c_1)\gamma_X(c_1, v_2) \\ x_2(c_1, c_2 \rightarrowtail c_1, v_2) = &\quad \alpha_2(v_2 \rightarrowtail c_2)\gamma_X(c_1, v_2) \\ z_1(v_1, v_2 \rightarrowtail v_1, c_2) = &\quad \alpha_2(v_2 \rightarrowtail c_2)\gamma_Z(v_1, c_2) \\ z_2(c_1, c_2 \rightarrowtail v_1, c_2) = &\quad \alpha_1(v_1 \rightarrowtail c_1)\gamma_Z(v_1, c_2) \end{aligned}$$

On peut exprimer cette définition de la façon suivante : l'étiquette de l'arête e est α(arête empruntée pour fabriquer e)γ(nœud de parité de e) : chaque arête du code produit est fabriquée à partir d'une arête d'un des deux graphes composants, plus un nœud (variable ou de parité). On peut alors constater que l'équation (6.12) est bien vérifiée pour tous v_1, v_2, c_1, c_2 tels que les arêtes $v_1 \rightarrowtail c_1$ et $v_2 \rightarrowtail c_2$ existent :

$$\begin{aligned} & x_1(v_1, v_2 \rightarrowtail c_1, v_2) z_1(v_1, v_2 \rightarrowtail v_1, c_2) + x_2(c_1, c_2 \rightarrowtail c_1, v_2) z_2(c_1, c_2 \rightarrowtail v_1, c_2) \\ = &\quad 2\alpha_1(v_1 \rightarrowtail c_1)\alpha_2(v_2 \rightarrowtail c_2)\gamma_X(c_1, v_2)\gamma_Z(v_1, c_2) \\ = &\quad 0 \end{aligned}$$

Il est difficile d'estimer si cette construction permet de trouver *toutes* les combinaisons vérifiant la condition d'orthogonalité. On peut cependant en avoir une

idée, en l'appliquant au code torique de paramètre n. En effet, ce code est à la fois un code produit et un code 2-régulier.

Dans la construction de la partie précédente, on sait exactement quelle est la dimension q-aire de l'espace des matrices possibles pour $\mathbf{H}_X^q$ et $\mathbf{H}_Z^q$: il y a exactement $2n^2 - 1$ éléments de ${\mathbb{F}_q}^*$ à choisir pour les arêtes de $\mathbf{H}_X^q$ (correspond à la taille de l'arbre couvrant maximal, donc au nombre de sommets du graphe moins 1). Pour $\mathbf{H}_Z^q$, il y a n^2 systèmes linéaires à résoudre, pour lesquels il y a un choix dans ${\mathbb{F}_q}^*$ pour chacun (en effet, l'espace des solutions est de dimension 1, donc il suffit de choisir la constante multiplicative). Cela donne au total $3n^2 - 1$ choix d'éléments non-nuls de ${\mathbb{F}_q}^*$.

Dans la construction basée sur les codes produit, il suffit de compter les choix pour chacune des fonctions α et β : cela donne une borne supérieure sur la dimension de l'espace obtenu. Le code composant (le code a répétition) a $2n$ arêtes, ce qui donne $2n + 2n$ choix à faire pour γ_1 et γ_2. Ce même code a n nœuds variable et n nœuds de parité, ce qui donne $n^2 + n^2 = 2n^2$ choix pour γ_X et γ_Z. Ainsi au total on a $2n^2 + 4n$ choix d'éléments non-nuls de ${\mathbb{F}_q}^*$.

Pour n assez grand, la construction basée sur les codes produit ne génère pas toutes les solutions possibles.

Chapitre 7

Codes spatialement couplés

Comme dans le chapitre précédent, on s'inspire d'une technique qui a fait ses preuves dans le monde classique pour l'adapter aux codes quantiques, dans deux constructions. La première, adaptée des codes produit (voir section 7.2), n'a pas donné de résultats satisfaisants, mais la seconde (voir section 7.3), publiée dans [AMT12b] puis améliorée et publiée dans [MTA13], a révélé d'excellentes performances pratiques avec un algorithme de décodage simple, bien que la distance minimale du code soit constante.

Les *codes spatialement couplés* ont été introduits plus récemment dans le paysage des codes classiques (dans [FZ99], où ils étaient appelés « codes convolutifs terminés »). Ils sont construits à partir d'une famille de codes LDPC, dont on place les graphes de Tanner « côte à côte », avant de mélanger entre elles les arêtes des codes voisins. On obtient ainsi un code agrandi (appelé spatialement couplé), dont la structure est proche du code d'origine.

Ces codes ont assez vite montré de très bons résultats pratiques (voir [LSZC05, KP10, KMS$^+$11, DJM11]) avec l'algorithme BP. On peut même adapter celui-ci pour que sa complexité n'augmente pas trop avec la longueur ([PIS$^+$10, ISUW11, IPS$^+$12]), tout en restant très performant : on pourra lire la section 7.1.2 pour plus de détails. Pour ne rien gâcher, leur distance minimale (sous quelques conditions sur les codes d'origine) est linéaire (voir [MPZC08, SPVC06, SPVC09, DDJ06]).

Enfin, et c'est là la grande force de cette construction, ces bons résultats pratiques sont *prouvés* : les performances de décodage sous l'algorithme BP sont celles du décodage du petit code LDPC d'origine, lorsqu'on utilise le décodage au *maximum de vraisemblance* (c'est-à-dire le décodage optimal du point de vue de la correction d'erreurs, souvent inutilisable en pratique à cause de sa trop grande complexité), et ce, *sur n'importe quel canal*. Ce résultat, d'abord conjecturé dans [LF10, KMRU10, LMFC10], puis montré dans le cas du canal à effacements seulement dans [KRU10], a été enfin prouvé pour tous les canaux dans [KRU13] ;

et il s'agit d'un des plus importants résultats de ces dix dernières années dans le domaine. De plus, il ouvre énormément de possibilités, car la construction est peu contrainte : on peut construire un code spatialement couplé à partir de n'importe quel code.

Adapter cette construction au cadre des codes quantiques semble très prometteur. En effet, pour un rendement fixé, si on augmente le degré des nœuds variable et de parité, le décodage optimal est meilleur, mais les performances pour le BP diminuent. Un code spatialement couplé a donc tout intérêt à avoir des degrés plus élevés, puisqu'il bénéficie alors des performances du décodage optimal. Or pour les codes quantiques, on cherche justement à augmenter le degré des nœuds de parité, car des nœuds de parité de degré faible signifient des stabilisateurs de petit poids, et donc une distance minimale non dégénérée très faible : si on peut augmenter cette distance sans perdre en efficacité de décodage (et même en l'augmentant), c'est un plus très appréciable.

Enfin, un dernier avantage non négligeable vient de la variante du BP utilisée (détaillée dans la section 7.1.2). Le décodage à fenêtre glissante ne met pas à jour tous les nœuds en même temps, ce qui évite le phénomène décrit en section 4.2 : une erreur ayant pour support une moitié de stabilisateur fait « hésiter » le décodeur, mais lorsqu'on la fixe sur certains nœuds, cette alternance disparaît, et l'algorithme peut mieux converger.

7.1 Présentation générale

7.1.1 Construction classique

Le principe de construction est le suivant. On commence par prendre un code $\mathscr{C}$ qu'on duplique $L+2\delta$ fois (on peut aussi choisir de prendre $L+2\delta$ codes différents, mais similaires dans leur construction). Chaque copie du code est appelée *couche*, et est indexée par son numéro de couche : ainsi, le nœud v de la couche t, est noté (v,t). Ensuite, pour chaque couche, on envoie des arêtes vers les couches voisines : envoyer une arête $(v,t)\rightarrowtail(c,t)$ de la couche t vers la couche t' revient à remplacer cette arête par l'arête $(v,t)\rightarrowtail(c,t')$. On limite également le déplacement des arêtes : ainsi les nœuds ne doivent pas être reliés à un autre nœud à distance supérieure à δ de leur propre couche (donc $|t-t'|\leqslant\delta$). On peut voir la figure 7.1 pour un petit exemple de telle construction.

On fixe ensuite les bits des couches extrêmes (les δ premières et les δ dernières) à 0. L doit être choisi suffisamment grand pour que la perte de rendement induite soit faible : si le code couche est de longueur n, de dimension k, et qu'on prend L

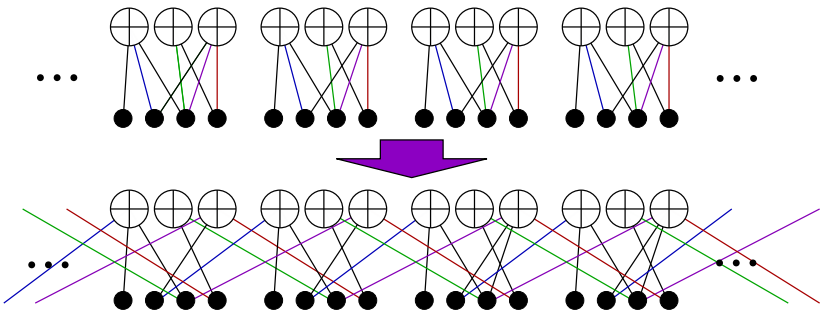


FIGURE 7.1 – Exemple de construction d'un code spatialement couplé. Ici, $\delta = 1$, et chaque arête d'une même couleur a été décalée de la même façon, par exemple les arêtes rouges ont été décalées d'une couche vers la droite.

couches et δ comme distance maximale, le rendement s'exprime :

$$R = \frac{kL}{(2\delta + L) \times n} \tag{7.1}$$

En effet, il faut déduire de la dimension les couches extrêmes (2δ couches de chaque côté), qui sont forcées à 0. Ce rendement tend bien vers $\frac{k}{n}$ quand L tend vers l'infini, ce qui signifie que le rendement du code spatialement couplé tend vers le rendement du petit code couche.

