Décodeurs rapides pour codes topologiques quantiques

par

Guillaume Duclos-Cianci

mémoire présenté au département de physique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 8 juillet 2010

Your file Votre référence
ISBN: 978-0-494-65607-5
Our file Notre référence
ISBN: 978-0-494-65607-5

Le 13 juillet 2010

le jury a accepté le mémoire de Monsieur Guillaume Duclos-Cianci dans sa version finale.

Membres du jury

Professeur David Poulin
Directeur de recherche
Département de physique

Professeur Patrick Fournier
Membre
Département de physique

Professeur David Sénéchal
Président rapporteur
Département de physique

À mes parents et amis

Sommaire

Mots-clés : Information quantique ; Correction d'erreur quantique ; Ordre/Codes topologiques ; Décodeurs ; Propagation de croyance ; Renormalisation ; Anyons.

L'encodage topologique de l'information quantique a attiré beaucoup d'attention, car c'est un modèle qui semble propice à résister aux erreurs locales.

Tout d'abord, le modèle du calcul topologique est basé sur la statistique anyonique non-Abélienne universelle et sur son contrôle. Des anyons indésirables peuvent apparaître soudainement, en raison de fluctuations thermiques ou de processus virtuels. La présence de ces anyons peut corrompre l'information encodée, il est nécessaire de les éliminer : la correction consiste à fusionner les défauts tout en préservant la topologie du système.

Ensuite, dans le cas des codes topologiques, on doit aussi protéger l'information encodée dans la topologie. En effet, dans ces systèmes, on n'a accès qu'à une fraction de l'information décrivant l'erreur. Elle est recueillie par des mesures et peut être interprétée en termes de particules. Ces défauts peuplent le code et doivent être annihilés adéquatement dans le but de préserver l'information encodée.

Dans ce mémoire, nous proposons un algorithme efficace, appelé décodeur, pouvant être utilisé dans les deux contextes décrits ci-haut. Pour y parvenir, cet algorithme s'inspire de méthodes de renormalisation et de propagation de croyance. Il est exponentiellement plus rapide que les méthodes déjà existantes, étant de complexité $\mathcal{O}(\ell^2 \log \ell)$ en série et, si on parallélise, $\mathcal{O}(\log \ell)$ en temps, contre $\mathcal{O}(\ell^6)$ pour les autres décodeurs. Le temps étant le facteur limitant dans le problème du décodage, cette caractéristique est primordiale. De plus, il tolère une plus grande amplitude de bruit que les méthodes existantes ; il possède un seuil de $\sim 16.5\%$ sur le canal dépolarisant surpassant le seuil déjà établi de $\sim 15.5\%$. Finalement, il est plus versatile. En effet, en étant limité au code de Kitaev, on ne savait pas décoder les codes topologiques de manière générale (e.g. codes de couleur). Or, le décodeur proposé dans ce mémoire peut traiter la grande classe des codes topologiques stabiliseurs.

Remerciements

Tout d'abord, merci à ma famille qui m'a toujours soutenu et qui a été très influente dans ma vie. Merci de m'avoir inculqué mes valeurs qui me soutiennent et me permettent de persévérer dans mon travail.

Ensuite, merci à mes frères d'armes qui sauront se reconnaître et avec qui j'ai partagé les hauts et les bas du quotidien ainsi que beaucoup de moments intenses.

Finalement, merci à mon superviseur, David Poulin, qui me pousse sans cesse à me dépasser. Merci de m'avoir permis d'en apprendre et d'en comprendre autant. Merci de m'avoir donné l'opportunité de travailler sur un projet passionnant. Merci de m'avoir donné l'opportunité de voyager pour participer à plusieurs conférences, dont certaines de renommée internationale.

Merci à l'organisme Mathematics of Information Technology and Complex Systems (MITACS), au Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) et au Fonds québécois de la recherche sur la nature et les technologies (FQRNT) pour leur soutien financier. J'aimerais souligner plus particulièrement la contribution de MITACS qui a financé la quasi-totalité du projet.

Table des matières

Liste des tableaux

Liste des figures

Introduction

D'une part, on sait que le monde dans lequel nous vivons est fondamentalement décrit par les lois de la mécanique quantique. Celle-ci a permis d'étudier les états de la matière comme jamais cela n'avait été fait. C'est le cas par exemple des métaux et des semi-conducteurs dont la structure de bande, découlant de la théorie de la mécanique quantique, permet d'expliquer la conductivité électrique et thermique de ces matériaux.

D'autre part, on a su utiliser ces systèmes pour construire des appareils, les ordinateurs, permettant le traitement classique de l'information. L'information classique et son traitement correspondent à la notion intuitive que l'on peut en avoir, i.e. un message ayant un sens bien précis, pouvant être lu, transmis, copié, etc. La température qu'indique le thermomètre est un exemple d'information classique, tout comme le solde d'un compte en banque. Cette capacité à traiter l'information efficacement a révolutionné nos sociétés.

Viens alors la question de pousser cette idée jusqu'au bout : si la nature est fondamentalement quantique, ne devrions-nous pas être en mesure de concevoir des appareils permettant le traitement de l'information quantique ? Peut-on espérer observer des effets quantiques sur une échelle de distance macroscopique ou bien la mécanique quantique est-elle limitée au monde microscopique ?

Plusieurs problèmes se dressent devant cette quête du traitement quantique de l'information (TQI). Premièrement, la mécanique quantique interdit la copie d'information (*no cloning theorem* [10,34]), processus naturel dans le cas classique. Deuxièmement, on sait qu'en mécanique quantique, le simple fait d'observer un système en modifie l'état, i.e. le processus de mesure affecte l'information. Troisièmement, pour être en mesure de tolérer un certain nombre d'erreurs, inévitables dans le monde réel, on a discrétisé l'information classique en bits. En effet, on sait qu'on ne peut pas concevoir un ordinateur classique analogique, car il ne tolère aucune erreur. Or, il n'est pas évident, à priori, d'opérer une telle discrétisation dans le cas quantique. De manière similaire, l'information quan-

tique continue ne peut être prémunie d'un continuum d'erreurs, que l'on nomme dans ce contexte décohérence. Bref, nous sommes aux prises avec des systèmes excessivement fragiles que l'on ne peut copier ou encore mesurer sous peine d'en altérer le contenu.

Toutefois, tous ces obstacles peuvent être surmontés. C'est ce que prouve le théorème du calcul quantique robuste, qui démontre l'existence d'un seuil de tolérance au bruit [1] : Si les dispositifs composant un appareil de TQI ont une amplitude d'erreur en-deçà d'un certain seuil, on sait construire une architecture permettant d'obtenir un appareil de TQI arbitrairement près du cas idéal. En théorie, cela permet le TQI, mais en pratique, les exigences requises par le théorème sont parfois très difficiles à obtenir en laboratoire. Par exemple, la construction suggérée requiert des couplages réglables ou des mesures sur des composantes éloignées l'une de l'autre alors que certains dispositifs sont limités au couplage au plus proche voisin. Toutefois, certaines de ces exigences peuvent être relaxées, sans trop affecter le seuil [32].

Il existe plusieurs candidats participant à la course au TQI : ions piégés [30], réseaux optiques [23], circuits supra-conducteurs [9], points quantiques [21], résonance magnétique nucléaire (RMN) [20], etc. Sur ces propositions de systèmes physiques s'ajoute une panoplie de modèles équivalents de calcul : modèle du circuit [8], modèle basé sur la mesure (*measurement-based quantum computation*, MBQC) [17], modèle topologique [14], modèle adiabatique [2], etc. Tous ont leurs forces et leurs faiblesses et la plupart du temps, les modèles s'adaptent le mieux à un nombre restreint d'architectures physiques. Par exemple, l'optique quantique semble être un bon candidat pour le MBQC, tandis que les circuits supra-conducteurs semblent bien s'accorder au modèle du circuit.

Dans ce travail, nous introduisons un algorithme, un *décodeur*, assurant la robustesse des mémoires et des calculs topologiques quantiques. Il s'applique à priori à tous les modèles de calcul. En effet, il existe un lien naturel avec le calcul topologique, discuté à la section suivante, mais on peut aussi le voir purement comme une méthode de correction d'erreurs dans le modèle du circuit. Aussi, Raussendorf a démontré comment le MBQC peut être robuste (*fault-tolerant*) en le basant sur le code de Kitaev, un code topologique [27].

Systèmes anyoniques

Le modèle du calcul topologique est basé sur la statistique anyonique [26, 31] qu'on voit apparaître dans certains systèmes à deux dimensions en physique de la matière

condensée présentant de l'ordre topologique. Dans ce contexte, l'information est encodée dans l'état fondamental dégénéré du système. On peut opérer des transformations sur cet état en déplaçant des excitations (quasi-particules) sur le plan 2D formé par le système. On distingue les anyons abéliens des anyons non-abéliens.

Considérons dans un premier temps les anyons abéliens, plus simples. Un exemple de système donnant lieu à ces anyons abéliens est une mince couche d'un matériau supra-conducteur. Le niveau fondamental est décrit par un état cohérent de tous les électrons du système. Des quasi-particules peuplent le système : les paires de Cooper (deux électrons appariés en un boson) et les fluxons, vortex de flux magnétique Φ. En déplaçant adiabatiquement une paire autour d'un vortex, dans un parcours fermé, l'état du système acquiert une phase $e^{i2e\Phi}$ (la paire de Cooper a une charge $2e$). On reconnaît l'effet Aharonov-Bohm. Or, cet effet est un invariant topologique, i.e. si on déforme de manière continue le chemin parcouru par le paire de Cooper, la phase acquise reste inchangée. Pour retrouver la description en termes d'échanges d'anyons indiscernables, on doit considérer des particules composites : une paire de Cooper appariée à un fluxon. Pour montrer qu'il s'agit bien d'un anyon, on aura besoin de l'effet Aharonov-Bohm et du théorème de spin-statistique. En effet, si une particule composite subit une rotation de 2π, la paire de Cooper s'enroule une fois autour du fluxon et une phase est acquise par effet Aharonov-Bohm comme discuté ci-haut. Or, par le théorème de spin-statistique, on sait que si on échange deux telles particules, la même phase sera acquise. Comme cette phase vaut $e^{i2e\Phi}$, la particule composite (paire de Cooper-fluxon) est un anyon.

On voit tout de suite l'avantage d'un tel système pour le TQI. Si on sait, avec un minimum de précision, déplacer des paires de Cooper autour d'un vortex, on pourra faire acquérir à l'état du système une phase $e^{i2ne\Phi}$ après n tours. Le contrôle parfait du parcours n'est pas nécessaire. Si de petites erreurs locales se glissent dans le trajet, l'effet reste inchangé.

Les particules décrites ci-haut sont de type anyons abéliens, car le fait de les échanger ou de les faire circuler l'une autour de l'autre ne fait qu'ajouter une phase à l'état du système. Or, la multiplication par des phases est une opération commutative. Si on fait plusieurs échanges, l'ordre dans lequel on les fait n'a pas d'importance, ils commutent.

L'effet Hall quantique fractionnaire présente aussi un comportement similaire pour certains facteurs de remplissages. Pour $\nu = 1/3$, l'état fondamental du gaz 2D d'électrons possède des excitations de charge qui est une fraction de e. Ces excitations peuvent être *tressées* (i.e. échangées) pour faire acquérir une phase à l'état du système. Il s'agit en-

core d'anyons abéliens. Toutefois, pour $\nu = 5/2$, on croit que les anyons présents sont non-abéliens [4]. Dans ce cas, le fait de tresser des anyons résulte en une opération unitaire sur le fondamental (plus générale que la simple acquisition d'une phase). Or, le produit de plusieurs opérations unitaires ne commute pas nécessairement, d'où l'appellation d'anyons non-abéliens. Ensuite, viens la question de l'universalité des opérations disponibles. Si les opérations unitaires possibles par tressage génèrent $SU(2^k)$, alors le fondamental sera dégénéré 2^k fois et il sera possible de simuler de façon robuste n'importe qu'elle opération unitaire sur ce sous-espace et c'est ce qu'on entend par universalité des opérations. Ainsi, on voit apparaître la possibilité de TQI robuste aux erreurs locales. Le fondamental n'étant pas nécessairement dégénéré, il est nécessaire de percer des trous dans le système ou bien de le mettre sur un tore. Ceci procure au système une topologie non-triviale dont découle la dégénérescence du fondamental. Nous reviendrons sur ce point au chapitre 3. On ne croit pas que les anyons non-abéliens pour $\nu = 5/2$ soient universels. Toutefois, il existe des méthodes hybrides pour compléter l'ensemble de portes élémentaires disponibles pour qu'il le devienne [4].

Codes topologiques

Ci-haut nous avons introduit des exemples de systèmes présentants de l'ordre topologique permettant ainsi le calcul topologique à l'aide d'anyons non-abéliens. Toutefois, la présence d'une phase topologique dans le système n'est pas essentielle. On peut concevoir des codes topologiques pour des systèmes physiques tels les circuits supraconducteurs ou pour la RMN, alors que ces systèmes ne présentent pas de phase topologique. Dans ce cas, on emprunte des idées et des concepts au calcul topologique quant au fonctionnement du code. Par exemple, sur une surface 2D couverte par un réseau de spins et repliée pour former un tore, on peut encoder de l'information dans des opérateurs s'enroulant autour du tore, c'est-à-dire qu'ils sont topologiquement non-triviaux. Dans ces codes, ce sont les états considérés et les opérateurs pertinents qui sont liés à la topologie, mais le système physique lui-même et son hamiltonien n'ont rien à voir avec une phase topologique, la statistique anyonique n'intervient pas.

Correction et décodage

D'une part, lorsqu'on fait du calcul topologique à l'aide d'anyons, des erreurs locales se produisent, que l'on appelle défauts. En effet, des anyons indésirables peuvent apparaître soudainement, en raison de fluctuations thermiques. La présence de ces anyons peut corrompre l'information encodée, il est donc nécessaire de les éliminer. Si la densité de défauts est suffisamment faible, ceux-ci seront suffisamment localisés pour être corrigés. Dans ce contexte, la correction consiste à fusionner les défauts tout en préservant la topologie du système. Pour décider des corrections à appliquer, c'est-à-dire déterminer quelles particules seront fusionnées ensemble et, pour ce faire, quelles trajectoires celles-ci devront suivre, il est nécessaire de faire appel à un algorithme classique.

D'autre part, dans le cas des codes topologiques, on a aussi besoin d'un algorithme classique pour protéger l'information encodée dans la topologie du système. En effet, dans ces systèmes, on n'a accès qu'à une fraction de l'information décrivant l'erreur. Cette information est recueillie par des mesures. On peut interpréter le résultat de ces mesures en terme de particules. Ces défauts peuplent le code et ceux-ci doivent être annihilés adéquatement dans le but de préserver l'information encodée. Encore une fois, un algorithme classique est nécessaire pour décider des corrections à appliquer.

Dans ce mémoire, nous proposons un algorithme efficace pouvant être utilisé dans les deux contextes, c'est-à-dire pour décider des fusions de défauts à faire dans un calcul topologique ou bien pour protéger l'information contenue dans un code topologique. Pour y parvenir, cet algorithme s'inspire de méthodes de renormalisation et de propagation de croyance. Il est exponentiellement plus rapide que les méthodes déjà existantes. Le temps étant le facteur limitant dans le problème du décodage, cette caractéristique est primordiale. De plus, il est non seulement plus rapide, mais il tolère une plus grande amplitude de bruit que les méthodes existantes. Finalement, il est plus versatile. En effet, on était limité au code de Kitaev ; on ne savait pas décoder les codes topologiques de manière générale (e.g. codes de couleur [3]). Or, le décodeur proposé dans ce mémoire peut traiter la grande classe des codes topologiques stabilisateurs.

Nous ne nous attarderons pas dans le présent texte à une architecture physique particulière, mais nous étudierons le modèle topologique et les codes topologiques ainsi que leur résistance aux erreurs locales. Pour ce faire, nous introduirons d'abord le concept de code et puis de code stabilisateur. Le formalisme décrivant les codes stabilisateurs s'adapte directement pour décrire certains codes topologiques, qui nous intéresserons. De

plus, pour tester les idées introduites plus bas, on se concentrera sur le code de Kitaev, une instance particulière de code topologique.

Chapitre 1

Codes stabilisateurs

1.1 Qubit

Un qubit est un système physique décrit par un espace de Hilbert à deux dimensions. À un instant donné, l'état du qubit est représenté par un vecteur normé de cet espace, ou plus généralement par un opérateur positif de trace 1 agissant sur cet espace, appelé opérateur densité ou matrice densité. On note $\mathcal{H}_q(\cong \mathbb{C}^2)$ l'espace de Hilbert associé au qubit. On privilégie une base orthonormée à priori arbitraire que l'on notera $\{|0\rangle, |1\rangle\}$ et que l'on appellera base de calcul.

1.2 Groupe de Pauli

1.2.1 Groupe de Pauli à un qubit, $\mathcal{P}_1$

Dans le but d'étudier un ou plusieurs qubits, il est utile, comme on le verra plus tard, de connaître l'effet d'une base d'opérateurs hermitiques appliqués sur les états de ce ou ces qubits. Les opérateurs de Pauli X, Y et Z en plus de l'identité, que l'on note I, forment une telle base, c'est-à-dire que tout opérateur peut s'écrire comme une combinaison linéaire de I, X, Y et Z :

$$\begin{array}{lll} X|0\rangle = |1\rangle & Y|0\rangle = i|1\rangle & Z|0\rangle = |0\rangle \\ X|1\rangle = |0\rangle & Y|1\rangle = -i|0\rangle & Z|1\rangle = -|1\rangle \end{array} \tag{1.1}$$

L'opérateur X est parfois appelé opérateur *bit-flip* et l'opérateur Z, opérateur *phase-flip*. De plus, à l'aide de ces définitions, on peut montrer quelques propriétés intéressantes,

- $Y = iXZ$.
- Ils sont hermitiques : $X^\dagger = X, \quad Y^\dagger = Y, \quad Z^\dagger = Z$.
- Ils sont unitaires : $X^2 = Y^2 = Z^2 = I$.

On vérifie alors que $\{I, X, Y, Z\} \otimes \{-1, 1, -i, i\}$ forme un groupe pour la multiplication dont les générateurs sont i, X et Z :

$$\mathcal{P}_1 = \langle i, X, Z \rangle, \tag{1.2}$$

où la notation $\langle \ldots \rangle$ désigne le groupe généré par les éléments entre crochets.

Explicitons les valeurs et vecteurs propres de ces opérateurs pour en approfondir notre compréhension. Pour faciliter ce travail on définit les vecteurs suivants

$$|\pm\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle), \qquad |\pm i\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle \pm i\,|1\rangle). \tag{1.3}$$

On vérifie alors que

$$X|\pm\rangle = \pm|\pm\rangle, \qquad Y|\pm i\rangle = \pm|\pm i\rangle, \qquad Z|\epsilon\rangle = (-1)^\epsilon|\epsilon\rangle, \tag{1.4}$$

avec $\epsilon \in \{0, 1\}$. Il est à noter que les états propres de Z sont les vecteurs de la base de calcul. De plus, comme on s'y attend pour des opérateurs unitaires 2×2, les valeurs propres de X, Y et Z sont 1 et -1.

Pour compléter notre survol des opérateurs de Pauli à un qubit, étudions les relations de commutation de ces opérateurs. Pour ce faire, utilisons la base de calcul.

$$\begin{aligned} XZ|0\rangle &= |1\rangle & XZ|1\rangle &= -|0\rangle \\ ZX|0\rangle &= -|1\rangle & ZX|1\rangle &= |0\rangle \end{aligned} \tag{1.5}$$

On voit donc que X et Z anti-commutent, $\{X, Z\} = 0$. De manière analogue, on peut montrer que

$$\{X, Z\} = 0, \qquad \{X, Y\} = 0, \qquad \{Z, Y\} = 0. \tag{1.6}$$

Évidemment, l'identité, I, commute avec tous ces opérateurs, de même que chaque opérateur avec lui-même.

1.2.2 Groupe de Pauli à n qubits, $\mathcal{P}_n$

Le groupe de Pauli sur n qubits, $\mathcal{P}_n$, est une extension naturelle de $\mathcal{P}_1$ et forme une base des opérateurs hermitiques sur n qubits,

$$\mathcal{P}_n = \mathcal{P}_1^{\otimes n}. \tag{1.7}$$

Les opérateurs de ce groupe ont donc une structure diagonale par bloc, ce qui simplifie considérablement leur traitement numérique comme cela sera expliqué plus loin. Pour tout $P \in \mathcal{P}_n$, on a

$$P = \bigotimes_{i=1}^{n} p_i, \tag{1.8}$$

avec $p_i \in \mathcal{P}_1$.

D'une part, étant donné cette structure, on comprend que les vecteurs propres des éléments $P \in \mathcal{P}_n$ sont aussi des produits tensoriels des vecteurs propres des opérateurs à un qubit correspondants.

D'autre part, les valeurs propres associées à ces vecteurs propres sont les produits des valeurs propres des vecteurs individuels composants le vecteur total. Comme toutes ces valeurs sont -1 ou 1, leur produit est aussi -1 ou 1.

De plus, on déduit que $\mathcal{P}_n$ est aussi un groupe généré par l'ensemble des générateurs à un qubit, c'est-à-dire

$$\mathcal{P}_n = \langle \{i\} \cup \{X_j, Z_j : 0 \leq j \leq n\} \rangle. \tag{1.9}$$

où X_i (Z_i) est défini comme l'identité agissant sur tous les qubits sauf le i-ème auquel un X (Z) est appliqué.

Cette structure permet de déduire simplement les relations de commutation des éléments de $\mathcal{P}_n$. Tout d'abord, rappelons que toutes les paires d'opérateurs de Pauli à un qubit commutent ou anti-commutent. Pour trouver la relation de commutation d'une paire d'opérateurs de $\mathcal{P}_n$, on doit comparer une à une chaque paire d'opérateurs à un qubit. Si le nombre de paires qui anti-commutent est impair, alors les deux opérateurs anti-commutent. Autrement, ils commutent.

Illustrons par un exemple dans $\mathcal{P}_4$. Considérons les deux opérateurs suivants,

$$P = X_1 Z_2 I_3 X_4, \qquad Q = I_1 Y_2 Z_3 I_4. \tag{1.10}$$

On a que X_1 et I_1 commutent, Z_2 et Y_2 anti-commutent, I_3 et Z_3 commutent, X_4 et I_4 commutent. Il y a un nombre impair d'anti-commutations (une) ce qui implique que P et Q anti-commutent.

Soulignons cette propriété des opérateurs de $\mathcal{P}_n$ de soit commuter, soit anti-commuter, sans autre possibilité, car elle jouera un rôle important dans ce qui suit.

1.3 Groupe de Clifford

Étant donné que le groupe de Pauli à n qubits est défini comme une extension du groupe à un qubit, il est naturel que nous ayons donné un ensemble de générateurs *locaux*, c'est-à-dire agissant de manière non-triviale sur un seul qubit à la fois. Toutefois, nous ne sommes pas tenus de choisir un tel ensemble générateur. Pour caractériser un ensemble quelconque de générateurs, attardons-nous d'abord à la structure particulière des relations de commutation et d'anti-commutation de l'ensemble spécifié ci-haut.

$$\begin{array}{cccccc} Z_1 & Z_2 & \dots & Z_{n-1} & Z_n \\ X_1 & X_2 & \dots & X_{n-1} & X_n \end{array} \tag{1.11}$$

On remarque que pour toute paire d'opérateurs choisis parmi ces $2n$, il y a anti-commutation seulement s'ils sont disposés un au-dessus de l'autre. Sinon, ils commutent puisqu'ils agissent sur deux qubits différents. On appellera *ensemble générateur canonique* un ensemble ayant cette structure de relations de commutation.

On peut montrer que tout ensemble canonique de $2n$ opérateurs de Pauli à n qubits génère une base des opérateurs unitaires à $2n$ qubits. Cela peut être déduit d'une équivalence (modulo la phase) entre $\mathcal{P}_n$ et $\mathbb{Z}_2^{2n}$ muni du produit symplectique. En effet, tout élément $P \in \mathcal{P}_n$ (modulo la phase) peut être décomposé en un produit de générateurs : $P = \prod_{i=1}^{n} X_i^{a_i} Z_i^{b_i}$ où $\vec{a}$ et $\vec{b}$ sont des vecteurs binaires à n composantes. Donc, à chaque élément de $\mathcal{P}_n$ (modulo la phase) correspond un unique élément de la forme $(\vec{a}, \vec{b}) \in \mathbb{Z}_2^{2n}$. Par contre, la phase étant abandonnée, les relations de commutations des éléments deviennent toutes triviales. Pour les retrouver, il suffit de considérer le produit symplectique des vecteurs binaires de $\mathbb{Z}_2^{2n}$. Le produit symplectique, $*$, est défini de

la manière suivante :

$$(\vec{a},\vec{b}) * (\vec{c},\vec{d}) = \vec{a}\cdot\vec{d} + \vec{b}\cdot\vec{c}, \tag{1.12}$$

où le produit scalaire, $\cdot$, et l'addition sont binaires. Intuitivement, on comprend que ce produit calcule la parité du nombre de chevauchements d'opérateurs X et Z entre les deux éléments considérés pour ainsi retrouver la relation de commutation.

Comme nous le verrons plus loin en abordant les codes stabilisateurs, c'est cette possibilité de choisir des générateurs non-locaux qui permettra d'encoder des qubits *logiques* de manière délocalisée.

Le groupe de Clifford est l'ensemble des opérateurs unitaires permettant un tel changement de générateurs de $\mathcal{P}_n$. Formellement, le groupe de Clifford est le normalisateur de $\mathcal{P}_n$ dans $SU(2^n)$, c'est-à-dire tous les opérateurs unitaires, U, à n qubits tels que $U\mathcal{P}_nU^\dagger = \mathcal{P}_n$. Pour l'explorer, nous introduisons deux opérateurs à un qubit, la porte de Hadamard (H) et la porte de phase (S), et un opérateur à deux qubits, le contrôle-non (C-NOT) :

$$\begin{array}{llll} & & \text{C-NOT}\,|00\rangle = |00\rangle \\ H\,|0\rangle = |+\rangle & S\,|0\rangle = |0\rangle & \text{C-NOT}\,|01\rangle = |01\rangle \\ H\,|1\rangle = |-\rangle & S\,|1\rangle = i\,|1\rangle & \text{C-NOT}\,|10\rangle = |11\rangle \\ & & \text{C-NOT}\,|11\rangle = |10\rangle \end{array} \tag{1.13}$$

Il se trouve que ces trois portes appliquées à tout qubit (ou toute paire de qubits) génèrent à elles seules le groupe de Clifford à n qubits. Grâce à ces trois opérateurs, il est donc possible de passer d'un ensemble de générateurs locaux du groupe de Pauli à un ensemble générateur canonique quelconque.

Comme on l'a dit plus haut, les éléments du groupe de Pauli à n qubits sont diagonaux par bloc. Or, les éléments du groupe de Clifford, manipulant ces opérateurs de Pauli, n'ont pas à être spécifiés par des matrices $2^n \times 2^n$, mais plutôt par des matrices $2n \times 2n$. Ce résultat est indirectement montré par le théorème de Gottesman-Knill [22] : Un ordinateur quantique qui ne ferait intervenir que des opérateurs de Pauli et des opérateurs de Clifford peut être efficacement simulé par un ordinateur classique. Cette représentation réduite des opérateurs de Clifford simplifie considérablement le traitement numérique permettant de considérer des systèmes composés de milliers de qubits.

1.3.1 Exemple de base

Prenons le cas de cinq qubits, on a alors comme base locale

$$\begin{array}{ccccc} Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \\ X_1 & X_2 & X_3 & X_4 & X_5 \end{array} \tag{1.14}$$

Considérons maintenant une nouvelle base :

$$\begin{array}{ccccc} s_1 = X_1Z_2Z_3X_4 & s_2 = X_2Z_3Z_4X_5 & s_3 = X_1X_3Z_4Z_5 & s_4 = Z_1X_2X_4Z_5 & \overline{Z} = Z_1Z_2Z_3Z_4Z_5 \\ t_1 = X_1X_3X_4X_5 & t_2 = X_2X_3 & t_3 = X_1X_5 & t_4 = X_2X_3X_4X_5 & \overline{X} = X_1X_2X_3X_4X_5 \end{array} \tag{1.15}$$

Pour qu'il s'agisse bien d'une base, il suffit que tous les éléments commutent deux-à-deux exceptés ceux un au-dessus de l'autre. On vérifie facilement que les relations de commutation canoniques sont respectées. Les noms donnés aux opérateurs seront justifiés dans la section sur les codes stabilisateurs (section 1.5).

Ce changement de base est effectué par une transformation unitaire U obtenue par multiplication des trois portes de base décrites ci-haut : H, S et C-NOT. En conjuguant un opérateur de la première liste, c'est-à-dire en appliquant la transformation $O \rightarrow UOU^\dagger$, on obtient l'opérateur correspondant de la seconde liste. Par exemple $UZ_3U^\dagger = P_3$. Le circuit permettant l'implémentation de U est décrit dans [15].