Il est à noter que certaines variantes de ces codes identifient les δ couches du début aux δ couches finales, formant ainsi une structure de code spatialement couplé « fermée ». Cette représentation, sensiblement équivalente, est parfois plus pratique pour traiter de certains aspects théoriques comme la distance minimale.

7.1.2 Décodage

On peut décoder les codes spatialement couplés avec un algorithme BP standard. Cependant, contrairement aux codes LDPC habituels, un nombre d'itérations constant n'est plus suffisant pour estimer correctement les probabilités finales. En effet, la structure « étalée » des codes spatialement couplés implique que pour que les informations transitent d'un bout à l'autre du graphe de Tanner, ces informations doivent parcourir une distance proportionnelle à L (dans le cas des LDPC usuels, cette distance est constante). Exécuter un BP classique sur l'ensemble du graphe peut donc être coûteux en temps.

On utilise dans ce cas une variante, appelée *décodage à fenêtre glissante*, voir par exemple [LF11]. Ce décodage consiste à utiliser une *fenêtre* de taille w fixée, sur laquelle on exécute le BP avec un nombre d'itérations constant, puis on décale la fenêtre d'une couche, et on recommence (voir figure 7.2).

Ainsi, pour chaque fenêtre, les couches déjà décodées « aident » les suivantes à se décoder plus facilement, et ces dernières, une fois décodées, aideront les couches suivantes à bien se décoder quand la fenêtre sera déplacée. Comme les bits des premières couches sont fixés à 0 (autrement dit, parfaitement décodés), ce « décodage assisté » peut commencer, et se propager le long des L couches.

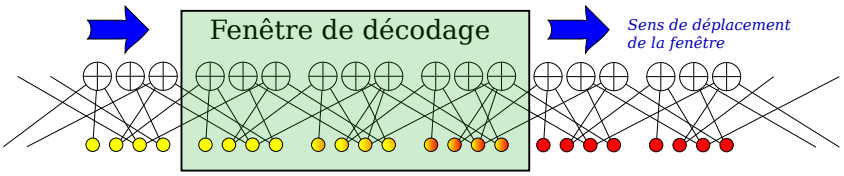


FIGURE 7.2 – Décodage itératif à fenêtre glissante : à gauche tout est parfaitement décodé (en jaune), et à droite rien n'est décodé. (en rouge) La fenêtre se déplace progressivement, et les couches parfaitement décodées aident à décoder les suivantes, et ainsi de suite.

La complexité est alors proportionnelle au nombre d'arêtes total, puisque chaque arête apparaît dans w couches, et est mise à jour un nombre constant de fois à chaque apparition. Dans le BP usuel, chaque arête serait mise à jour un nombre de fois proportionnel à L (de façon à bien transmettre les informations d'un bout à l'autre du graphe), donc la complexité serait proportionnelle à ce nombre d'arêtes multiplié par L. Il est montré dans [ISUW11] que l'efficacité de l'algorithme augmente exponentiellement avec la taille de la fenêtre : ainsi, pour profiter des bonnes performances, il n'est pas nécessaire d'avoir une fenêtre très grande : on prend w fixée, en général de l'ordre de grandeur de $2\delta + 1$.

7.1.3 Application aux codes quantiques

Comme expliqué dans l'introduction, ce nouveau type de construction semble très prometteur pour des codes quantiques.

Cependant, pour une version quantique, on se heurte à deux principaux problèmes. Le premier, lié à la construction, est la condition d'orthogonalité : un code CSS (qu'il soit spatialement couplé ou non) doit vérifier cette condition, il faut donc coupler les codes couches de façon à conserver cette orthogonalité.

Le second problème est de nature différente. Rappelons que pour fonctionner correctement, dans le cas classique, les bits aux extrémités doivent être fixés à 0. Or dans le cas du quantique, ce n'est pas possible, du moins pas avec le schéma habituel : on ne peut pas fixer simplement un qubit à $|0\rangle$.

En effet, lors du décodage en quantique, on n'a pas accès aux qubits en tant que tels. On n'a accès qu'à un syndrome, et on cherche, pour ce syndrome, l'erreur

qui l'a causée (au stabilisateur près). L'équivalent des « bits à 0 » est donc de connaître, d'une façon ou d'une autre, l'erreur *exacte* sur un qubit précis. Or, la façon usuelle (c'est à dire avec un code stabilisateur) d'obtenir cette erreur serait d'avoir deux nœuds de parité, reliés uniquement au nœud variable correspondant à ce qubit, l'un en X et l'autre en Z (ou un en X et un en Y, ou encore un en Y et un en Z), comme on mettrait en classique un nœud de parité relié uniquement à ce nœud variable pour fixer le bit associé à 0. Mais dans ce cas les stabilisateurs associés à ces deux nœuds de parité ne commutent pas : ce n'est donc pas possible.

Il faut donc utiliser une autre technique pour obtenir ces qubits extrémaux. Une technique couramment utilisée est celle de la *correction d'erreur assistée par intrication* (introduite dans [BDH06]). Elle nécessite d'avoir à sa disposition des paires de qubits intriqués, dont l'un des qubits serait *déjà* de l'autre côté du canal. On obtient alors, en reprenant les notations de la formule 7.1 :

$$R = \frac{kL - 2\delta n}{(2\delta + L) \times n} \tag{7.2}$$

En effet, il faut compter, en plus des 2δ couches qui doivent être fixées, les 2δ demi-paires de qubits intriqués, qui sont envoyées au préalable, et ce, sans erreurs : on peut les voir comme des qubits qu'on aurait envoyés dans une communication antérieure, ou encore que sur les qubits contenant de l'information, il faut en « garder » un certain nombre de côté pour transmettre les demi-paire intriquées pour la prochaine communication.

Une autre méthode, moins complexe, mise en place pour les codes présentés dans la section suivante, consiste à utiliser un autre code correcteur (dit *code interne*), qui peut être de rendement plus faible que celui cherché, pour protéger les qubits extrémaux, selon le schéma de la figure 7.3.

Cette technique peut être moins difficile à mettre en place que des qubits intriqués, mais en retour il peut y avoir quelques erreurs restantes après le décodage du code interne, et il faut en tenir compte (et espérer que le décodage y soit robuste).

Le rendement du code s'exprime alors, avec les mêmes notations que la formule 7.1, et en supposant que le code interne est de rendement R_i :

$$R = \frac{kL}{n(2\delta/R_i + L)} \tag{7.3}$$

En effet, les 2δ couches extrêmes sont encodées avec un code de rendement R_i : il faut donc $2n\delta/R_i$ qubits à ajouter dans la longueur du code, à la place des $2\delta n$ qubits. La dimension, elle, ne change pas.

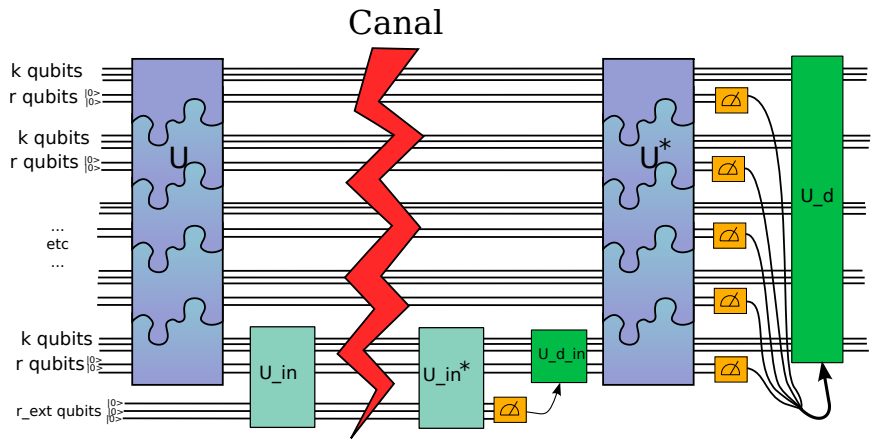


FIGURE 7.3 – Schéma d'encodage et de décodage, tel 3.3. On encode les qubits extrémaux *après* avoir encodé l'ensemble, puis à la sortie du canal, on mesure et corrige les qubits extrémaux avant de faire la mesure plus large sur le code spatialement couplé.

7.2 Codes produit spatialement couplés

On rappelle que, pour faire un code quantique CSS, il faut construire deux codes classiques qui soient orthogonaux (voir la définition équivalente 33, section 3.4.2). En supposant qu'on ait un code quantique, il n'est pas forcément évident d'en faire une version spatialement couplée, car il faut faire en sorte de garder cette condition d'orthogonalité. Ici, on étudie une construction de codes spatialement couplés, à partir du produit de codes (voir le chapitre 5, page 89 pour plus de détails sur ce produit).