Voici un exemple de base non-locale. La paire d'opérateurs de chacune des colonnes génère l'algèbre d'un spin effectif. Chacun de ces spins effectifs fait intervenir de manière non-triviale plusieurs qubits physiques à la fois. Il s'agit de modes collectifs des spins physiques localisés.

1.4 Code

Toute réalisation physique d'un qubit sera imparfaite. Le qubit physique aura un temps de vie fini, les opérations qu'on lui appliquera auront une précision finie, etc. Se pose alors la question suivante : Pourrions-nous utiliser plusieurs qubits imparfaits de manière à en retirer un plus petit nombre de qubits effectifs plus performants ? Voilà l'intuition derrière l'encodage.

Plus formellement, considérons un système physique quelconque. Un code, $\mathcal{C}$, est un sous-espace de son espace de Hilbert. Si la dimension du code est de la forme 2^k, alors

on dit qu'y sont encodés k qubits logiques.

Un parallèle intéressant existe avec la notion d'*ensemble complet d'observables qui commutent* (ECOC). En effet, en spécifiant les valeurs propres d'un ECOC, on peut décrire un état. Si on considère plutôt un EOC, en relâchant la contrainte qu'il soit complet, on spécifie un sous-espace, qui peut être interprété comme un code. Les valeurs propres de ces opérateurs sont appelées nombres quantiques. Le code est donc caractérisé par des nombres quantiques fixes, mais qui ne sont pas en nombre suffisant pour spécifier un état unique. Les codes stabilisateurs, sur lesquels nous nous attarderons, ont précisément cette structure, avec un EOC constitué d'opérateurs de Pauli.

Pour éviter toute confusion, notons que les codes stabilisateurs sont les analogues quantiques des codes linéaires classiques. Ces derniers consistent en un sous-espace vectoriel des chaînes de n bits possibles. En ce sens, les codes stabilisateurs constituent une généralisation des codes linéaires classiques où on peut considérer des superpositions de mots codes. Il ne faut pas confondre cette définition des codes classiques avec celle où on parle plutôt d'un ensemble d'instructions ou d'un programme.

1.5 Codes stabilisateurs

Les premiers codes quantiques ont fait leur apparition dans les travaux de Peter Shor, où a été introduit le code à neuf qubits [28], et de Andrew Steane, où a été introduit le code à sept qubits [29]. Après ces travaux pionniers, une théorie quantique des codes a pu émerger. Daniel Gottesman en a fait une synthèse importante dans sa thèse de doctorat en y présentant le formalisme très général des codes stabilisateurs, englobant plusieurs autres types de codes [15]. D'autres auteurs, comme John Preskill, ont par la suite écrit des ouvrages ou des notes de cours présentant ce formalisme [26].

Ceci étant dit, amorçons notre étude des codes stabilisateurs et pour ce faire, concentrons-nous sur des systèmes de n qubits et d'espace de Hilbert $\mathcal{H}_q^{\otimes n}$. Les codes stabilisateurs et leur formalisme permettent un découpage intuitif et élégant de l'espace de Hilbert.

On sait que chaque paire $\{X_i, Z_i\}$ définit un degré de liberté de qubit physique. Or, comme on l'a vu, grâce aux opérateurs de Clifford, on peut transformer cet ensemble canonique local en un nouvel ensemble canonique plus général. Dans un tel ensemble, chaque paire $\{P_i, Q_i\}$ définit un degré de liberté de spin collectif. Pour définir un code stabilisateur, on redécoupe l'espace de Hilbert à l'aide de ces nouveaux degrés de liberté collectifs dont un certain nombre seront fixés. Les degrés non-contraints contiendront

l'information quantique à protéger.

L'espace code qui en résulte est très délocalisé, c'est-à-dire qu'on ne peut pas passer d'un état d'un sous-espace à un autre du même sous-espace par le biais d'un opérateur local, i.e. agissant sur un petit nombre de qubits.

Dans les sous-sections suivantes, nous présentons le formalisme des codes stabilisateurs dans le but de construire un tel espace code.

1.5.1 Groupe stabilisateur

Choisissons un ensemble canonique générateur d'une base d'opérateurs à n qubits, $\{P_i, Q_i \in \mathcal{P}_n : 1 \leq i \leq n\}$ (l'équation 1.15 en est un exemple pour $n = 5$). Puis, définissons $s_i = P_i$ pour $1 \leq i \leq m$:

$$\begin{array}{cccccc} s_1 & \dots & s_m & P_{m+1} & \dots & P_n \\ Q_1 & \dots & Q_m & Q_{m+1} & \dots & Q_n \end{array} \tag{1.16}$$

Les $\{s_i\}$, étant indépendants et commutants deux à deux, génèrent un sous-groupe Abélien $\mathcal{S}$ de $\mathcal{P}_n$, que l'on nommera groupe stabilisateur,

$$\mathcal{S} = \langle \{s_i : 1 \leq i \leq m\} \rangle. \tag{1.17}$$

Notons que puisque les opérateurs de Pauli sont des observables et que puisque nous avons défini un sous-groupe Abélien, nous pouvons reconnaître que $\mathcal{S}$ est un EOC. De plus, rappelons que puisque $\mathcal{S}$ est un sous-groupe de $\mathcal{P}_n$, tous les opérateurs qu'il contient ont pour valeurs propres -1 et +1.

On définit le code stabilisateur, $\mathcal{C}_\mathcal{S}$, spécifié par $\mathcal{S}$ comme étant le sous-espace propre simultané de valeurs propres +1 de tous les $s \in \mathcal{S}$:

$$\mathcal{C}_\mathcal{S} = \{|\psi\rangle : s\,|\psi\rangle = |\psi\rangle \quad \forall s \in \mathcal{S}\}. \tag{1.18}$$

Formulation hamiltonienne

Étant donné la manière dont le sous-espace code est caractérisé dans le formalisme des codes stabilisateurs, il est aisé de donner une formulation hamiltonienne du code. Construisons un hamiltonien dont le fondamental est dégénéré et coïncidant avec le code. Pour ce faire, il suffit d'attribuer une pénalité en énergie à tout état qui n'est pas dans le

code. Utilisons directement les générateurs du stabilisateur. En effet, ceux-ci caractérisent déjà les états codes. Considérons le hamiltonien suivant,

$$H = -\sum_i s_i. \tag{1.19}$$

On vérifie que tout état du code a une énergie minimale de $-m$, alors que tout autre état propre simultané des s_i hors du code a une énergie d'au moins $-m+2$ puisqu'il violera au moins une des conditions $s_i |\psi\rangle = |\psi\rangle$. Ainsi, le code correspond à l'espace fondamental dégénéré de ce hamiltonien.

1.5.2 Groupe logique

Ci-haut, nous avons montré comment spécifier un sous-espace code. Toutefois, cela ne suffit pas. En effet, pour être en mesure de reconnaître l'information qui y est encodée, il est nécessaire de s'accorder sur une convention quant aux degrés de liberté à l'intérieur du code. À priori, n'importe quelle base du code fait l'affaire, mais une fois celle-ci fixée, il faut s'y tenir.

On définit $\overline{Z}_i = P_{m+i}$ et $\overline{X}_i = Q_{m+i}$ pour $0 \leq i \leq k$ (où $k = n - m$) :

$$\begin{array}{cccccc} s_1 & \dots & s_m & \overline{Z}_1 & \dots & \overline{Z}_k \\ Q_1 & \dots & Q_m & \overline{X}_1 & \dots & \overline{X}_k \end{array} \tag{1.20}$$

Rappelons que chaque colonne génère l'algèbre d'un qubit effectif. Ceux définis par $\{(\overline{Z}_i, \overline{X}_i)\}$ seront nos qubits logiques et c'est pourquoi on appelle les $\overline{Z}_i$ et $\overline{X}_i$ opérateurs logiques. De plus, on peut se convaincre qu'ils n'agissent qu'à l'intérieur du code en constatant qu'ils commutent avec les $\{s_i\}$. Un état du code sera donc transformé en un autre état du code sous l'action des opérateurs logiques.

Avec le concept d'opérateurs logiques introduit, on comprend facilement qu'on encode k qubits logiques à partir de n qubits physiques et que pour ce faire, on a besoin de $m = n - k$ générateurs du stabilisateur. Typiquement, si on a à travailler avec des états explicités, on les exprimera dans la base des états propres des $\overline{Z}_i$ (ou des $\overline{X}_i$ dans certains cas particuliers).

Le produit de deux opérateurs logiques est encore un opérateur logique. Ce sous-

ensemble de $\mathcal{P}_n$ génère donc un groupe, qu'on appelle groupe logique,

$$L = \langle \{ \overline{Z}_i, \overline{X}_i : 1 \leq i \leq k \} \rangle. \tag{1.21}$$

1.5.3 Groupe des erreurs pures

Pour terminer de spécifier nos générateurs, on définit les erreurs pures $t_i = Q_i$ pour $1 \leq i \leq m$:

$$\begin{array}{cccccc} s_1 & \dots & s_m & \overline{Z}_1 & \dots & \overline{Z}_k \\ t_1 & \dots & t_m & \overline{X}_1 & \dots & \overline{X}_k \end{array} \tag{1.22}$$

Encore une fois, chaque colonne génère l'algèbre d'un qubit effectif. Ceux définis par $\{(s_i, t_i)\}$ sont nos qubits stabilisateurs. Les $\{t_i\}$ génèrent aussi un groupe que l'on appellera groupe des erreurs pures,

$$T = \langle \{ t_i : 1 \leq i \leq m \} \rangle. \tag{1.23}$$

On les appelle ainsi, car ces opérateurs n'affectent pas l'information à proprement parler. Ils ne font que déplacer le code, c'est-à-dire prendre le sous-espace en bloc et le transformer en un autre isomorphe, mais possédant des nombres quantiques différents. Dans la mesure où ces opérateurs n'agissent pas à l'intérieur des sous-espaces, on les dit purs. On explorera en détail cet aspect au chapitre suivant. D'une part, comme les $\{t_i\}$ commutent avec les opérateurs logiques, on confirme qu'ils n'ont pas d'impact à l'intérieur du code. D'autre part, comme ils anti-commutent avec les $\{s_i\}$, ils changeront certaines des valeurs propres $+1$ du code vis-à-vis des $\{s_i\}$ en -1 et donc sortent l'état du code.

1.5.4 Exemples

Code à trois qubits

Pour illustrer les concepts introduits ci-haut, étudions un exemple simple : le code à trois qubits. Considérons donc un système de trois qubits et la base suivante

$$\begin{array}{ccc} Z_1Z_2 & Z_2Z_3 & Z_1Z_2Z_3 \\ X_2X_3 & X_1X_2 & X_1X_2X_3 \end{array} \tag{1.24}$$

On vérifie facilement qu'il s'agit bien d'une base canonique. On a donc les générateurs du stabilisateur suivants,

$$s_1 = Z_1 Z_2, \quad s_2 = Z_2 Z_3. \tag{1.25}$$

Le code est le sous-espace propre $+1$ simultané de ces deux générateurs. Pour l'identifier, étudions l'effet de s_1 et s_2 sur les états propres de $\overline{Z}_1 = Z_1 Z_2 Z_3 : \{|\delta\epsilon\sigma\rangle : \delta, \epsilon, \sigma \in \{0, 1\}\}$. Tout d'abord, ils sont tous des états propres de s_1 et s_2. Reste à déterminer leur valeur propre,

$$\begin{aligned} s_1 |\delta\epsilon\sigma\rangle &= Z_1 Z_2 |\delta\epsilon\sigma\rangle = (-1)^{\delta+\epsilon} |\delta\epsilon\sigma\rangle \\ s_2 |\delta\epsilon\sigma\rangle &= Z_2 Z_3 |\delta\epsilon\sigma\rangle = (-1)^{\epsilon+\sigma} |\delta\epsilon\sigma\rangle \end{aligned} \tag{1.26}$$

Alors, on doit avoir $\delta = \epsilon$ et $\epsilon = \sigma$, ce qui nous laisse deux *mots codes* : $|\overline{0}\rangle = |000\rangle$ et $|\overline{1}\rangle = |111\rangle$. Le code est le sous-espace généré par ces deux états,

$$\mathcal{C} = \{\alpha |000\rangle + \beta |111\rangle : |\alpha|^2 + |\beta|^2 = 1\}. \tag{1.27}$$

De plus, on vérifie facilement que les opérateurs logiques agissent sur ces mots codes comme les opérateurs de $\mathcal{P}_1$ sur les états d'un qubit :

$$\begin{aligned} \overline{X} |\overline{0}\rangle &= |\overline{1}\rangle \qquad \overline{Z} |\overline{0}\rangle = |\overline{0}\rangle \\ \overline{X} |\overline{1}\rangle &= |\overline{0}\rangle \qquad \overline{Z} |\overline{1}\rangle = -|\overline{1}\rangle \end{aligned} \tag{1.28}$$

Chapitre 2

Problème du décodage

Dans le chapitre précédent, nous avons vu qu'il était possible d'utiliser l'encodage pour tirer quelques qubits performants de plusieurs qubits imparfaits. Pour vérifier que ce procédé nous conduit bien à une amélioration et surtout dans le but de mesurer celle-ci, nous analyserons l'impact des erreurs de Pauli sur le code. Rappelons que les opérateurs de Pauli forment une base pour tout opérateur, il s'agit donc d'un point de départ justifié.

2.1 Code à trois qubits

Avant d'entrer dans le vif du sujet, arrêtons-nous au code à trois qubits qui est un exemple simple de code, grâce auquel on peut déjà introduire certains concepts de base de la correction d'erreur. Supposons qu'un état $|\psi\rangle = \alpha|000\rangle + \beta|111\rangle$ soit encodé à l'aide du code à trois qubits et qu'une erreur X ce soit produite sur le premier qubit. L'état devient alors $|\psi\rangle = \alpha|100\rangle + \beta|011\rangle$. Évidemment, nous ne savons pas à priori qu'une telle erreur s'est produite.

Pour corriger l'erreur, il faut trouver un moyen d'identifier le qubit ayant subi une inversion. Or, on ne peut mesurer les qubits un à un dans la base de calcul, car cela aurait pour effet de détruire toute superposition. À la suite d'une telle mesure, on se retrouverait avec soit $|\psi\rangle = |100\rangle$, soit $|\psi\rangle = |011\rangle$, deux états à toutes fins pratiques classiques et les amplitudes, α et β, seraient détruites.

Pour permettre à l'état de demeurer intact malgré la mesure, il faut que, et $|000\rangle$ et $|111\rangle$ soient des états propres de même valeur propre de l'opérateur de mesure. C'est le cas par exemple de l'opérateur $s_1 = Z_1Z_2$, un stabilisateur du code. En effet, on sait que

nos deux mots codes sont des vecteurs propres de valeur propre +1 de s_1 :

$$Z_1Z_2\,|000\rangle = |000\rangle\,, \qquad Z_1Z_2\,|111\rangle = |111\rangle\,. \tag{2.1}$$

Remarquons qu'il en va de même pour $|100\rangle$ et $|011\rangle$ excepté que ceux-ci sont de valeur propre -1. En fait, Z_1Z_2 évalue la parité des deux premiers qubits : s'ils sont identiques, la valeur propre sera +1 et s'ils sont différents, la valeur propre sera -1. On comprend alors qu'obtenir -1 lors de la mesure de Z_1Z_2 nous apprend qu'un des deux qubits a subi une erreur d'inversion sans rien nous apprendre sur les qubits individuels, protégeant ainsi une superposition d'états.

De manière analogue, la mesure du stabilisateur $s_2 = Z_2Z_3$ nous renseigne sur la parité des deux derniers qubits sans nous renseigner sur les qubits individuels. Dans l'exemple décrit ci-haut, on pourrait mesurer s_1 et s_2 et obtenir les valeurs -1 et +1. On pourrait alors conclure que les deux premiers qubits sont opposés, alors que les deux derniers sont alignés. Deux erreurs sont plausibles : soit le premier qubit a été renversé, soit les deux derniers qubits l'ont été. On écrira ces deux erreurs :

$$e_1 = X_1, \qquad e_2 = X_2X_3. \tag{2.2}$$

En supposant un bruit indépendant et local, il est plus probable que seulement un qubit ait été renversé, donc le premier. Si le troisième qubit avait été renversé au lieu du premier, on aurait obtenu les valeurs +1 et -1 lors de la mesure de s_1 et s_2 respectivement. Si c'était le deuxième qubit, on aurait -1 et -1. Les trois erreurs d'inversion à un qubit donnent trois résultats de mesure différents. Soulignons le rôle crucial que jouent les générateurs du stabilisateur pour diagnostiquer une erreur.

Remarquons que les deux erreurs plausibles sont reliées par un opérateur logique (éq. 2.3). De plus, leur expression dans la base délocalisée nous apprend que toutes deux contiennent t_1 (éq. 1.15 et éq. 2.4), ce qui n'est pas fortuit. Nous reviendrons sur ces points à la section 2.2.

$$e_1e_2 = X_1X_2X_3 = \overline{X} \tag{2.3}$$

$$e_1 = t_1\overline{X} \qquad e_2 = t_1 \tag{2.4}$$

L'encodage a pour effet d'augmenter la résistance au bruit : le qubit seul ne peut tolérer aucune erreur X, alors que le qubit encodé peut en subir une et être corrigé. En

conséquence, le code à trois qubits utilisé sur un canal où chaque qubit a une probabilité p d'inversion produit un qubit logique dont la probabilité d'inversion est $\mathcal{O}(p^2)$ (terme dominant, avec préfacteur négligé). Toutefois, le code à trois qubits n'est pas « complet », car il ne détecte pas les erreurs Z. En effet, celles-ci commutent avec s_1 et s_2 et passent donc inaperçues.

2.1.1 Codes concaténés

On a vu que l'on peut construire des codes de n qubits physiques encodant k qubits logiques. Comme on l'a dit à la section précédente, l'encodage permet de diminuer l'amplitude du bruit sur l'information, sans toutefois le supprimer complètement. Il serait intéressant de concevoir une procédure permettant d'utiliser plusieurs codes donnés pour construire une famille de codes diminuant à un niveau arbitrairement bas l'amplitude du bruit. C'est ce que permet la concaténation [19]. Nous nous attardons à un cas simple de concaténation, quoiqu'il puisse être directement généralisé : on construit une famille de codes utilisant n^l qubits encodant 1 qubit en utilisant un code à n qubits encodant 1 qubit.

Considérons d'abord un code, $\mathcal{C}$, de n qubits encodant 1 qubit défini par un choix de générateurs donné :

$$\begin{array}{cccc} s_1 & \dots & s_{n-1} & \overline{Z}_1 \\ t_1 & \dots & t_{n-1} & \overline{X}_1 \end{array} \tag{2.5}$$

La procédure est la suivante. On prend un qubit et on lui accole $n-1$ qubits supplémentaires. Puis, on effectue un premier niveau d'encodage pour obtenir une instance de $\mathcal{C}$. Jusque là, il s'agit de la procédure d'encodage habituelle. Ensuite, on accole $n-1$ qubits à chacun des n qubits déjà utilisés. On encode alors chacun de ceux-ci en utilisant le même code que précédemment (dans le cas général, il n'est pas nécessaire qu'il s'agisse du même code). On a alors n^2 qubits encodant 1 qubit. On peut répéter cette manoeuvre de manière itérative. Après l niveaux d'encodages, on a n^l qubits encodant 1 qubit.

Pour bien comprendre ce qui se passe, il est nécessaire d'analyser comment la base évolue d'une étape de concaténation à l'autre. Pour faciliter la discussion, rabattons-nous sur le code à trois qubits et concentrons-nous sur ce cas particulier. Au départ, on a un qubit à encoder. Celui-ci est caractérisé par la base d'opérateur $\{X, Z\}$. Après le

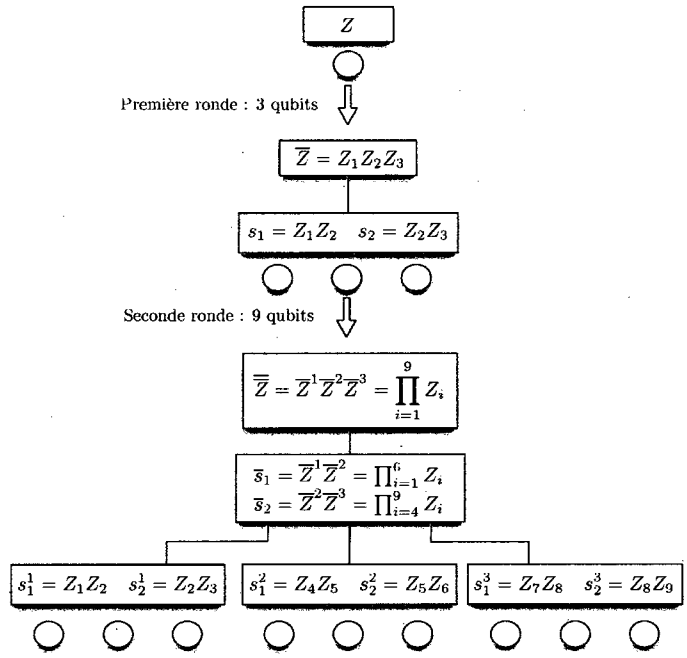


FIGURE 2.1 – Codes concaténés : Ce diagramme illustre l'évolution des stabilisateurs et de l'opérateur logique Z au fil des niveaux d'encodages. Une analyse analogue peut être faite pour les erreurs pures et l'opérateur logique X. Les cercles représentent les qubits et les boîtes les différents niveaux d'encodages.

premier niveau d'encodage, on obtient la base d'opérateurs introduite à la section 1.5.4. Remarquons que l'encodage a pour effet de transformer X et Z en $\overline{X}$ et $\overline{Z}$ respectivement. Le second niveau d'encodage aura alors pour effet de transformer chacun des X_i et Z_i en $\overline{X_i}$ et $\overline{Z_i}$. Chacun des s_i sera lui-même encodé, c'est-à-dire qu'il deviendra un $\overline{s}_i$, e.g. $\overline{s}_1 = \overline{Z}^1 \overline{Z}^2$, où l'exposant des opérateurs fait maintenant référence au code plutôt qu'au qubit. Le qubit logique sera doublement encodé, $\overline{\overline{Z}} = \overline{Z}^1 \overline{Z}^2 \overline{Z}^3$. La figure 2.1 illustre l'effet de deux niveaux d'encodages.

On a vu que le code à trois qubits utilisé sur un canal où chaque qubit a une probabilité p d'inversion produit un qubit logique dont la probabilité d'inversion est $\mathcal{O}(p^2)$. Si on répète la concaténation j fois, on trouve plutôt un taux d'erreurs $\mathcal{O}(p^{2^j})$, c'est-à-dire doublement exponentiel dans le nombre de niveaux de concaténation.

2.2 Erreurs et syndromes

Ci-haut, nous avons donné un exemple et présenté intuitivement plusieurs concepts à la base de la correction d'erreur quantique. Dans cette section et les suivantes, nous formalisons davantage ces intuitions en traitant un cas général.

2.2.1 Erreurs de Pauli

Considérons un code stabilisateur, $\mathcal{C}$, encodant un état $|\psi\rangle$ quelconque et une base de $\mathcal{P}_n$ le caractérisant :

$$\begin{array}{cccccccc} s_1 & \dots & s_m & \overline{Z}_1 & \dots & \overline{Z}_k \\ t_1 & \dots & t_m & \overline{X}_1 & \dots & \overline{X}_k \end{array} \tag{2.6}$$

Puis, considérons un sous-ensemble, E, de $\mathcal{P}_n$, qui contiendra les erreurs susceptibles d'affecter le code. Alors pour toute erreur e de E, on peut trouver une décomposition dans la nouvelle base,

$$e = tls \qquad t \in T, l \in L, s \in \mathcal{S}, \tag{2.7}$$

avec $t = \prod_{i=1}^{n} t_i^{a_i}$ où $a_i \in \{0, 1\}$. Alors, $\forall s_i$

$$e\,|\psi\rangle = es_i\,|\psi\rangle = (tls)s_i\,|\psi\rangle = (-1)^{a_i} s_i e\,|\psi\rangle\,. \tag{2.8}$$

La première égalité vient du fait que $|\psi\rangle$ appartient au code et est donc état propre $+1$ des s_i. La seconde utilise la décomposition de e dans la nouvelle base. La troisième utilise les relations de commutations des $\{s_i\}$ et des $\{t_i\}$. Comme on s'y attend, puisqu'il s'agit d'opérateurs de Pauli, s_i et e commutent ou anticommutent.

On en déduit deux propriétés importantes des relations de commutations entre e et les s_i :

- Elles spécifient la composante $t \in T$ de e.
- Elles sont encodées dans les valeurs propres des opérateurs s_i associées à l'état propre $e\,|\psi\rangle$.

On peut alors accéder à l'ensemble de ces valeurs propres en mesurant l'ensemble des générateurs du groupe stabilisateur et ainsi connaître la composante t de e. C'est pourquoi les deux corrections plausibles de l'exemple présenté avec le code à trois qubits

contenaient toutes deux t_1 dans leur décomposition. Cette série de valeurs est appelée syndrome de l'erreur e. Il s'agit de la signature laissée par son action sur le code et c'est toute l'information mise à notre disposition pour l'identifier. De plus, il est important de noter que puisque $e|\psi\rangle$ est un état propre des s_i, il ne sera pas déformé par la mesure et l'information encodée dans $|\psi\rangle$ ne sera pas affectée davantage lors de la lecture du syndrome.

Notons que puisque les erreurs sont dans $\mathcal{P}_n$, elles sont unitaires et hermitiques, donc leur propre inverse. Si une erreur est identifiée, il suffit de l'appliquer à nouveau pour la corriger.

Nous possédons maintenant tous les ingrédients nécessaires pour décider si on peut corriger un ensemble d'erreurs E. Trois scénarios sont envisageables.

1. Si deux erreurs e_1 et e_2 sont telles que $e_1e_2 = s$, où $s \in \mathcal{S}$, alors elles ont le même syndrome, mais il n'est pas nécessaire de les distinguer. En effet, pour que $e_1e_2 = s$, on doit avoir $e_1 = tls'$ et $e_2 = tls''$ avec $s's'' = s$. Alors, on peut corriger une erreur avec l'autre et vice-versa, c'est-à-dire $e_1e_2|\psi\rangle = e_2e_1|\psi\rangle = s|\psi\rangle = |\psi\rangle = e_1^2|\psi\rangle = e_2^2|\psi\rangle$. Il n'y a donc aucun problème à ce qu'elles aient le même syndrome. Ces erreurs sont dites *dégénérées.*
2. Si deux erreurs e_1 et e_2 sont telles que $e_1e_2 \in L\mathcal{S}$ et $e_1e_2 \notin \mathcal{S}$, alors elles auront le même syndrome, mais il serait nécessaire de savoir les distinguer pour connaître la correction à appliquer. En effet, si le système subit l'erreur e_1, qui est indiscernable de e_2 par ses syndromes, et que nous tentons de corriger cette erreur en appliquant la transformation e_2, il en résultera la transformation $e_2e_1 = ls$ avec $l \neq I$, un changement de l'état logique du système. Si une telle paire d'erreurs existe, on ne pourra pas corriger E.
3. Si deux erreurs e_1 et e_2 sont telles que $e_1e_2 \notin L\mathcal{S}$, alors elles auront des syndromes distincts et on saura les identifier.

Intuitivement, on peut résumer en disant qu'il est nécessaire et suffisant que toutes les erreurs ayant un effet logique différent sur le code possèdent des syndromes distincts afin de pouvoir corriger un ensemble d'erreurs E.

2.2.2 Cas général

Jusqu'ici, nous avons supposé que les erreurs affectant le code étaient des opérateurs de Pauli. Or, ces erreurs ne sont pas les plus générales. En effet, on sait que les erreurs

possibles forment un continuum d'opérateurs. Toutefois, notons que toute erreur peut être décomposée dans la base des opérateurs de Pauli. Pour l'instant, supposons que cette somme soit limitée à E, un ensemble d'erreurs corrigibles par le code. Étudions alors le cas d'une erreur générale g ; générale dans le sens où elle fait partie de l'algèbre générée par le groupe d'erreurs de Pauli E. On peut écrire

$$g = \sum_i \alpha_i e_i, \tag{2.9}$$

où $\alpha_i \in \mathbb{C}$ et $e_i \in E$. Récrivons g en groupant ses composantes par syndrome,

$$g = \sum_t \lambda_t g'_t, \tag{2.10}$$

où les g'_t regroupent toutes les erreurs partageant le même syndrome t de telle sorte que chacun des g'_t a un syndrome différent. De plus, toutes les erreurs entrant dans la composition d'un g'_t sont équivalentes puisqu'on a supposé E corrigible (i.e. $e_i e_j \in \mathcal{S}$).