7.2.1 Construction

Définition 76 (Code produit spatialement couplé)**.** *Soient $\mathscr{C}_1$ et $\mathscr{C}_2$ deux codes classiques (définis par leurs graphes de Tanner respectifs $\mathscr{G}[\mathscr{C}_i] = (V_i, C_i, E_i)$, δ et L deux entiers positifs, et f_1 et f_2 deux fonctions de E_1 (respectivement E_2) dans $[\![-\delta, \delta]\!]$. Le code produit spatialement couplé de $\mathscr{C}_1$ et $\mathscr{C}_2$, composé de L couches avec les fonctions paramètres f_1 et f_2 est un code quantique CSS défini par son graphe de Tanner $\mathscr{G} = (V, C_X, C_Z, E_X, E_Z)$ tel que :*

- $V = V_1 \times V_2 \times T \cup C_1 \times C_2 \times T$, *où* $T = [\![0, L + \delta - 1]\!]$
- $C_X = C_1 \times V_2 \times T$
- $C_Z = V_1 \times C_2 \times T$
- $E_X = \{(v_1, v_2, \mathbf{t}) \rightarrowtail (c_1, v_2, \mathbf{t} + \mathbf{f_1}(\mathbf{v_1} \rightarrowtail \mathbf{c_1})), v_1 \rightarrowtail c_1 \in E_1, v_2 \in V_2\}$
  $\cup \{(c_1, c_2, \mathbf{t}) \rightarrowtail (c_1, v_2, \mathbf{t} + \mathbf{f_2}(\mathbf{v_2} \rightarrowtail \mathbf{c_2})), v_2 \rightarrowtail c_2 \in E_2, c_1 \in C_1\}$

– $E_Z = \{(v_1, v_2, \mathbf{t}) \rightarrowtail (v_1, c_2, \mathbf{t} + \mathbf{f_2}(\mathbf{v_2} \rightarrowtail \mathbf{c_2})), v_2 \rightarrowtail c_2 \in E_2, v_1 \in V_1\}$
$\cup \{(c_1, c_2, \mathbf{t}) \rightarrowtail (v_1, c_2, \mathbf{t} + \mathbf{f_1}(\mathbf{v_1} \rightarrowtail \mathbf{c_1})), v_1 \rightarrowtail c_1 \in E_1, c_2 \in C_2\}$

où les coordonnées de la 3e composante sont prises modulo $L + \delta$.

Informellement, on peut voir la construction comme cela : chaque nœud est un triplet de deux nœuds des deux graphes d'origine et d'une composante temporelle (t). On trace une arête entre deux nœuds si la première coordonnée ne bouge pas, et la seconde correspond à une arête e_2 du 2e graphe : on ajoute alors à t un décalage qui est égal à $f_2(e_2)$. On a le symétrique lorsque la première coordonnée change selon une arête e_1, et que la 2e coordonnée ne bouge pas, on ajoute alors $f_1(e_1)$ à la composante temporelle.

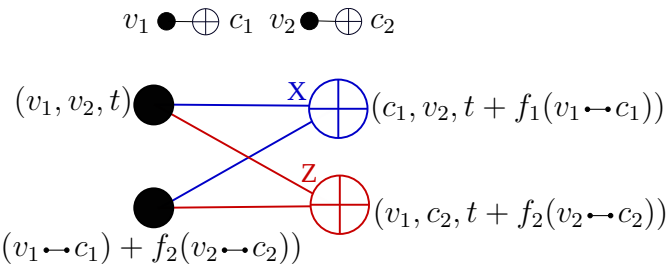


FIGURE 7.4 – Schéma de la construction du code produit spatialement couplé (voir définition 76) : pour obtenir une arête du graphe du produit, on fixe une coordonnée et on choisit une arête de l'autre coordonnée, puis on ajoute un décalage donné par f_1 ou f_2 de l'arête empruntée.

Avec le modulo $L + \delta$, c'est une construction « fermée », où les deux séries de δ couches extrêmes sont confondues. De plus, c'est une construction périodique : chaque arête du graphe couche est décalée de la même façon. Par exemple, si on a une arête $(v_1, v_2, 1) \rightarrowtail (c_1, v_2, 3)$, alors on a aussi d'autres arêtes de la forme $(v_1, v_2, t) \rightarrowtail (c_1, v_2, t + 2)$.

Il faut encore vérifier que le code décrit ci-dessus est un code quantique valide :

Propriété 44 (Validité du code produit spatialement couplé)**.** *Le code décrit dans la définition 76 est un code quantique valide, c'est-à-dire pour tous $c_X \in C_X$, $c_Z \in C_Z$, on a*

$$\mathcal{S}(c_X) \star \mathcal{S}(c_X) = 0$$

Démonstration. La preuve est très similaire au cas non-spatialement couplé. On considère deux nœuds de parité, $(c_1, v_2, t) \in C_X$ et $(v_1, c_2, t') \in C_Z$. On suppose qu'ils anti-commutent en une position $(v_1', v_2', t_0) \in V_1 \times V_1 \times T$ (vue la symétrie de la construction, le raisonnement marche de la même façon s'ils anti-commutent en une position de $C_1 \times C_2 \times T$). On a alors trois propriétés qui découlent nécessairement de la construction :

- $v'_1 = v_1$, et $v'_2 = v_2$, car un nœud de parité et un nœud de variable adjacents doivent avoir une coordonnée commune.
- Il existe, dans les deux graphes des codes composants, $v_1 \bullet\!\!\rightarrow c_1 \in E_1$ et $v_2 \bullet\!\!\rightarrow c_2 \in E_2$.
- $t_0 + f_1(v_1 \bullet\!\!\rightarrow c_1) = t$ et $t_0 + f_2(v_2 \bullet\!\!\rightarrow c_2)$.

Par construction, il existe également deux arêtes : $(c_1, c_2, t + f_2(v_2 \bullet\!\!\rightarrow c_2)) \bullet\!\!\rightarrow (c_1, v_2, t) \in E_X$ et $(c_1, c_2, t' + f_1(v_1 \bullet\!\!\rightarrow c_1)) \bullet\!\!\rightarrow (v_1, c_2, t) \in E_Z$. Or comme $t' + f_1(v_1 \bullet\!\!\rightarrow c_1) = t + f_2(v_2 \bullet\!\!\rightarrow c_2)$, les deux origines de ces arêtes sont bien le même nœud : les deux nœuds de parité anti-commutent en une autre position. Comme il n'y a pas d'autre position où ils peuvent anti-commuter, cela signifie qu'ils anti-commutent en exactement deux positions, et donc commutent.

■

On peut remarquer que, pour obtenir une structure spatialement couplée à partir du produit, on pourrait appliquer n'importe quel « décalage » au paramètre t, mais pour garder la condition d'orthogonalité, il faut faire en sorte de garder le même schéma en « papillon ». Considérons le cas où on choisit d'appliquer quatre fonctions de décalage quelconques g_1, g_2, g_3 et g_4 correspondantes aux différentes arêtes obtenues par produit (voir la figure 7.5 pour les notations), et voyons quelles conditions ces fonctions doivent vérifier.

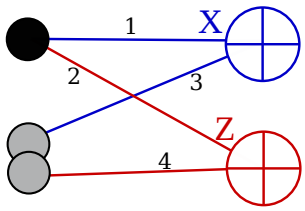


FIGURE 7.5 – Schéma de construction qui ne commute pas (encore), avec les arêtes numérotées. Pour garder l'orthogonalité, il faut faire en sorte que les deux sommets gris soient confondus.

L'arête 1 relie un nœud variable (v_1, v_2, t_1) à un nœud de parité (c_1, v_2, t_2). Elle est construite à partir d'un nœud $v_2 \in V_2$ et d'une arête $e_1 = v_1 \bullet\!\!\rightarrow c_1 \in E_1$. On peut donc écrire :

$$t_2 = g_1(t_1, e_1, v_2)$$

De même, l'arête 2 $((v_1, v_2, t_1) \bullet\!\!\rightarrow (v_1, c_2, t_3))$ est construite à partir d'une arête $e_2 = v_2 \bullet\!\!\rightarrow c_2 \in E_2$ et d'un nœud variable $v_1 \in V_1$:

$$t_3 = g_2(t_1, e_2, v_1)$$

L'arête 3 $((c_1, c_2, t_4) \rightarrowtail (c_1, v_2, t_2))$ est construite à partir d'une arête $e_2 = v_2 \rightarrowtail c_2 \in E_2$ et de $c_1 \in C_1$:

$$t_4 = g_3(t_2, e_2, c_1)$$

Enfin, l'arête 4 $((c_1, c_2, t_4) \rightarrowtail (v_1, c_2, t_3))$ est construite à partir d'une arête $e_1 = v_1 \rightarrowtail c_1 \in E_1$ et de $c_2 \in C_2$:

$$t_4 = g_4(t_3, e_1, c_2)$$

En combinant ces quatre équations, on obtient la condition nécessaire et suffisante pour avoir un code CSS valide :

$$\begin{aligned} &\forall t \in T,\ \forall e_1 = v_1 \rightarrowtail c_1 \in E_1,\ \forall e_2 = v_2 \rightarrowtail c_2 \in E_2, \\ &\quad g_3(g_1(t, e_1, v_2), e_2, c_1) = g_4(g_2(t, e_2, v_1), e_1, c_2) \end{aligned} \tag{7.4}$$

Ainsi, dans la construction, on a choisi $g_1(t, e_1, v_2) = t + f_1(e_1)$ et les autres sous une forme analogue. D'autres combinaisons seraient possibles, mais il est difficile de garder la condition (7.4) tout en gardant une forme simple pour les g_i.