Supposons que g agissent sur notre système où est encodé un état $|\psi\rangle$,

$$g\,|\psi\rangle = \sum_t \lambda_t g'_t\,|\psi\rangle\,. \tag{2.11}$$

Pour acquérir de l'information sur g, on mesure son syndrome, c'est-à-dire qu'on mesure chacun des générateurs du stabilisateur s_i. Or, d'après le postulat de la mesure projective, puisque chacun des $g'_t\,|\psi\rangle$ appartient à un sous-espace propre différent, avec probabilité $|\lambda_t|^2$ l'état après la mesure sera l'état initial projeté sur l'unique $g'_t\,|\psi\rangle$ dont le syndrome correspond aux valeurs mesurées. On assiste donc à une discrétisation des erreurs par la mesure du syndrome. Il suffit alors de savoir corriger chacune des éventualités, c'est-à-dire chacun des g'_t, pour corriger g.

Bref, si on sait corriger un ensemble discret d'erreurs de Pauli, on sait aussi corriger toute l'algèbre qu'il génère. Ce résultat est crucial, car à priori, il semble impossible de corriger un continuum d'erreurs. Toutefois, grâce à la mesure du syndrome, on peut arriver à discrétiser, au moins en partie, cet ensemble continu, permettant ainsi la correction d'erreur quantique.

2.2.3 Code à cinq qubits

Considérons un système de cinq qubits [22] et la base suivante introduite à la section 1.3.1 (éq. 1.15),

$$\begin{array}{lllll} s_1 = X_1Z_2Z_3X_4 & s_2 = X_2Z_3Z_4X_5 & s_3 = X_1X_3Z_4Z_5 & s_4 = Z_1X_2X_4Z_5 & \overline{Z}_1 = Z_1Z_2Z_3Z_4Z_5 \\ t_1 = X_1X_3X_4X_5 & t_2 = X_2X_3 & t_3 = X_1X_5 & t_4 = X_2X_3X_4X_5 & \overline{X}_1 = X_1X_2X_3X_4X_5 \end{array} \tag{2.12}$$

Notons que les s_i sont générés par permutation cyclique de s_1. Le code est le sous-espace propre +1 simultané de ces générateurs. Nous n'en expliciterons pas les mots codes et en fait, ce n'est jamais nécessaire de le faire.

Supposons que notre système soit soumis à un ensemble d'erreurs simples, mais réalistes en première approximation : l'ensemble des opérateurs à un qubit. Comme nous l'avons vu, pour corriger cet ensemble continu d'erreurs, il suffit de savoir corriger une base de l'algèbre des opérateurs à un qubit, c'est-à-dire dans ce cas les opérateurs de Pauli à un qubit : I,X_i,Y_i et Z_i (pour chacun des cinq qubits). On notera cet ensemble E :

$$E = \{I, X_i, Y_i, Z_i : 1 \leq i \leq 5\} \tag{2.13}$$

Commençons par étudier les erreurs sur le premier qubit. I commute trivialement avec tous les s_i et a pour syndrome (1,1,1,1). X_1 anticommute avec s_4 seulement, son syndrome est (1,1,1,-1). De manière analogue, on trouve pour Y_1, (-1,1,-1,-1) et pour Z_1, (-1,1,-1,1). Étant donné que les s_i sont reliés par permutation cyclique, on obtient immédiatement :

I	i	X	Y	Z
(1,1,1,1)	1	(1,1,1,-1)	(-1,1,-1,-1)	(-1,1,-1,1)
	2	(-1,1,1,1)	(-1,-1,1,-1)	(1,-1,1,-1)
	3	(-1,-1,1,1)	(-1,-1,-1,1)	(1,1,-1,1)
	4	(1,-1,-1,1)	(-1,-1,-1,-1)	(-1,1,1,-1)
	5	(1,1,-1,-1)	(1,-1,-1,-1)	(1,-1,1,1)

TABLEAU 2.1 – Syndrome des erreurs à un qubit pour le code à cinq qubits

On conclut que pour le code à cinq qubits, toute erreur à un qubit est corrigible, car chaque erreur devant être distinguée possède un syndrome différent : on a 16 erreurs et

16 syndromes possibles. En effet, on a l'identité plus trois erreurs par qubit à distinguer ($1 + 3 \times 5 = 16$) et on a quatre générateurs du stabilisateur disponibles pour y parvenir ($2^4 = 16$ syndromes). Il n'y a donc pas de dégénérescence et cela s'explique par le fait que le groupe stabilisateur n'a pas d'élément de poids plus petit que trois. En ce sens, le code à cinq qubits est parfait et ce cas est exceptionnel.

Une façon de caractériser l'amélioration obtenue par l'encodage, pour des taux de bruits suffisament faibles, est de considérer l'ordre dominant du bruit. Supposons qu'un qubit a une probabilité p de subir une erreur donnée ($p \ll 1$, pour justifier l'analyse à l'ordre dominant). De plus, supposons qu'une erreur agissant sur k qubits a une probabilité $\mathcal{O}(p^k)$ de se produire. Un qubit physique aura un taux d'erreur $\mathcal{O}(p)$, alors qu'après encodage avec le code à cinq qubits, le taux d'erreur du qubit logique sera de $\mathcal{O}(p^2)$ puisqu'on sait corriger toute erreur à un qubit. Ainsi, on obtient une amélioration de $\mathcal{O}(p)$ du qubit effectif par rapport au qubit physique.

2.3 Modèles de bruit

Dans la section précédente, nous avons vu l'importance de savoir corriger les erreurs de Pauli, car dans le cadre des codes stabilisateurs, toute erreur est corrigible si et seulement si les opérateurs de Pauli qui la composent le sont eux-mêmes. C'est pourquoi nous soumettrons les codes à des canaux de Pauli pour les étudier [22].

Toutefois, nous ne procéderons pas exactement comme ci-haut. En effet, plutôt que de sélectionner un sous-ensemble d'erreurs possibles, nous attribuerons une probabilité à tout opérateur de Pauli. Une correction parfaite devient alors impossible. Nous utiliserons un nouveau concept pour caractériser les codes et leurs décodeurs : le seuil de tolérance au bruit. Il s'agit du taux de bruit à partir duquel il devient utile d'encoder un qubit physique. Intuitivement, on pourrait dire qu'il s'agit du point où le temps de vie du qubit logique devient supérieur à celui des qubits physiques utilisés dans l'encodage.

Pour une famille de codes de taille variable, ce seuil est analogue à une transition de phase ordonnée-désordonnée en physique de la matière condensée. Nous verrons que lorsque le taux de bruit est supérieur au seuil, alors la probabilité d'erreur du décodeur tendra vers 1 dans la limite des codes de taille infinie alors qu'elle tendra vers 0 si le taux de bruit est inférieur au seuil.

Les distributions de probabilité que nous utiliserons seront assujetties à une hypothèse d'indépendance du bruit. Chaque erreur à un qubit aura une certaine probabilité de se

produire et la probabilité d'une erreur agissant sur plusieurs qubits à la fois sera donnée par le produit des probabilités individuelles.

Bien que nos simulations soient effectuées sur ce modèle spécifique, il est fort concevable que les résultats qualitatifs s'appliquent à un vaste éventail de modèles de bruit, en autant que l'amplitude des erreurs à plusieurs qubits ait un comportement similaire, en particulier une suppression exponentielle.

Avant de donner des exemples concrets de modèles de bruit, il est important de spécifier ce qu'on entend exactement par « canal ». D'une part, ces transformations agissent sur des opérateurs densités plutôt que des états pures. On note ρ l'opérateur densité décrivant l'état d'un système. D'autre part, les canaux considérés plus bas sont des transformations complètement positives présevant la trace. Il s'agit du type de transformation le plus général concevable si l'on veut qu'un opérateur densité en soit encore un après une application du canal.

2.3.1 Canal d'inversion de bit/de phase

En théorie de l'information classique, le modèle de bruit le plus simple est canal d'inversion de bit symétrique. Avec une probabilité p, le bit est inversé et avec probabilité $1-p$, il est inchangé. Ce canal est simple tout en étant assez général pour caractériser la tolérance au bruit de différents codes.

On peut généraliser directement le canal d'inversion de bit symétrique classique au cas quantique. Dans ce contexte, ce sont les états de la base de calcul $\{|0\rangle, |1\rangle\}$ qui ont une probabilité p d'être échangés. Cependant, dans le cas quantique, on peut imaginer la version Z de ce canal, c'est-à-dire un canal d'inversion de phase, changeant $\alpha|0\rangle+\beta|1\rangle$ en $\alpha|0\rangle-\beta|1\rangle$. Quoique similaire à l'inversion de bit, l'inversion de phase n'a pas d'équivalent classique puisqu'elle affecte la phase relative entre deux états superposés. Les équations suivantes donnent l'expression du canal d'inversion de bit à un qubit, $\mathcal{E}_1^X$, et du canal d'inversion de phase à un qubit, $\mathcal{E}_1^Z$:

$$\mathcal{E}_1^X(\rho) = (1-p)\rho + pX\rho X, \qquad \mathcal{E}_1^Z(\rho) = (1-p)\rho + pZ\rho Z. \tag{2.14}$$

Ultimement, tout dispositif de traitement de l'information doit être implémenté par un système physique. Or, dans ce contexte, on peut identifier les processus à l'origine de ces canaux. Tout d'abord, les états 0 et 1 correspondant à deux niveaux d'énergies distincts du système, on comprend que l'inversion de bit est due à un échange d'énergie

entre le (qu)bit et son environnement. Par contre, la phase ne fait pas intervenir l'énergie directement. En effet, celle-ci est plutôt sensible à la différence d'énergie entre deux niveaux. Si le système est influencé par son environnement, ses niveaux ne seront pas parfaitement définis et leur énergie respective fluctuera, changeant ainsi cette différence d'énergie et affectant la phase relative d'un état superposé.

Comme expliqué ci-haut, les canaux à n qubits correspondants sont simplement le produit du même canal répété indépendamment sur les n qubits, que l'on dénote

$$\mathcal{E}_n^X = (\mathcal{E}_1^X)^{\otimes n}, \qquad \mathcal{E}_n^Z = (\mathcal{E}_1^Z)^{\otimes n}. \tag{2.15}$$

2.3.2 Canal à dépolarisation

Tout comme le canal d'inversion de bit dans la théorie de l'information classique, le canal à dépolarisation, $\mathcal{E}_1^D$, est un standard dans la théorie quantique. Il existe deux interprétations équivalentes du canal à dépolarisation. Tout d'abord, on peut dire qu'avec probabilité $1-q$ le qubit demeure intact et qu'avec probabilité q il est remplacé par un qubit dans un état complètement mélangé dont la matrice densité est $\frac{1}{2}I$. Une interprétation équivalente est de considérer qu'il est intouché avec probabilité $1-p$ et qu'avec probabilité p, il est affecté par une erreur de Pauli choisie de manière aléatoire uniforme. L'expression de ces interprétations équivalentes est donnée ci-dessous :

$$\mathcal{E}_1^D(\rho) = (1-q)\rho + q\frac{I}{2} = (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z), \tag{2.16}$$

avec $q = \frac{4}{3}p$.

Il est important de noter qu'il ne s'agit pas d'une concaténation de $\mathcal{E}_1^X$ et $\mathcal{E}_1^Z$, car $Y = iXZ$ corrèle les deux types d'erreurs. En effet, Y correspond à appliquer les deux erreurs X et Z à la fois. Si on sait que le système a subi au moins une erreur X sur un qubit donné, alors la probabilité qu'il ait subi une erreur Z aussi sera $p(Z|X) = 1/2$, car l'erreur est soit X, soit Y, de manière équiprobable. De manière analogue, $p(X|Z) = 1/2$. Si les deux types de bruits n'étaient pas corrélés, être au courant des erreurs d'un type ne changerait en rien notre connaissance des erreurs de l'autre type. Comme nous le verrons au chapitre 5, il s'agit d'un atout qu'on peut exploiter pour améliorer les performances de décodeurs connus.

Comme dans le cas des canaux d'inversion, le canal à dépolarisation à n qubits consiste du canal à dépolarisation appliqué de façon indépendante à chaque qubit : $\mathcal{E}_n^D = (\mathcal{E}_1^D)^{\otimes n}$.

Dans ce contexte, on peut donner une formule de la probabilité des erreurs de Pauli, e, à n qubits. Pour ce faire, introduisons la fonction $\omega : \mathcal{P}_n \mapsto \mathbb{N}$ qui associe à un opérateur de Pauli, e, son poids, i.e. le nombre d'opérateurs non-identité qu'il contient. Alors, on rend compte de notre hypothèse de localité et d'indépendance par la formule suivante pour la probabilité $P(e)$ d'une erreur e :

$$P(e) = \left(\frac{p}{3}\right)^{\omega(e)} (1-p)^{n-\omega(e)}. \tag{2.17}$$

On reconnaît une distribution binomiale des erreurs possibles qui rend compte de l'indépendance des erreurs. Par contre, la localité est plus subtile. Plus le poids d'une erreur est grand, moins cette erreur est probable. La distance géométrique séparant les erreurs individuelles n'intervient pas.

2.3.3 Canal à effacement

Intuitivement, le canal à effacement se comprend bien. Imaginons par exemple une fibre optique à l'aide de laquelle on transmet des photons. Une fois ceux-ci lancés, on s'attend à ce qu'une certaine fraction d'entre eux, dans ce cas p, soit absorbée ou perdue avant d'être parvenue à l'autre extrémité de la fibre.

De plus, ce canal a cette particularité que les erreurs sont annoncées. On peut modéliser mathématiquement ce canal en introduisant une variable binaire prenant la valeur 1 lorsque le qubit est perdu et la valeur 0 lorsqu'il ne l'est pas. Il s'agit de la seconde variable d'entrée du canal et on l'appelle le bit annonciateur d'une erreur. En effet, pour revenir à notre exemple, si on sait que les photons sont émis à une certaine cadence, on saura lorsqu'un photon n'est pas détecté alors qu'il aurait dû l'être. Dans ce cas, le vide est l'état annonciateur d'une perte.

$$\mathcal{E}_1(\rho \otimes |0\rangle) = (1-p)\rho \otimes |0\rangle + p\frac{I}{2} \otimes |1\rangle \tag{2.18}$$

2.4 Décodeurs

Pour la suite, nous continuons de travailler avec une distribution de probabilité sur tous les poids d'erreur possibles plutôt qu'avec un ensemble fini d'erreurs comme discuté à la section précédente. Nous nous concentrons sur le canal à dépolarisation. Rappelons

la formule donnant la probabilité d'une erreur, e, de poids $\omega(e)$,

$$P(e) = \left(\frac{p}{3}\right)^{\omega(e)} (1-p)^{n-\omega(e)}. \tag{2.19}$$

On rappelle que la mesure du syndrome fixe l'opérateur t dans la décomposition dans la base TLS d'une erreur $e = tls$. Dans ce contexte, le décodeur est l'algorithme qui décide la correction à appliquer étant donné qu'un syndrome, t, a été observé. On verra que pour quelques décodeurs, on peut trouver un problème de physique statistique analogue au problème du décodage.