On peut espérer que sa distance minimale soit au moins aussi bonne que celle du code produit. Or celle-ci est nettement plus complexe à calculer que pour le code produit.

7.2.2 Décodage et performances

Un décodage à fenêtre glissante a été implémenté pour ce type de codes avec les paramètres suivants :

- $\delta = 2$,
- $w = 9$,
- $L = 50$
- $d_V = 4$, $d_C = 9$

Les longueurs utilisées pour le code couche sont plus petites que celles utilisées dans la section 5.3, pour des raisons de complexité :

	longueur code composant	longueur code produit (couche)	longueur totale
Calcul	n	$n_p = n^2 + \left(\frac{4n}{9}\right)^2$	$N = (2\delta + L)n_p$
Code 1	90	9 700	523 800
Code 2	135	21 825	1 178 550

Même si les performances pourraient probablement s'améliorer en augmentant le degré des nœuds et la longueur du code composant, ces premiers résultats peu encourageants (malgré une longueur finale pourtant déjà grande) n'ont pas mené à d'autres expérimentations. On retrouve en fait les problèmes liés au code produit (détaillées dans la section 5.3.3), avec la structure qui est proche de plusieurs

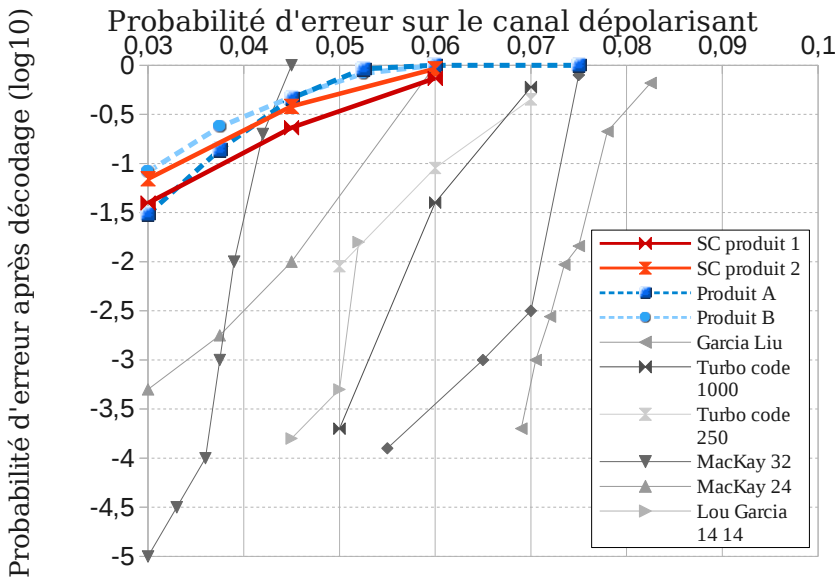


FIGURE 7.6 – Performances du code produit spatialement couplé, comparées avec celles du code produit simple, et d'autres constructions de codes quantiques (voir section 3.7).

instances d'un code. Cependant, il est possible qu'avec une grande longueur et des degrés élevés (évitant ainsi les problèmes de distance minimale non dégénérée faible) les performances s'améliorent un peu.

7.3 Construction LDGM

Cette construction est une version spatialement couplée de la construction de codes quantiques de [LGF06, GFL08], qui se base sur des codes à matrice génératrice creuse (Low Density Generating Matrix en anglais, d'où l'acronyme LDGM). Cette construction conduit à des codes quantiques dont la distance minimale est constante, mais pas trop faible, et des performances correctes, avec une assez grande liberté de construction.

La version spatialement couplée de cette construction a abouti à des performances excellentes comparées à celles existantes (publiée dans [MTA13]), malgré une distance minimale constante. De plus, on obtient une structure de graphe de Tanner assez particulière, qui ne fait pas apparaître de trop petits cycles : un algorithme de décodage tenant compte des corrélations entre X et Z (le BP CSS mixte, voir 4.2.5), dont les performances sont d'ordinaire entravées par ces petits cycles, donne des résultats encore meilleurs.

7.3.1 Présentation de la construction de Liu et Garcìa-Frìas

Souvent, l'idée de base de la construction d'un code CSS est une situation où, « naturellement », deux matrices vérifient $A\,{}^{\mathbf{t}}(B) = 0$. Ici, cette idée est simple : pour un code classique $\mathscr{C}$, sa matrice de parité H et sa matrice génératrice G vérifient cette condition, quasiment par définition (voir section 1.2.1).

Prendre pour matrices $\mathbf{H}_X$ et $\mathbf{H}_Z$ les matrices H et G donnerait, a priori, un code de dimension 0, mais l'idée peut être raffinée : en effet, en ôtant plusieurs lignes de H ou de G (ou en ne gardant qu'un certain nombre de combinaisons linéaires de ces lignes), la condition d'orthogonalité reste vraie, mais la dimension peut devenir positive.

La construction se fait en plusieurs étapes.

- On construit une matrice carrée P, de dimensions $n/2 \times n/2$, de degré ligne et colonne d. P est donc creuse.
- On écrit $\tilde{\mathbf{H}}_X = (P|I)$ et $\tilde{\mathbf{H}}_Z = (I|\,{}^{\mathbf{t}}(P))$. On a alors de façon évidente $\tilde{\mathbf{H}}_X\,{}^{\mathbf{t}}\left(\tilde{\mathbf{H}}_Z\right) = 0$, donc un code CSS de longueur n et de dimension nulle, du moins a priori ($k \geqslant n - (n/2) - (n/2) = 0$).
- On construit deux matrices M_X et M_Z, de dimension $l \times n/2$, creuses également. On choisit en l'occurence un degré colonne constant, égal à un paramètre y, et un degré ligne variable : s_1 lignes de degré 1, et s_2 lignes de degré x.
- On écrit $\mathbf{H}_X = M_X\tilde{\mathbf{H}}_X$ et $\mathbf{H}_Z = M_Z\tilde{\mathbf{H}}_Z$. Ainsi on a toujours $\mathbf{H}_X\,{}^{\mathbf{t}}(\mathbf{H}_Z) = M_X\tilde{\mathbf{H}}_X\,{}^{\mathbf{t}}\left(\tilde{\mathbf{H}}_Z\right)\,{}^{\mathbf{t}}(M_Z) = 0$, et la dimension vaut au moins : $k \geqslant n - 2l$.

Pour obtenir le rendement voulu R, il faut choisir les paramètres de la façon suivante :

$$l = n\frac{1-R}{2} \tag{7.5}$$

$$s_1 = \frac{lx - yn/2}{x-1} \tag{7.6}$$

$$s_2 = \frac{yn/2 - l}{x-1} \tag{7.7}$$

En effet, la dimension finale du code vaut (si on suppose qu'il n'y a pas de ligne redondante dans M_X et M_Z) $k = n - 2l$ puisque la matrice $\mathbf{H}_X$ a pour dimension $l \times n$. Donc pour obtenir un rendement $R = \frac{k}{n}$, il faut choisir $\frac{n-2l}{n} = R$, ce qui donne bien l'équation (7.5).

Les deux autres équations, (7.6) et (7.7), s'obtiennent en résolvant le système

suivant :

$$\begin{cases} yn/2 &= s_1 + s_2 x \\ l &= s_1 + s_2 \end{cases}$$

La première équation du système est celle qui compte le nombre d'arêtes du graphe de M_X (c'est à dire de le nombre de 1 de la matrice M_X), l'autre le nombre de lignes de la matrice M_X (ou M_Z).

On obtient un graphe de Tanner similaire à celui de la figure 7.7.

On a cependant une distance minimale constante :

Propriété 45 (Distance minimale de la construction LDGM)**.** *La distance minimale du code ainsi construit est d'au plus* $d+1$.

Démonstration. Pour montrer la borne supérieure, il suffit d'exhiber une erreur sérieuse de poids $d+1$.

Considérons une ligne x de la matrice $\tilde{\mathbf{H}}_X = (P|I)$. Elle est de poids $d+1$, et vérifie $\tilde{\mathbf{H}}_Z\,{}^{\mathbf{t}}(x) = 0$, donc également $\mathbf{H}_Z\,{}^{\mathbf{t}}(x) = M_Z\tilde{\mathbf{H}}_Z\,{}^{\mathbf{t}}(x) = 0$: c'est donc une erreur indétectée. Il existe $n/2$ telles erreurs, et elles sont indépendantes.

Pour qu'aucune d'elles ne soit une erreur sérieuse, il faudraient qu'elles soient toutes dans VectLi($\mathbf{H}_Z$). Or cet espace est de dimension inférieure ou égale à l (le nombre de lignes de la matrice), qui est strictement inférieur à $n/2$ si on souhaite avoir une dimension strictement positive. Ce n'est donc pas possible : il existe au moins une de ces erreurs x qui soit une erreur sérieuse.

■

Pour le décodage, même si le graphe de la figure 7.7 a l'air plus complexe à première vue que le graphe condensé de $\mathbf{H}_X$ et $\mathbf{H}_Z$ (obtenu en calculant $\mathbf{H}_X$ et en donnant son graphe de Tanner), le décodage est plus efficace sur ce graphe, entre autres car le nombre d'arêtes est moins grand. On peut également voir l'intérêt des matrices M_X et M_Z, qui mélangent un peu plus les informations circulant sur les arêtes.

7.3.2 Version spatialement couplée

L'avantage de cette construction est qu'elle permet d'être modifiée facilement, entre autres on peut en faire une version spatialement couplée. Pour cela, on construit $L+2\delta$ tels codes, et on choisit de coupler à deux endroits : au niveau de P et au niveau de M_X (et M_Z). Plus précisément, on choisit deux paramètres, δ_M et δ_P (qui peuvent être égaux). On choisit alors de coupler les arêtes de P ensemble avec le paramètre δ_P d'une part, les arêtes de M_X avec le paramètre δ_M et les arêtes de M_X avec le paramètre δ_M d'autre part (voir figure 7.8)

Le mélange des arêtes consiste, informellement, à répartir les arêtes sur les $2\delta_P+1$ couches voisines (incluant la couche courante). Plus formellement, le couplage de P s'effectue de la façon suivante :

- Pour chaque couche, on partitionne les arêtes de P en $2\delta_P + 1$ ensembles de même taille, indexés par $\{-\delta_P, -\delta_P + 1, \dots 0, 1, \dots \delta_P\}$, aléatoirement.
- Pour chaque couche t, on envoie les arêtes de l'ensemble E_i dans la couche $t + i$.

Pour ${}^{\mathbf{t}}(P)$, on envoie les arêtes vers les mêmes couches que l'arête équivalente dans P (ie, si on envoie $(v,t)\rightarrowtail(c,t)$ vers la couche t', alors on envoie aussi $(c,t)\rightarrowtail(v,t)$ de ${}^{\mathbf{t}}(P)$ vers t'

On effectue la même opération pour M_X et M_Z, en prenant δ_M à la place de δ_P.

Il faut encore vérifier que la construction obtenue est bien un code CSS valide, autrement dit que la condition d'orthogonalité est toujours vérifiée. Pour cela, il suffit de le vérifier pour le couplage de P : si on a $\tilde{\mathbf{H}}_X^+$ et $\tilde{\mathbf{H}}_Z^+$, les versions spatialement couplées de $\tilde{\mathbf{H}}_X$ et $\tilde{\mathbf{H}}_Z$ qui vérifient la condition d'orthogonalité, alors naturellement ${M_X}^+\tilde{\mathbf{H}}_X^+$ et ${M_Z}^+\tilde{\mathbf{H}}_Z^+$ la vérifient aussi.

Or, lorsqu'on envoie une arête $(v,t)\rightarrowtail(c,t)$ vers la couche t', sur la couche t, cela revient à dire que, dans la matrice P locale, on enlève un 1 en position (v,c). Mais comme on envoie aussi l'arête correspondante $(c,t)\rightarrowtail(v,t)$ du graphe de ${}^{\mathbf{t}}(P)$ vers t', on a aussi ôté un 1 en position (v,c) : on a toujours deux matrices dont l'une est la transposée de l'autre, ce qui permet d'avoir l'orthogonalité. Le raisonnement marche également du côté de la couche t' pour l'arête qui arrive (un 1 est ajouté dans la matrice P et aussi dans la matrice transposée).

Il est nécessaire de coupler P, car si lors du décodage, toute la partie supérieure (correspondant à M_X) est résolue, alors on peut enlever du graphe de Tanner cette partie, et il reste le graphe de Tanner de plusieurs codes de longueur n, de rendement $1/2$ (voir figure 7.9, partie basse) : certes ce sont des codes de rendement plus faible que le code classique d'origine, mais ces codes sont *non spatialement couplés*. On ne profite donc pas de la puissance du couplage spatial, et de plus on empire les performances : il suffit qu'une seule des couches échoue son décodage pour que le décodage complet échoue. Et plus L augmente, plus la probabilité d'échec sur une seule couche est élevée.

De même, il est nécessaire de coupler M_X et M_Z, car si tout le décodage de la partie concernant P s'est déroulé correctement, on peut ôter cette partie du graphe, et alors on obtient plusieurs graphes de Tanner équivalents à ceux des différentes matrices M_X (ou M_Z), c'est-à-dire des codes classiques de rendement $\frac{n-l}{n/2} = R$, mais là encore *non spatialement couplés* (voir figure 7.9, partie haute). Pour les mêmes raisons que plus haut, cela donne de mauvaises performances de décodage.

7.3.3 Décodage et performances

Le décodage de ce code CSS peut s'effectuer avec un BP CSS simple (voir 4.2.4), mais il est plus intéressant d'effectuer un BP CSS mixte (voir 4.2.5). En effet, ce dernier décodage permet de tenir compte des corrélations entre X et Z, et la structure particulière du graphe de Tanner fait apparaître des cycles de taille 12 au lieu de cycles de taille 4.

Les paramètres suivants ont été utilisés : $\delta = \delta_M = \delta_P = 2$, et

Code	$n/2$	x	y	L	n_{final}
SC 1A	480	9	3	150	147 840
SC 1C	1920	9	3	150	591 360
SC 2A	480	13	4	150	147 840
SC 2C	1920	13	4	150	591 360

On peut voir sur la figure 7.10 une comparaison des courbes de décodage de ces différents codes. On peut alors faire différents constats :

- les courbes ne sont pas très loin de la borne de hachage (voir la figure 7.11 pour une comparaison avec d'autres constructions existantes), il est même possible de dépasser la borne de hachage CSS (grâce à l'utilisation de l'algorithme BP CSS mixte, qui utilise les corrélations entre X et Z), ce qui est excellent ;
- les codes 1A et AC ont un seuil légèrement meilleur (c'est-à-dire que lorsque p diminue, la probabilité résultante plonge plus vite), mais la courbe des codes 2A et 2C descend plus bas ;
- le décodage est bien meilleur lorsqu'on utilise le BP CSS mixte (qui utilise la corrélation entre X et Z) que lorsqu'on utilise le BP CSS simple (voir section 4.2 pour plus de détails sur les différents algorithmes). Dans ce dernier cas, notamment, on ne peut pas dépasser la borne de hachage CSS.

On peut voir également sur la figure 7.11 quelques-uns de ces codes comparés aux codes existants, et constater à quel point ces codes fonctionnent très bien : ce sont les premiers (pour ce rendement) à s'approcher raisonnablement de la borne de hachage, et à dépasser la borne de hachage CSS.

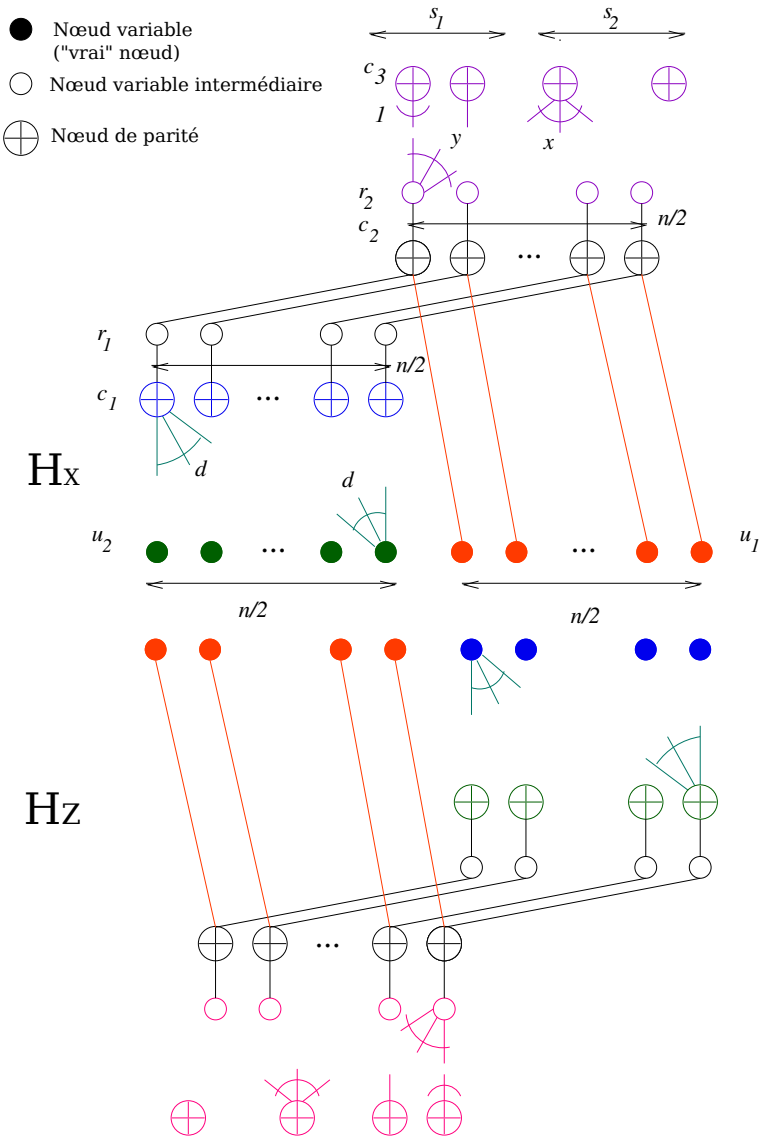


FIGURE 7.7 – Représentation sous forme de graphe de Tanner de la construction de [GFL08]. Les points pleins sont les « vrais » nœuds variables, ceux qui correspondent à des qubits. Les nœuds creux sont des nœuds variables artificiels ajoutés (ils n'ont donc aucune probabilité a priori dans l'algorithme de décodage). La partie basse de $\mathbf{H}_X$ est décomposée en deux sections : à gauche, le graphe représentant P, et à droite un graphe représentant l'identité. En haut, la matrice M_X qui mélange les lignes de $\tilde{\mathbf{H}}_X$. On retrouve ces éléments (ou les éléments équivalents) pour $\mathbf{H}_Z$ en bas.