Pour être d'une quelconque utilité, un décodeur doit décider suffisamment rapidement la correction à appliquer lorsqu'un syndrome est mesuré. En effet, si l'échelle de temps nécessaire au décodeur pour traiter le syndrome est comparable ou plus longue à l'échelle de temps sur laquelle agit le bruit, le décodeur est inutile. C'est pourquoi il est crucial de concevoir des décodeurs efficaces.

Il existe trois avenues principales que l'on peut prendre lorsqu'on décode. On distingue les décodages marginal, logique et à distance minimale.

Le décodage marginal cherche à optimiser la correction sur chacun des (qu)bits indépendamment les uns des autres. Cette méthode permet de préserver l'état de la majorité des bits. Si l'intégralité des bits n'est pas essentielle, cette méthode de décodage est appropriée. Par exemple, si on considère un cd-rom contenant de la musique, nous ne sommes pas intéressés à protéger la totalité des bits. Plutôt, on est prêt à accepter des bruits parasites mineurs, en autant que l'essentiel de la trame soit conservé. La capacité de cette méthode à préserver une information utile dépendra de l'application. Par exemple, pour les codes éparses quantiques (LDPC) [25], il s'agit du décodage approprié, tandis que pour le code de Kitaev, dont nous traitons principalement dans ce texte, ce n'est pas le cas. Pour ce dernier, le décodage logique est l'approche la plus performante.

Dans certains cas, seulement un petit nombre d'erreurs dominent la distribution de probabilité, ce qui rend superflu le fait de la traiter dans son ensemble. Il est possible de bien identifier l'information encodée en comparant seulement ces quelques erreurs dominantes. C'est ce que fait le décodage à distance minimale. De toutes les erreurs, il cherche la plus probable et suppose que c'est celle-là qui s'est produite. La distribution de probabilité des erreurs peut être telle que cette méthode soit aussi performante que le décodage logique. Cela dépend du code et du modèle de bruit.

Le décodage logique cherche à optimiser la correction globale plutôt que d'optimiser

la correction de chaque (qu)bit de façon indépendante. Il prend en compte tous les paramètres possibles telles la dégénérescence des erreurs et les corrélations entre celles-ci pour extraire l'information encodée de manière globale. Toutefois, dans ce contexte, l'information est soit complètement sauvegardée avec très grande fidélité, soit complètement perdue. Pour reprendre l'exemple précédent du cd-rom contenant de la musique, cette méthode de décodage n'est pas appropriée, car on ne veut pas soit avoir la trame sonore parfaite, soit l'avoir complètement perdue. Lorsque le cd-rom est égratigné à un endroit, on veut être en mesure de récupérer l'information contenue sur le reste du cd-rom.

On parle de décodeur plutôt que de décodage lorsqu'on considère un algorithme spécifique qui implémente une méthode de décodage ou une approximation de celle-ci. On peut alors étudier sa complexité et son seuil. Dans les deux sous-sections suivantes, on traite du décodeur à distance minimale et du décodeur logique. On discutera en détail du décodeur physique au chapitre 4.

2.4.1 Décodeur à distance minimale

Pour développer une intuition du fonctionnement du décodeur à distance minimale, il est utile de l'étudier à l'oeuvre sur un code simple pour lequel le décodeur à distance minimale fonctionne bien : le code à cinq qubits. Rappelons d'abord l'ensemble canonique le spécifiant :

$$\begin{array}{lllll} s_1 = X_1Z_2Z_3X_4 & s_2 = X_2Z_3Z_4X_5 & s_3 = X_1X_3Z_4Z_5 & s_4 = Z_1X_2X_4Z_5 & \overline{Z}_1 = Z_1Z_2Z_3Z_4Z_5 \\ t_1 = X_1X_3X_4X_5 & t_2 = X_2X_3 & t_3 = X_1X_5 & t_4 = X_2X_3X_4X_5 & \overline{X}_1 = X_1X_2X_3X_4X_5 \end{array} \tag{2.20}$$

Supposons qu'une erreur affecte notre code. Nous mesurons les stabilisateurs s_1 à s_4 et obtenons les valeurs : $+1, -1, -1, +1$. Ceci fixe la valeur de t pour l'erreur exprimée dans la base TLS : $t = t_2t_3 = X_1X_2X_3X_5$. Ayant déterminé t, reste à passer au travers de toutes les possibilités pour l et s. On remarque que $t_2t_3\overline{X} = X_4$ est une erreur de poids un. Comme on l'a vu à la section 2.2.3, si on passait à travers toutes les possibilités, on découvrirait qu'il s'agit de la seule erreur de poids un et que toute autre a un poids d'au moins trois. Si la probabilité d'erreur, p, pour un qubit est suffisamment faible, l'erreur de poids un dominera complètement la distribution de probabilité des erreurs possibles (conditionnelle au syndrome) par un facteur $\mathcal{O}(\frac{1}{p^2})$. Inutile alors de chercher plus loin, on a peu de chances de se tromper si on choisit de corriger l'erreur par $e = X_4$.

Le décodeur à distance minimale est le plus direct. De manière générale, il cherche l et s tel que $e = tls$ ait la plus grande probabilité, i.e. un poids minimal, et il en déduit que c'est l'erreur qui s'est produite :

$$e = t \underset{l \in L, s \in \mathcal{S}}{\operatorname{argmin}} \omega(tls). \tag{2.21}$$

On peut définir un problème analogue de physique statistique. Définissons l'énergie, E, d'une erreur comme étant son poids. Ainsi les erreurs les moins probables, ayant un poids plus élevé, auront une plus grande énergie.

$$E(e) = J\omega(e), \tag{2.22}$$

où J fixe les unités. À ce point-ci, cette définition semble artificielle, mais au chapitre suivant où on étudiera les codes topologiques, on verra qu'on peut donner une interprétation géométrique des erreurs et que les stabilisateurs étant locaux, les défauts pourront être identifiés à des particules. La notion d'énergie apparaîtra alors plus naturelle et justifiée.

Trouver l'erreur la plus probable revient à trouver celle ayant la plus faible énergie. On dit alors que ce décodeur opère à température nulle, $T = 0$, dans le sens où il ne s'intéresse qu'à l'énergie et pas à la dégénérescence (entropie) des erreurs probables.

2.4.2 Décodeur logique

Le décodeur à distance minimale est sous-optimal, car il ne tient pas compte de la dégénérescence des erreurs. En effet, puisque les opérateurs $s \in \mathcal{S}$ du stabilisateur agissent de manière triviale sur le code, on a que deux erreurs $e = tls$ et $e' = tls'$ sont équivalentes, car elles ne diffèrent que par un opérateur à l'action triviale.

Le décodeur logique prend en compte cette dégénérescence en calculant plutôt la marginale de chacune des classes d'équivalence logiques $l \in L$ étant donné t,

$$P(l, t) = \sum_{s \in \mathcal{S}} P(tls), \tag{2.23}$$

où $P(tls)$ est définie par le modèle de bruit.

Puis, le décodeur choisit l'opérateur l ayant la plus haute marginale et corrige par un

opérateur de cette classe : $e = tl$ (tous les s sont équivalents),

$$e = t \operatorname*{argmax}_{l \in L} P(l, t). \tag{2.24}$$

Encore une fois, on peut formuler ce problème en terme de physique statistique. En effet, on peut ramener ce problème à celui de minimiser une énergie libre de Gibbs. Tout d'abord, notons que l'on peut récrire la probabilité associée à une erreur (cf. éq. 2.19) comme suit

$$P(tls) = \frac{e^{-\beta E(tls)}}{\mathcal{Z}(\beta)}, \qquad \beta = \frac{1}{J} \ln \frac{3(1-p)}{p}, \tag{2.25}$$

où $\mathcal{Z}(\beta) = \sum_{tls} e^{-\beta E(tls)}$ est la fonction de partition servant à normaliser la distribution et β est la température inverse de Nishimori.

Avec ces quantités définies, on peut écrire

$$P(t, l) = \sum_{s \in \mathcal{S}} P(tls), \quad P(tls|t, l) = \frac{P(tls)}{P(t, l)}, \tag{2.26}$$

On peut en déduire que

$$\ln P(tls|t, l) = \ln P(tls) - \ln P(t, l) = -\beta E(tls) - \ln \mathcal{Z}(\beta) - \ln P(t, l). \tag{2.27}$$

Avec ces quantités, on peut donner une expression pour l'entropie et l'énergie moyenne,

$$S(t, l) = -\sum_{s \in \mathcal{S}} P(tls|t, l) \ln P(tls|t, l), \qquad E(t, l) = \sum_{s \in \mathcal{S}} P(tls|t, l) E(tls). \tag{2.28}$$

On peut formuler l'énergie libre de Gibbs, F,

$$\begin{aligned} \beta F(t, l) &= \beta E(t, l) - S(t, l), \\ &= \textstyle\sum_{s \in \mathcal{S}} P(tls|t, l) \left[\beta E(tls) + \ln P(tls|t, l)\right], \\ &= \textstyle\sum_{s \in \mathcal{S}} P(tls|t, l) \left[-\ln \mathcal{Z}(\beta) - \ln P(t, l)\right], \text{ d'après (2.27)} \\ &= \left[-\ln \mathcal{Z}(\beta) - \ln P(t, l)\right] \textstyle\sum_{s \in \mathcal{S}} P(tls|t, l), \\ &= \left[-\ln \mathcal{Z}(\beta) - \ln P(t, l)\right], \end{aligned} \tag{2.29}$$

car $\sum_{s\in\mathcal{S}} P(tls|t,l) = 1$. En arrangeant les termes et en prenant l'exponentielle, on obtient

$$P(t,l) = \frac{e^{-\beta F(t,l)}}{\mathcal{Z}(\beta)}, \tag{2.30}$$

et on peut conclure que maximiser $P(t,l)$ par rapport à l revient à minimiser l'énergie libre F par rapport à l.

Décodeur logique de codes concaténés

Rappelons que la concaténation est une procédure permettant d'utiliser plusieurs codes de taille donnée pour construire une famille de codes à taille variable diminuant à un niveau arbitrairement bas l'amplitude du bruit (cf. section 2.1.1). On peut utiliser plusieurs décodeurs logiques pour décoder efficacement et de manière optimale des codes concaténés [24]. Pour y parvenir, on doit utiliser des versions douces (*soft*) du décodeur logique. La version adoucie d'un décodeur ne choisit pas de correction à appliquer. Elle ne fait que calculer et transmettre en sortie la distribution de probabilité des différentes corrections possibles.

Rappelons certains éléments illustrés sur la figure 2.1 qui nous permettrons de construire un décodeur efficace. Premièrement, les blocs d'un niveau ne font intervenir que les opérateurs logiques du niveau immédiatement inférieur. Deuxièmement, on constate que les différents blocs d'un même niveau sont indépendants. La première affirmation nous permet de conclure que l'on peut décoder le code niveau par niveau. On décode le dernier niveau en fonction du modèle de bruit (dépolarisation, inversion, etc.). Puis on soumet comme entrée au décodeur du niveau supérieur la distribution de probabilité sur les opérateurs logiques que l'on vient de décoder, c'est-à-dire que cette distribution sert de modèle de bruit effectif au niveau précédent de concaténation. Puis celui-ci décode à son tour et passe la nouvelle information au niveau qui le précède, etc. On peut répéter cette procédure jusqu'à ce qu'on atteigne le premier niveau qui est le qubit logique encodé au départ. La dernière étape consiste à choisir l'erreur logique la plus probable pour corriger. La figure 2.2 présente la structure des décodeurs logiques doux.

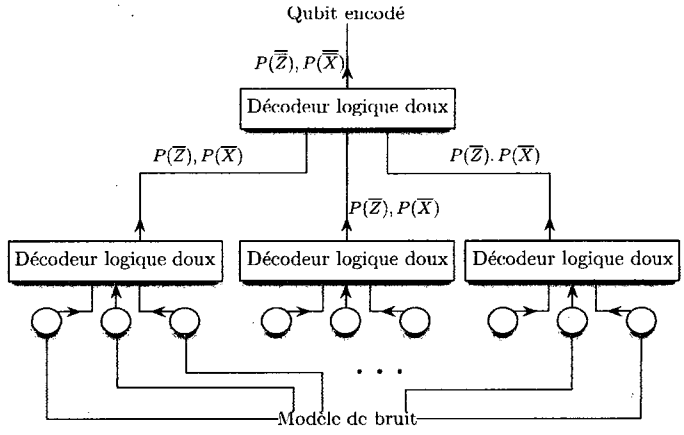


FIGURE 2.2 – Décodeur de codes concaténés. Exemple analogue à ce qui a été présenté à la figure 2.1.

2.5 Complexité

Décodeur à distance minimale

À priori, le décodeur à distance minimale n'est pas efficace, car il doit chercher à travers tous les l possibles alors qu'il y en a 2^k et aussi à travers tous les s possibles alors qu'il y en a 2^m. D'une part, pour des codes avec m et k petits, ceci ne pose pas problème. Par exemple, dans le cas du code à cinq qubits, on peut aisément dresser la table des corrections à appliquer comme on l'a vu au Tableau 2.1. D'autre part, dans plusieurs cas, les détails du code permettent de concevoir des algorithmes efficaces pour trouver cette erreur minimale. Par exemple, pour le code torique que nous étudierons au chapitre 3, il existe un décodeur à distance minimale de complexité polynomiale, quoique limité en pratique. De plus, la classe des codes algébriques [6] a été conçue spécifiquement pour qu'on sache en effectuer le décodage à distance minimale de manière efficace. Par contre, de tels codes ont un faible rendement, c'est-à-dire qu'il faut beaucoup de qubits physiques pour obtenir quelques qubits logiques (k/n petit).

Décodeur logique

Encore une fois, il faut traiter les $2^m \times 2^k$ possibilités pour s et l, ce qui rend ce décodeur inefficace, quoiqu'optimal. Le décodeur proposé dans ce mémoire tente d'approximer de manière efficace ce décodeur. Pour certains codes, comme les codes concaténés, il est

possible de contourner cette difficulté comme expliqué à la section suivante.

Décodeur logique de codes concaténés

Analysons la complexité de ce décodeur. On a l niveaux de décodage à faire et la plus longue d'entre elle nécessite n^{l-1} décodeurs. Grâce à la parallélisation, tous les décodeurs d'un niveau peuvent opérer simultanément. Les ressources temporelles nécessaires sont donc $\mathcal{O}(l)$. De plus, à la section 2.1.1, on a vu que le taux de suppression du bruit est très sensible au nombre de niveaux d'encodages. Il croît comme la double exponentielle du nombre de niveaux l : $\frac{1}{\epsilon} \sim \mathcal{O}(2^{2^l})$ ou encore $\log \frac{1}{\epsilon} \sim \mathcal{O}(2^l)$ où ϵ est le taux de bruit effectif sur l'information encodée. Ceci signifie que même un petit nombre de niveaux de concaténation améliore la performance du code initial de manière significative.

Chapitre 3

Codes topologiques

3.1 Code de Kitaev

Le code de Kitaev [7, 18], aussi connu sous le nom de code torique, est l'archétype de la mémoire 2D utilisant la topologie pour se prémunir contre les erreurs locales. Il s'agit d'un modèle jouet permettant de tester facilement la résistance des codes topologiques aux erreurs. Comme nous le verrons, il s'agit d'un code stabilisateur encodant deux qubits logiques à l'aide de $n = 2\ell^2$ qubits physiques où ℓ est la dimension linéaire du système.

3.1.1 Base de Pauli spécifiant le code de Kitaev

Tore

Considérons un réseau 2D carré de dimension $\ell \times \ell$. Pour en faire un tore, il suffit d'identifier l'arête de droite à celle de gauche et celle du bas à celle du haut. Dans le code de Kitaev, on dispose un qubit sur chaque arête de notre réseau, comme illustré à la figure 3.1. On en déduit qu'il y a $n = 2\ell^2$ qubits physiques disponibles pour l'encodage.

stabilisateur

L'étape suivante consiste à spécifier les générateurs du groupe stabilisateur, $\mathcal{S}$. Pour ce faire, on définit deux ensembles d'opérateurs. D'une part, à chaque sommet, s, on associe un opérateur, A_s, appelé croix ou opérateur de site, qui consiste à appliquer l'opérateur X à chacun des quatre qubits voisins du sommet en question. On a alors défini un opérateur par site, donc ℓ^2 croix au total. D'autre part, à chaque plaquette, p, i.e. sommet du réseau

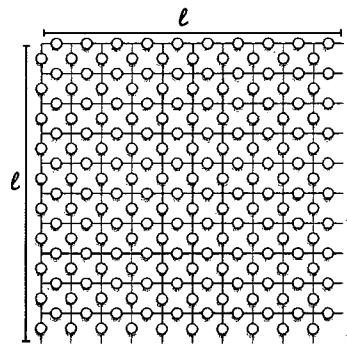


FIGURE 3.1 – Les qubits sont disposés sur les arêtes d'un tore, réseau carré aux conditions de frontières périodiques.

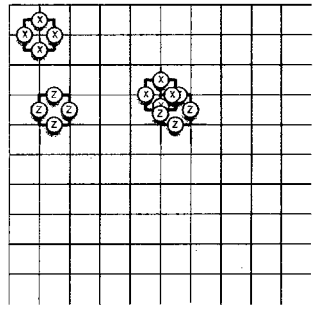


FIGURE 3.2 – Opérateurs croix (bleu) et opérateurs plaquettes (vert).

dual, on associe un opérateur, appelé opérateur de plaquette, B_p, qui consiste à appliquer l'opérateur Z à chacun des quatre qubits formant la plaquette. Un exemple de chacun est illustré à la figure 3.2. Ces deux ensembles d'opérateurs définissent les générateurs du stabilisateur.

$$A_s = \prod_{q \in v(s)} X_q, \quad B_p = \prod_{q \in v(p)} Z_q, \tag{3.1}$$

où $v(s)$ est l'ensemble des qubits voisins du sommet s et $v(p)$, l'ensemble des qubits voisins de la plaquette p.

Rappelons que le groupe stabilisateur doit être abélien. Évidemment, les croix commutent entre elles ainsi que les plaquettes, car ces opérateurs ne sont formés ou bien que de X ou bien que de Z : $(\forall i, i')$ $[A_i, A_{i'}] = 0$ et $[B_i, B_{i'}] = 0$. La figure 3.2 nous illustre

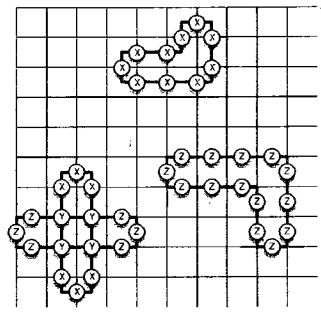


FIGURE 3.3 – Exemple d'élément du stabilisateur du code de Kitaev.

pourquoi une croix et une plaquette commutent. En effet, on voit que soit l'intersection de leur support est vide, soit elle contient deux qubits. Il y a donc toujours un nombre pair d'anticommutations et on en déduit que les croix et les plaquettes commutent : $(\forall s, p) \quad [A_s, B_p] = 0$.

Notons que chacun des qubits participe à exactement deux croix et deux plaquettes. Comme $X^2 = I$, on a que le produit de toutes les croix donne l'identité, $\prod_{s=1}^{\ell^2} A_s = I$. On obtient le même résultat avec les plaquettes $(Z^2 = I) : \prod_{p=1}^{\ell^2} B_p = I$. Ces deux équations nous permettent de conclure que seulement $\ell^2 - 1$ des croix sont indépendantes ainsi que $\ell^2 - 1$ des plaquettes. On a donc $2\ell^2 - 2$ générateurs du stabilisateur.

Les opérateurs qui consistent en une ligne de X sur le réseau dual ou de Z sur le réseau primitif sont appelés opérateurs lignes. Quand la ligne n'a pas de frontière, i.e pas de bouts, on l'appelle un cycle. Si le cycle « s'enroule » autour du corps ou du trou du tore, on dit qu'il s'agit d'un cycle non-trivial. Comme les opérateurs croix et plaquettes forment tous les cycles triviaux de taille minimale, i.e. quatre, on en déduit qu'il s'agit d'une base pour tous les cycles triviaux. On en conclut que le groupe stabilisateur est l'ensemble de tous les produits de cycles triviaux possibles. Un exemple d'élément du stabilisateur est donné à la figure 3.3.

Opérateurs logiques

Comme on vient de le voir, l'ensemble des cycles triviaux forme le stabilisateur. On s'attend à avoir deux qubits logiques dans ce code, car on a $n - 2$ générateurs du stabilisateur. De plus, on peut conclure que tout élément commutant avec tous les cycles triviaux doit être un cycle. En effet, si un opérateur ligne possédait une frontière, il y aurait anticommutation avec les opérateurs s'y situant comme on le verra à la section

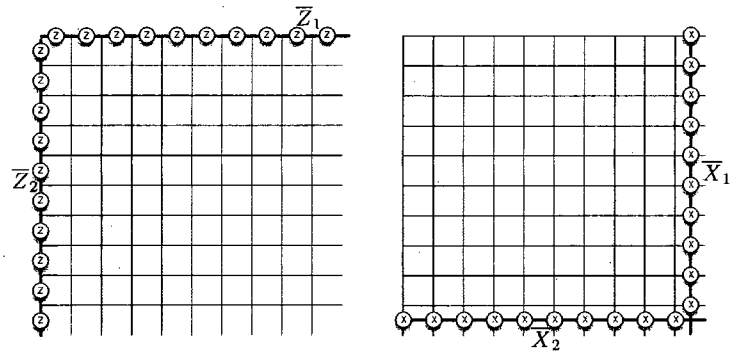


FIGURE 3.4 – Base possible pour le groupe logique du code de Kitaev.

suivante. On en déduit directement que le groupe logique est constitué de l'ensemble des cycles non-triviaux. La figure 3.4 présente un choix de jauge possible pour le groupe logique. Toute déformation de ces opérateurs par des cycles triviaux serait aussi valable.

Ces quatre opérateurs commutent avec tous les éléments du stabilisateur, car ce sont des cycles. De plus, on vérifie $[\overline{X}_i, \overline{Z}_j] = 0$ pour $i \neq j$ et $\{\overline{X}_i, \overline{Z}_i\} = 0$ ce qui correspond à l'algèbre de deux qubits effectifs. Ceux-ci seront nos qubits logiques délocalisés et $\overline{X}_i$ et $\overline{Z}_i$ les caractérisent comme X et Z caractérisent un qubit physique. Comme $\overline{X}_i$ et $\overline{Z}_i$ sont des cycles non-triviaux, on dira que nos qubits sont encodés dans les degrés de liberté topologiques du système, d'où l'appellation de code topologique.

Erreurs pures

On a spécifié le groupe stabilisateur et le groupe logique. Reste à spécifier un choix de jauge pour les erreurs pures. Il ne peut s'agir de cycles, puisque ceux-ci commutent avec les éléments du stabilisateur. Il s'agira donc d'opérateurs lignes avec des frontières. Un exemple de tels opérateurs est donné à la figure 3.5. On voit que ces opérateurs lignes X (Z) anticommutent avec les plaquettes (croix) où se situent leur frontière.

Rappelons qu'une des croix et une des plaquettes sont redondantes. Elles ne seront donc pas mesurées lors de la lecture du syndrome. Celles-ci sont illustrées en pointillé sur la figure 3.6. Aussi, on peut y voir deux exemples d'erreurs pures. Les erreurs pures des plaquettes, $\{t_p^B\}$, consistent en une ligne horizontale de X partant du coin sud-est de la grille. Puis, elles se poursuivent par une ligne verticale qui atteint la plaquette en question (cf. Fig. 3.6). Les erreurs pures des croix, $\{t_s^A\}$, consistent en ligne horizontale de Z partant du coin nord-ouest de la grille. Puis, elles se poursuivent par une ligne

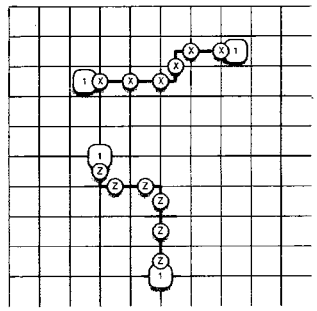


FIGURE 3.5 – Exemples d'opérateurs lignes qui ne sont pas des cycles.

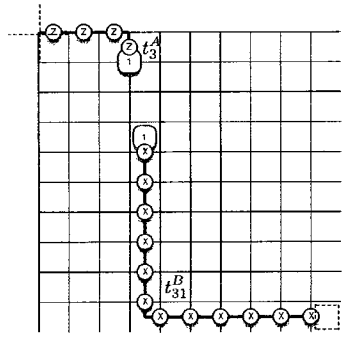


FIGURE 3.6 – Exemples d'erreurs pures. La croix et la plaquette en pointillé ne font pas partie des générateurs du stabilisateur puisqu'elles sont redondantes.

verticale qui atteint la croix en question (cf. Fig. 3.6).

D'une part, on s'assure que les erreurs pures commutent entre elles, car les deux types ne se croisent jamais. D'autre part, la figure 3.6 nous permet de nous convaincre qu'elles commutent aussi avec les opérateurs logiques. Finalement, on voit aussi qu'elles anticommutent avec les deux stabilisateurs se trouvant à leurs extrémités et qu'un seul de ceux-ci est un générateur du stabilisateur.

Récapitulons en explicitant le tableau caractérisant la base de Pauli spécifiant ce code :

$$\begin{array}{cccccccc} A_1 & \dots & A_{\ell^2-1} & B_1 & \dots & B_{\ell^2-1} & \overline{Z}_1 & \overline{Z}_2 \\ t_1^A & \dots & t_{\ell^2-1}^A & t_1^B & \dots & t_{\ell^2-1}^B & \overline{X}_1 & \overline{X}_2 \end{array} \tag{3.2}$$

3.1.2 Erreurs et décodage

À partir de ce point, on peut adopter un point de vue beaucoup plus physique de la situation. En effet, considérons la formulation hamiltonienne du code,

$$H = -\sum_{s=1}^{\ell^2-1} A_s - \sum_{p=1}^{\ell^2-1} B_p. \tag{3.3}$$

Ce hamiltonien est la somme de termes qui commutent. Il est donc simple de le résoudre exactement. Il possède un gap, sous lequel le sous-espace fondamental est dégénéré. Rappelons que le code consiste en ce sous-espace fondamental. De plus, on sait que chaque paire $\{\overline{X}, \overline{Z}\}$ encode un qubit effectif dans ce sous-espace. Comme on l'a vu à la section précédente, il y a deux telles paires, le fondamental est donc dégénéré quatre fois (2^2). Remarquons que la dégénérescence est déterminée par la topologie de la surface. En effet, à chaque cycle non-trivial on a pu associer un opérateur logique, encodant un qubit et multipliant ainsi par deux la dégénérescence du fondamental. On comprend alors que pour une surface qui posséderait une topologie différente, on pourrait déterminer la dégénérescence de son fondamental en identifiant les cycles non-triviaux qui l'habitent : 2^k où k est le nombre de cycles non-triviaux.

L'effet d'une erreur isolée est d'injecter de l'énergie dans le système en inversant la valeur propre des deux générateurs du stabilisateur voisins de l'erreur. Comme ces générateurs sont locaux, on peut les interpréter comme des particules dont la création a un coût énergétique. Une erreur crée deux particules (ou quatre pour une erreur Y) sur le réseau. On les appelle aussi défauts. D'autres erreurs locales subséquentes pourront annihiler ou propager des particules existantes ou bien créer de nouvelles paires. Notons que ces particules apparaissent toujours en paires. Les erreurs X créent des défauts sur les plaquettes, qu'on appelle défauts X, et les erreurs Z créent des défauts sur les croix, qu'on appelle défauts Z. Comme Y est à la fois une erreur X et une erreur Z, Y crée une paire de chaque sorte.