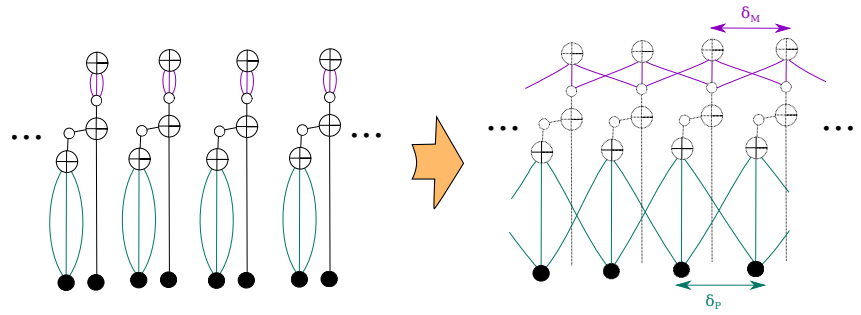


FIGURE 7.8 – Couplage des arêtes : chaque petit graphe représente une couche du code (uniquement la partie en X, mais la partie en Z est symétrique), et est une version compressée de la figure 7.7. On mélange les arêtes de M_X et les arêtes de P séparément.

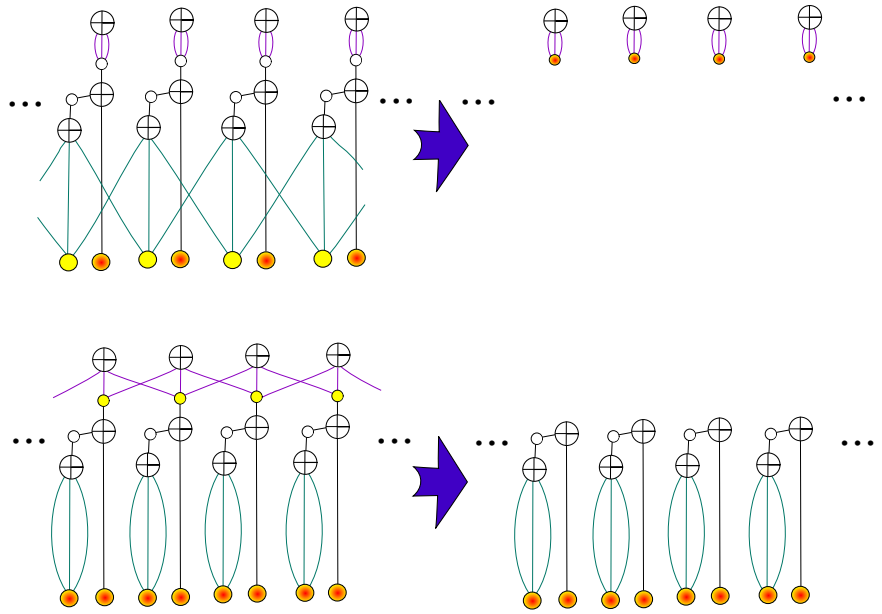


FIGURE 7.9 – Cas de couplages qui ne marchent pas. En haut, si on couple au niveau de P et pas au niveau de M, alors on peut se retrouver (si toute la partie concernant P est décodée), à un graphe équivalent à plusieurs instances de M, non spatialement couplé. En bas, si on couple au niveau de M et pas au niveau de P, on a le même problème : si tout l'étage M est décodé, alors on a un graphe équivalent à plusieurs instances de $(P|I)$, et non spatialement couplé.

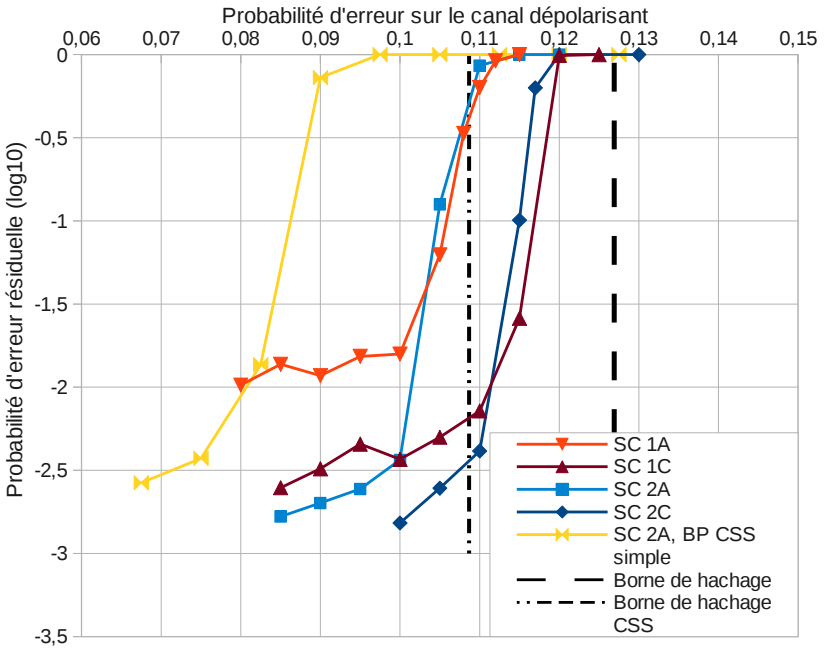


FIGURE 7.10 – Comparaison de différents types de codes spatialement couplés. La cinquième courbe, « SC 2A BP CSS simple », consiste à prendre le code correspondant et à le décoder avec l'algorithme BP CSS simple, c'est-à-dire en ne tenant pas compte des corrélations entre X et Z.

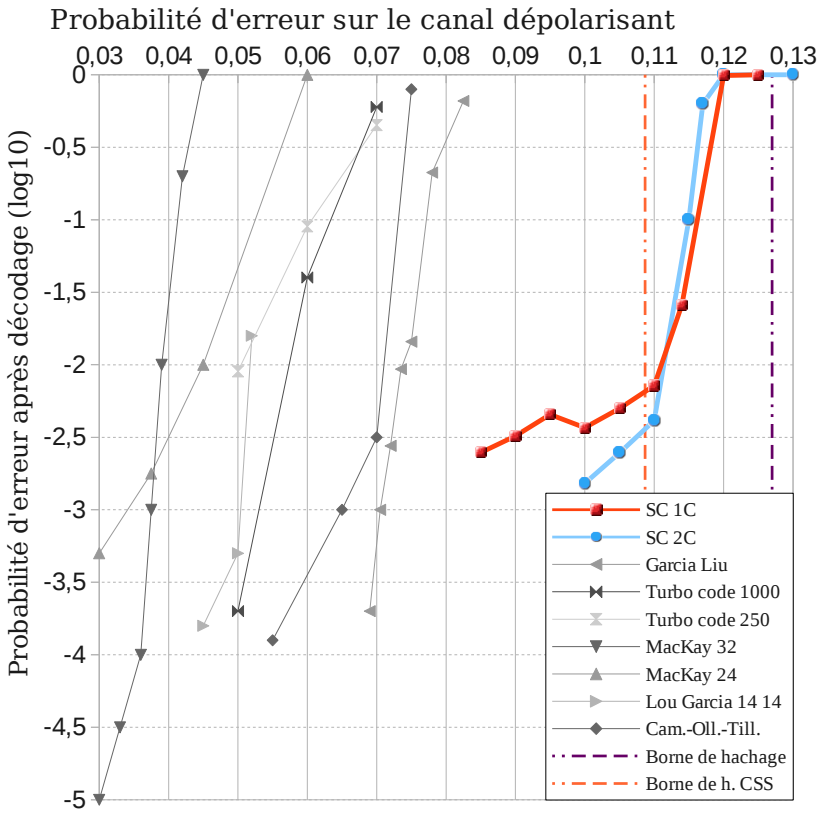


FIGURE 7.11 – Comparaison des codes spatialement couplés avec divers codes de même rendement pris dans la littérature : les résultats sont excellents, même si les courbes de décodage ne descendent pas très bas (à cause de la distance minimale constante).

Conclusion

La correction d'erreurs quantique est un sujet assez large, dans lequel il reste encore beaucoup de choses à découvrir, même si beaucoup de progrès ont été faits.

Notamment, la borne de hachage (introduite en section 3.6), qui est censée apporter une borne *inférieure* sur la capacité de correction, était encore loin d'être atteinte par des codes en pratique. Les raisons sont nombreuses, dans le cas des codes LDPC, l'une d'elles est la difficulté à exploiter la dégénérescence.

Les travaux du chapitre 5 ont mis en valeur le fait que la construction produit n'a pas de bonnes performances du fait de sa structure. Quand à ses variantes, elles n'ont soit pas été testées, soit ont donné des résultats assez décevants en termes de performances. Le problème vient principalement de sa structure en « couches » presque indépendantes (voir 5.3.3), et qui font décroître la probabilité d'erreur très lentement avec la longueur. Or, si pour compenser on augmente trop la longueur, les problèmes de distance minimale non dégénérée apparaissent alors. Cependant, son étude porte à croire que ces performances pourraient s'améliorer en utilisant mieux la dégénérescence : la longueur du code pourrait alors augmenter suffisamment pour que la probabilité d'erreur devienne sufisamment faible. On peut d'ailleurs remarquer que le code torique en est un cas particulier ; or il a de très bonnes performances sous un algorithme adapté. De plus, son schéma en produit, relativement simple, peut donner lieu à d'autres variantes (une récente en est [KP12b]), susceptibles de mieux fonctionner. Sa variante tridimensionnelle, qui semblait simple au premier abord, est en réalité bien plus complexe à étudier : même dans un cas simple (le code torique tridimensionnel), sa dimension est non-triviale, et sa distance minimale inconnue, bornée par $\sqrt{n}$.

Les codes q-aires, étudiés au chapitre 6, donnent un excellent moyen d'améliorer à peu de frais les performances d'un code quantique. En revanche, les propriétés du code telles que la dimension et la distance minimale sont difficiles à estimer dans le cas général. La dimension peut éventuellement diminuer, mais est toujours supérieure ou égale à $n - r_X - r_Z$; par contre la distance minimale reste inconnue dans le cas général, car elle est plus difficile à calculer dans le cas q-aire. Cependant, les résultats sur le code torique laissent à penser qu'elle serait probablement égale ou plus grande à celle du code binaire. Il reste à tester cette construction sur

d'autres codes 2-réguliers, pour voir si les performances sont toujours aussi bonnes par rapport au code binaire d'origine. Il faudrait cependant ne pas oublier de tester les deux parties (en X et en Z), car la construction n'est pas symétrique : notamment si la partie en X du code q-aire vérifie la condition de cycles, il n'y a pas de raisons que ce soit le cas pour la partie en Z (c'est le cas pour le code torique, mais cela vient de la structure très particulière de celui-ci). Il se peut donc que l'un des décodages soit meilleur que l'autre.

Enfin, le dernier chapitre (7) étudiant des constructions spatialement couplées, met en valeur l'intérêt de ce nouveau modèle de codes quantiques, qui même sans tenir compte de la dégénérescence obtient d'excellentes performances. D'une part, les simulations avec les codes LDGM spatialement couplés peuvent encore être améliorées : en effet, la génération des graphes de Tanner est pour le moment peu optimisée, notamment le couplage. En effet, actuellement, il se peut qu'un nœud de parité donné voit toutes ses arêtes partir dans des couches du « même côté », ce qui rend le décodage moins efficace. D'autre part, les travaux de [KRU13] prouvent que pour des codes classiques les performances sont bonnes : cela pose la question d'un résultat analogue dans le cas des codes quantiques. Même en l'absence d'une réponse claire, on peut espérer de bonnes performances pratiques avec des constructions spatialement couplées d'autres familles de codes quantiques, pour peu qu'on puisse garder l'orthogonalité en couplant spatialement (ce qui n'est pas a priori évident).

De plus, il est intéressant de constater la différence significative de performances entre le cas où on exploite les corrélations entre X et Z ou non (voir section 7.3.3). Si ces corrélations ne sont pas toujours simples à exploiter avec le BP quantique, du fait de la présence des petits cycles de taille 4 (voir section 4.2), il est clairement utile de s'intéresser à des techniques permettant de les exploiter.

Bibliographie

[Abb13] Mamdouh Abbara. *Turbo-codes quantiques*. Thèse, École Polytechnique, Avril 2013.

[Aly07] Salah A. Aly. A class of quantum LDPC codes constructed from finite geometries. *CoRR*, abs/0712.4115, 2007.

[AMT12a] I. Andriyanova, D. Maurice, and J.-P. Tillich. Quantum LDPC codes obtained by non-binary constructions. In *Proc. IEEE Int. Symp. Info. Theo.*, pages 343–347, 2012.

[AMT12b] I. Andriyanova, D. Maurice, and J.-P. Tillich. Spatially coupled quantum LDPC codes. In *Information Theory Workshop (ITW), 2012 IEEE*, pages 327–331, Sept 2012.

[Aud13] Benjamin Audoux. An application of Khovanov homology to quantum codes. *CoRR*, abs/1307.4677, 2013.

[Bac06] Dave Bacon. Operator quantum error-correcting subsystems for self-correcting quantum memories. *Phys. Rev. A (Atomic, Molecular, and Optical Physics)*, 73(1) :012340, 2006.

[BB84] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE international Conference on Computers, Systems and Signal Processing, Bangalore, India*, page 175, New York, 1984. IEEE Press.

[BD03] L. Barnault and D. Declercq. Fast decoding algorithm for LDPC over $GF(2^q)$. In *Information Theory Workshop, 2003. Proceedings. 2003 IEEE*, pages 70–73, 2003.

[BDH06] T. A. Brun, I. Devetak, and M.-H. Hsieh. Correcting quantum errors with entanglement. *Science*, 314 :436–439, 2006.

[BDS97] Charles H. Bennett, David P. DiVincenzo, and John A. Smolin. Capacities of quantum erasure channels. *Phys.Rev.Lett.*, 78 :3217–3220, 1997.

[BH13] S. Bravyi and M. B. Hastings. Homological product codes. *ArXiv e-prints*, November 2013.

[BK98] Sergey B. Bravyi and Alexei Yu. Kitaev. Quantum codes on a lattice with boundary. 1998.

[BKLM02] David Burshtein, Michael Krivelevich, Simon Litsyn, and Gadi Miller. Upper bounds on the rate of LDPC codes. *IEEE Transactions on Information Theory*, 48(9) :2437–2449, 2002.

[BLT11] S. Bravyi, B. Leemhuis, and M. Terhal. Topological order in an exactly solvable 3D spin model. *Ann. Phys.*, 326 :4 :839, 2011.

[BM07] H. Bombin and M. A. Martin-Delgado. Exact topological quantum order in D=3 and beyond: Branyons and brane-net condensates. *Physical Review B*, 75(7) :075103, February 2007.

[BMD06] H. Bombin and M. A. Martin-Delgado. Topological quantum distillation. *Phys.Rev.Lett.*, 97 :180501, 2006.

[BMD07] H. Bombin and M. A. Martin-Delgado. Homological error correction: Classical and quantum codes. *J.Math.Phys.*, 48 :052105, 2007.

[Bom10] H. Bombin. Topological subsystem codes. *Phys. Rev. A*, 81 :032301, Mar 2010.

[CDZ11] A. Couvreur, N. Delfosse, and G. Zémor. A construction of quantum LDPC codes from Cayley graphs. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 643–647, 2011.

[COT07] T. Camara, H. Ollivier, and J.-P. Tillich. A class of quantum LDPC codes: construction and performances under iterative decoding. In *Proceedings of ISIT 2007*, pages 811–815, Nice, June 2007. IEEE.

[CRSS98] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction via codes over $GF(4)$. *IEEE Trans. Info. Theor.*, 44 :1369, 1998.

[CS96] A. R. Calderbank and P. W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54 :1098–1105, 1996.

[CT06] T. M. Cover and J. A. Thomas. *Elements of Information Theory, 2nd edition*. Hoboken, NJ: Wiley, 2006.

[DDJ06] D. Divsalar, S. Dolinar, and C. Jones. Construction of protograph LDPC codes with linear minimum distance. In *Information Theory, 2006 IEEE International Symposium on*, pages 664–668, July 2006.

[DJM11] David L. Donoho, Adel Javanmard, and Andrea Montanari. Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing. *CoRR*, abs/1112.0708, 2011.

[Djo08] I. B. Djordjevic. Quantum LDPC codes from balanced incomplete block designs. *IEEE Communication Letters*, 12(5) :389–391, May 2008.

[DM98] M. C. Davey and D. J. C. MacKay. Low density parity check codes over $GF(q)$. *IEEE Communications Letters*, 2 :165–167, June 1998.

[DZ10] N. Delfosse and G. Zémor. Quantum erasure-correcting codes and percolation on regular tilings of the hyperbolic plane. In *Information Theory Workshop (ITW), 2010 IEEE*, pages 1–5, 2010.

[Fey82] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7) :467–488, 1982.

[FML02] M. H. Freedman, D. A. Meyer, and F. Luo. $\mathbb{Z}_2$-systolic freedom and quantum codes. In *Mathematics of quantum computation*, Chapman & Hall/CRC, pages 287–320, Boca Raton, FL, 2002.

[FZ99] A. J. Felström and K. S. Zigangirov. Time-varying periodic convolutional codes with low-density parity-check matrix. *IEEE Trans. Info. Theor.*, 45(5) :2181–2190, September 1999.

[Gal62] R. G. Gallager. Low-density parity-check codes. *IRE Trans. Info. Theo.*, 8 :21, 1962.

[Gal68] R. G. Gallager. *Information theory and reliable communication*. John Wiley & Sons, 1968.

[GFL08] J. García-Frías and K. Liu. Design of near-optimum quantum error-correcting codes based on generator and parity-check matrices of LDGM codes. In *Proceedings of CISS*, pages 562–567, Princeton, March 2008.

[Gib] P. Giblin. *Graphs, Surfaces and Homology*. Cambridge University Press.

[GL13] L. Guth and A. Lubotzky. Quantum error-correcting codes and 4-dimensional arithmetic hyperbolic manifolds. *ArXiv e-prints*, October 2013.

[Got97] Daniel Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, 1997.

[Got13] D. Gottesman. What is the overhead required for fault-tolerant quantum Computation? *ArXiv e-prints*, October 2013.

[Haa11] Jeongwan Haah. Local stabilizer codes in three dimensions without string logical operators. *Phys. Rev. A*, 83 :042330, Apr 2011.

[HBD08] M.-H. Hsieh, T. A. Brun, and I. Devetak. Quantum quasi-cyclic low-density parity check codes, March 2008. `arXiv:0803.0100v1 [quant-ph]`.

[HEA05] X. Hu, E. Eleftheriou, and D. M. Arnold. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Trans. Info. Theor.*, 51(1) :386–398, January 2005.

[HI07] M. Hagiwara and H. Imai. Quantum quasi-cyclic LDPC codes. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 806–810, 2007.

[Hu02] X. Hu. *Low-delay low-complexity error-correcting codes on sparse graphs*. PhD thesis, EPFL, Lausanne, Switzerland, 2002.

[IPS+12] Aravind R. Iyengar, Marco Papaleo, Paul H. Siegel, Jack K. Wolf, Alessandro Vanelli-Coralli, and Giovanni Emanuele Corazza. Windowed decoding of protograph-based LDPC convolutional codes over erasure channels. *IEEE Transactions on Information Theory*, 58(4) :2303–2320, 2012.

[ISUW11] Aravind R. Iyengar, Paul H. Siegel, Rüdiger L. Urbanke, and Jack K. Wolf. Windowed decoding of spatially coupled codes. *CoRR*, abs/1106.0075, 2011.

[KHIK11] K. Kasai, M. Hagiwara, H. Imai, and Sakaniwa K. Quantum error correction beyond the bounded distance decoding limit. *IEEE Trans. Info. Theor.*, 2011. to appear, see also arXiv :1007.17782v2[cs.IT].

[Kim07] I. H. Kim. Quantum codes on Hurwitz surfaces. Master's thesis, MIT, 2007. available at `http://dspace.mit.edu/handle/1721.1/40917`.

[Kit03] A. Y. Kitaev. Fault-tolerant quantum computation by anyons. *Ann. Phys.*, 303 :2, 2003.

[KMRU10] Shrinivas Kudekar, Cyril Measson, Thomas J. Richardson, and Rüdiger L. Urbanke. Threshold saturation on BMS channels via spatial coupling. *CoRR*, abs/1004.3742, 2010.

[KMS+11] Florent Krzakala, Marc Mézard, François Sausset, Yifan Sun, and Lenka Zdeborová. Statistical physics-based reconstruction in compressed sensing. *CoRR*, abs/1109.4424, 2011.

[KP10] Shrinivas Kudekar and Henry D. Pfister. The effect of spatial coupling on compressive sensing. *CoRR*, abs/1010.6020, 2010.

[KP12a] A. A. Kovalev and L. P. Pryadko. Improved quantum hypergraph-product LDPC codes. *ArXiv e-prints*, February 2012.

[KP12b] A. A. Kovalev and L. P. Pryadko. Quantum "hyperbicycle" low-density parity check codes with finite rate. *ArXiv e-prints*, December 2012.

[KRU10] Shrinivas Kudekar, Tom Richardson, and Rüdiger L. Urbanke. Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC. In *ISIT*, pages 684–688. IEEE, 2010.

[KRU13] S. Kudekar, T. Richardson, and R. L. Urbanke. Spatially coupled ensembles universally achieve capacity under belief propagation. *Information Theory, IEEE Transactions on*, 59(12) :7761–7813, 2013.

[LF10] M. Lentmaier and G. P. Fettweis. On the thresholds of generalized LDPC convolutional codes based on protographs. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 709–713, June 2010.

[LF11] M. Lentmaier and G. Fettweis. Coupled LDPC Codes: complexity aspects of threshold saturation. In *Proc. of the IEEE Inform. Theory Workshop*, pages 668–672, 2011.

[LGF06] H. Lou and J. García-Frías. On the application of error-correcting codes with low-density generator matrix over different quantum channels. In *Proceedings of Turbo-coding 2006*, Munich, April 2006.

[Llo97] Seth Lloyd. Capacity of the noisy quantum channel. *Phys. Rev. A*, 55 :1613, 1997.

[LMFC10] M. Lentmaier, D.G.M. Mitchell, G. Fettweis, and D.J. Costello. Asymptotically good LDPC convolutional codes with AWGN channel thresholds close to the Shannon limit. In *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, pages 324–328, Sept 2010.

[LMSS01] Michael Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on Information Theory*, 47(2) :585–598, 2001.

[LSZC05] M. Lentmaier, A. Sridharan, K. S. Zigangirov, and D. J. Costello. Terminated LDPC convolutional codes with thresholds close to capacity. In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 1372–1376, Sept 2005.

[Mac99] D. J. C. MacKay. Good error correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2) :399–431, 1999.

[MMM04] D. J. C. MacKay, G. Mitchison, and P. L. MacFadden. Sparse graph codes for quantum error-correction. *IEEE Trans. Info. Theor.*, 50(10) :2315–2330, 2004.

[MN95] D. J. C. MacKay and R. M. Neal. Good codes based on very sparse matrices. In Colin Boyd, editor, *Cryptography and Coding. 5th IMA Conference*, number 1025 in Lecture Notes in Computer Science, pages 100–111. Springer, Berlin, 1995.

[MN97] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6) :457–458, Mar 1997.

[MPZC08] D. G. M. Mitchell, A. E. Pusane, K. Sh. Zigangirov, and D. J. Costello. Asymptotically good LDPC convolutional codes based on protographs. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 1030–1034, July 2008.

[MS78] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[MTA13] D. Maurice, J.-P. Tillich, and I. Andriyanova. A family of quantum codes with performances close to the hashing bound under iterative decoding. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 907–911, 2013.

[PC10] David Poulin and Yeojin Chung. On the iterative decoding of sparse quantum codes. *Quantum Information & Computation*, 8(10) :987–1000, 2010.

[PIS+10] M. Papaleo, A. R. Iyengar, P. H. Siegel, J. K. Wolf, and G. E. Corazza. Windowed erasure decoding of LDPC convolutional codes. In *Information Theory Workshop (ITW), 2010 IEEE*, pages 1–5, Jan 2010.

[Pre99] J. Preskill. Lecture notes for physics 229: quantum information and computation, 1999.

[PTO09] D. Poulin, J. Tillich, and H. Ollivier. Quantum serial turbo codes. *Information Theory, IEEE Transactions on*, 55(6) :2776–2798, June 2009.

[RDR12] J. M. Renes, F. Dupuis, and R. Renner. Efficient polar coding of quantum information. *Physical Review Letters*, 109(5) :50504, 2012.

[RU08] T. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[Sho95] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52 :2493, 1995.

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26 :1484–1509, 1997.

[SPVC06] R. Smarandache, A. E. Pusane, P. O. Vontobel, and D. J. Costello. Pseudo-codewords in LDPC convolutional codes. In *Information Theory, 2006 IEEE International Symposium on*, pages 1364–1368, July 2006.

[SPVC09] R. Smarandache, A. E. Pusane, P. O. Vontobel, and D. J. Costello. Pseudocodeword performance analysis for LDPC convolutional codes.

Information Theory, IEEE Transactions on, 55(6) :2577–2598, June 2009.

[SRK08] Pradeep Kiran Sarvepalli, Martin Rötteler, and Andreas Klappenecker. Asymmetric quantum LDPC codes. *CoRR*, abs/0804.4316, 2008.

[Ste96] A. M. Steane. Multiple particle interference and quantum error correction. *Proc. R. Soc. Lond. A*, 452 :2551–2577, 1996.

[Tan81] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Info. Theor.*, 27 :533–547, September 1981.

[TL10] P. Tan and J. Li. Efficient quantum stabilizer Codes: LDPC and LDPC-convolutional constructions. *IEEE Trans. Info. Theor.*, 56(1) :476–491, 2010.

[TZ09] J.-P. Tillich and G. Zémor. Quantum LDPC codes with positive rate and minimum distance proportional to $n^{\frac{1}{2}}$. In *Proceedings of ISIT 2009*, pages 799–803, July 2009.

[VDV+10] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard. Low-complexity decoding for non-binary LDPC codes in high order fields. *Communications, IEEE Transactions on*, 58(5) :1365–1375, 2010.

[WG13] Mark M. Wilde and Saikat Guha. Polar codes for classical-quantum channels. *IEEE Transactions on Information Theory*, 59(2) :1175–1187, 2013.

[WH11] M. M. Wilde and M.-H. Hsieh. Entanglement boosts quantum turbo codes. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 445–449, July 2011.

[WZ82] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886) :802–803, 1982.

[Zé09] Gilles Zémor. On Cayley graphs, surface codes, and the limits of homological coding for quantum error correction. In *IWCC*, volume 5557 of *Lecture Notes in Computer Science*, pages 259–273. Springer, 2009.