La figure 3.7 donne quelques exemples pour illustrer. À gauche, on voit qu'une erreur X isolée crée une paire de particules sur ses plaquettes avoisinantes. En effet, celles-ci anti-commutent avec l'erreur X. Leur valeur propre +1 se fait alors inverser à -1 par X. Le système acquiert deux fois deux unités d'énergie (1-(-1)=2). Comme cette énergie est localisée sur les plaquettes, on peut l'interpréter en termes de particules. C'est ainsi que l'erreur X résulte en la création de deux particules sur les plaquettes qui se

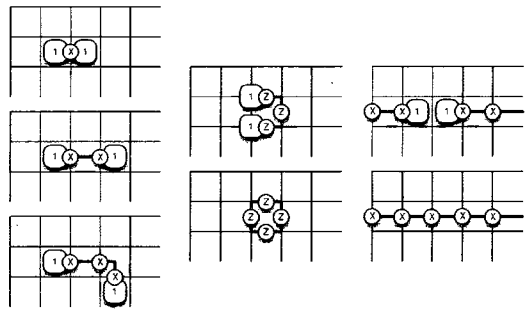


FIGURE 3.7 – Effet des erreurs sur le code de Kitaev : une erreur crée une paire de particules. Les erreurs subséquentes peuvent propager ou annihiler ces erreurs ou encore en créer de nouvelles. Deux défauts s'annihilant après avoir parcouru un cycle non-trivial résulte en une opération effectuée sur le fondamental, i.e. le code.

situent de part et d'autre. À gauche toujours, on voit comment des erreurs subséquentes propagent les particules. Une deuxième erreur X venant affecter une plaquette contenant déjà un défaut ajoute une anti-commutation (i.e. un *flip*) et retourne ladite plaquette à son niveau d'énergie le plus bas, supprimant le défaut. Toutefois, cette nouvelle erreur X anti-commute aussi avec son autre plaquette avoisinante. Cette dernière subit aussi une inversion et un défaut y apparaît. Au bilan, le défaut s'est déplacé d'une plaquette à l'autre comme le montre la figure 3.7.

Au centre de la figure 3.7, on voit comment une erreur peut annihiler deux défauts s'ils se trouvent de part et d'autre de l'erreur. Remarquons que si les défauts sont fusionnés, ils n'ont pas nécessairement rebroussé chemin exactement. En fait, il n'est pas obligatoire que ces défauts soient déjà connectés par une ligne d'erreurs. La seule contrainte est qu'on ne peut annihiler que des défauts du même type, X ou Z. Toutefois, si on fusionne des particules étant effectivement connectées par une ligne d'opérateurs, l'opérateur ligne résultant, qu'on appelle ligne de vie, forme un cycle. En général, on appelle ligne de vie l'opérateur ligne résultant de la création, diffusion, puis annihilation d'une particule ou d'un ensemble de particules. Au centre de la figure 3.7, on voit qu'après avoir été annihilés, deux défauts Z ont effectué un cycle trivial. Or, ce cycle est sans impact sur l'état du code. Cependant, comme on le voit à droite, on comprend que si deux défauts sont fusionnés après avoir parcouru un cycle non-trivial, cela a pour effet d'effectuer une opération logique sur le code.

Dans ce contexte, on a un protocole pour passer d'un état fondamental à un autre

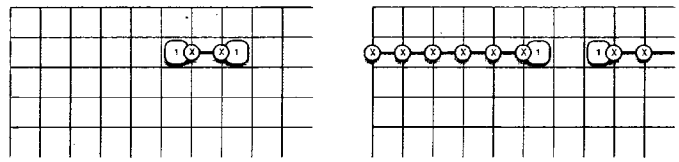


FIGURE 3.8 – Exemples de correction pour un appariement de deux particules X sur le code torique.

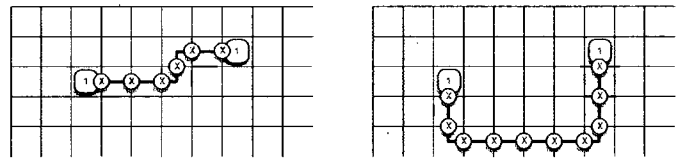


FIGURE 3.9 – Exemples de corrections appartenants à la même classe logique.

état fondamental en manipulant la topologie du réseau. On crée deux particules X par exemple, puis on les fait voyager autour du tore en appliquant localement une suite de X pour finalement les annihiler. En les fusionnant, on retourne au fondamental du système et on a effectué un cycle non-trivial d'opérateurs X, i.e. une opération logique $\overline{X}$ (cf Fig. 3.7, doite). On peut faire de même pour $\overline{Z}$.

De plus, le décodage peut aussi être interprété dans ce contexte. En effet, le syndrome correspond à la configuration des différentes particules sur la grille. Le problème du décodage consiste alors à choisir une procédure pour retourner dans le sous-espace fondamental du système. Pour ce faire, on choisit un appariement des particules puis on les annihile conséquemment. Toutefois, il n'y a pas que l'appariement qui compte, car quatre classes logiques de recombinaisons sont possibles pour toute paire de défauts. La figure 3.8 présente deux corrections de classes différentes pour un même appariement.

Deux corrections appartiennent à la même classe logique si et seulement si leur produit est un élément du stabilisateur, i.e un cycle trivial. Un exemple est donné à la figure 3.9. On comprend alors que les deux corrections de la figure 3.8 appartiennent à deux classes différentes. Comme il y a deux générateurs de la logique X, il y aura quatre classes X ; de même pour Z. Donc, en combinant toutes ces possibilités, on a un total de seize classes logiques différentes.

Un exemple plus complet est donné à la figure 3.10. En rouge, on voit un exemple d'erreur produite pas l'environnement : deux paires de défauts sont créées. En dessous,

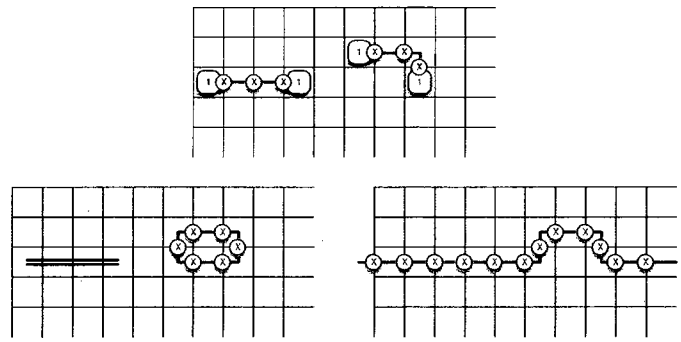


FIGURE 3.10 – Exemple plus général d'erreur. En rouge, l'erreur produite par l'environnement. En bleu, différentes corrections possibles. À gauche, correction adéquate résultant en un cycle trivial. À droite, correction déficiente, résultant en un cycle non-trivial. Insistons sur le fait que les deux corrections proposées apparient des défauts différents.

on voit deux corrections possibles en bleu. Celle de gauche est adéquate, puisque les lignes de vie résultantes des particules après correction sont triviales. Celle de droite est déficiente, puisque les lignes de vie forment un cycle non-trivial. Insistons sur le fait que ce ne sont pas les mêmes paires de particules qui sont fusionnées dans les deux corrections proposées. Le but de cet exemple est de mettre l'accent sur le fait qu'on doit tout d'abord choisir un appariement de tous les défauts et ensuite choisir une classe homologique pour chacun de ces appariements. On a donc beaucoup plus de corrections possibles que seize classes homologiques par paire, car il faut aussi compter les différentes façons de former ces paires.

En résumé, lorsqu'une erreur survient, on obtient la configuration des défauts en mesurant les générateurs du stabilisateur. Puis, on utilise un décodeur pour choisir une correction. Si la classe logique de la correction est la même que celle de l'erreur, alors, après annihilation, les défauts n'auront fait que des parcours triviaux et on aura bien corrigé. Par contre, si les classes ne concordent pas, certaines paires de défauts auront parcouru un chemin non-trivial, ce qui résultera en une erreur de correction et l'information sera perdue. Il existe seize telles classes de chemins qui correspondent aux seize classes logiques du code.

3.1.3 Décodeur à distance minimale : algorithme de Edmond

Il existe un décodeur efficace en principe pour le code torique et qui est utilisé dans la littérature. Il s'agit de l'algorithme de Edmond [12]. Cet algorithme trouve l'appariement de distance minimale étant donné un graphe complètement connecté. Il est aussi connu sous le nom de *minimum weight perfect matching algorithm* (PMA). Cet algorithme est exact, mais pour un décodage sous-optimal comme on le verra.

Dans le contexte de notre problème, PMA trouve la correction ayant le plus petit poids. PMA est donc un décodeur à distance minimale. Dans le chapitre précédent, on a vu pourquoi ce type de décodeurs est sous-optimal. Il opère à $T = 0$ et trouve une correction d'énergie minimale. Ce décodeur possède un seuil de $p_{th}^{BF} \sim 10.3\%$ sur le canal d'inversion et de $p_{th}^{D} \sim 15.5\%$ [16] sur le canal dépolarisant. Il décode les erreurs X et Z indépendamment alors qu'on sait que pour le canal dépolarisant, il existe des corrélations entre X et Z introduites par Y.

De plus, malgré que cet algorithme soit de complexité polynomiale, le temps nécessaire à son exécution croît comme $\mathcal{O}(\ell^6)$, ce qui en pratique limite son usage à des tailles relativement petites de $\ell \sim 10^2$. Or, les effets de taille sont importants dans les petits réseaux, ce qui diminue la qualité du code pour de telles valeurs de ℓ. La transition de phase ordonnée-désordonnée présentée par la correction d'erreur ne prendra effet que pour des ℓ grands ($\ell \gtrsim 10^3$).

Finalement, ce décodeur est développé dans le cadre spécifique du code de Kitaev. D'autres codes topologiques, comme les codes de couleur [3], comportent des règles de fusion de particules plus complexes. Par exemple, il est possible qu'une erreur crée un triplet de particules plutôt qu'une paire. Dans ce cas, PMA est inadéquat, et il n'existait auparavant aucun algorithme efficace de décodage. Or, la méthode introduite dans ce travail s'applique à de tels codes.

3.1.4 Décodeur logique : classes homologiques

Rappelons que le décodeur logique calcule la probabilité marginale de chacune des classes logiques l étant donné un syndrome t. Pour ce faire, on somme sur tous les stabilisateurs,

$$P(l,t) = \sum_{s \in \mathcal{S}} P(tls). \tag{3.4}$$

Or, dans le cas du code de Kitaev, les erreurs pures sont des configurations, C, de particules, les stabilisateurs sont des cycles triviaux et les classes logiques, des classes homologiques, Ω, de lignes de vie γ. Donc, décoder revient à sommer, pour chaque classe homologique, sur toutes ses lignes de vie compatibles avec la configuration des défauts ; celles-ci ne différent que par un cycle trivial :

$$P(\Omega, C) = \sum_{\gamma \in \Omega \cap C} P(\gamma). \tag{3.5}$$

Ainsi, plutôt que de rechercher les trajectoires de plus petit poids appariant les défauts, on cherche la classe homologique de trajectoires la plus probable. En effet, deux lignes de vie appartenant à la même classe homologique ont le même sens logique pour le code et n'ont donc pas à être distinguées. Elles sont équivalentes. Une fois la classe la plus probable identifiée, on peut corriger par n'importe quelle trajectoire de cette classe puisqu'elles ont toutes le même effet.

3.2 Codes de surface

Comme on l'a dit plus haut, le code de Kitaev est un modèle jouet en ce sens qu'on n'espère pas construire en laboratoire un tore sur lequel serait déposé des qubits. Toutefois, on peut obtenir un résultat très similaire à l'aide d'un réseau 2D plat plutôt que torique [5, 7, 13]. En effet, on verra qu'on peut perforer cette feuille de qubits pour lui ajouter des degrés de liberté topologiques.

Il existe deux manières de choisir les frontières d'un réseau 2D plat : frontière lisse (*smooth*) ou frontière rugueuse (*rough*). La figure 3.11 montre ces deux types de frontières. L'analyse des deux réseaux est analogue, nous choisissons de nous concentrer sur un réseau aux frontières lisses. On calcule qu'on a $n = 2\ell^2 + 2\ell$ qubits sur ce réseau. Les opérateurs plaquettes sont définis comme précédemment. Il y en a ℓ^2 et elles sont toutes indépendantes. On peut aussi définir des croix sur tous les sommets contenus à l'intérieur des frontières, il y en a $(\ell - 1)^2$. Pour les sommets situés sur une frontière, mais pas sur un coin, on peut définir l'équivalent d'une croix, mais possédant seulement trois branches (cf. Fig. 3.12), il y en a $4(\ell - 1)$. Finalement, pour les sommets sur les coins, on définit une croix à deux branches, il y en a quatre. Toutes ces croix ne sont pas indépendantes. En effet, on peut exprimé un des coins comme le produit de toutes les autres croix. On a alors $\ell^2 + 2\ell$ croix indépendantes. Au total, on se retrouve avec $2\ell^2 + 2\ell$ générateurs du

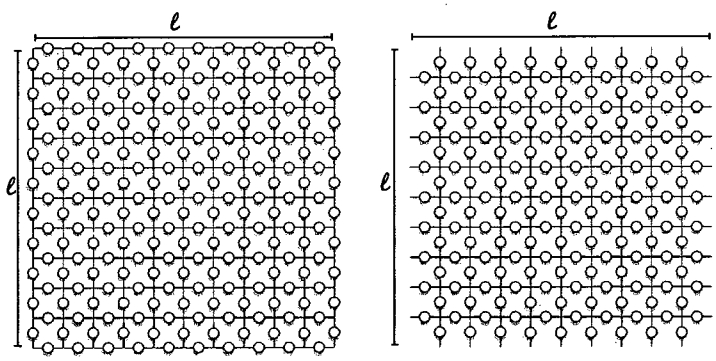


FIGURE 3.11 – Les deux types de frontières possibles pour un réseau plat 2D : lisse et rugueuse respectivement.

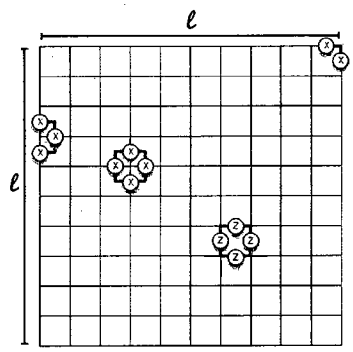


FIGURE 3.12 – Différents stabilisateurs du code de surface 2D à frontières lisses.

stabilisateur, ce qui correspond au nombre de qubits sur le réseau. Nous sommes donc devant un hamiltonien dont le fondamental n'est pas dégénéré et alors aucun qubit n'y est encodé. Ce code spécifie un état unique plutôt qu'un sous-espace.

Avant de voir comment on peut modifier le système pour augmenter la dégénérescence du fondamental pour en faire un code, nous traiterons rapidement les erreurs pures, car elles sont sensiblement différentes de celles du code de Kitaev dans le cas des plaquettes. On voit que tout opérateur ligne X joignant une plaquette à la frontière n'anticommutera qu'avec ladite plaquette et est donc candidat pour être une erreur pure. De plus, on comprend que le produit de deux tels candidats donne un élément du stabilisateur, ce qui est attendu. Reste à choisir une jauge. On choisit de définir l'erreur pure d'une plaquette comme étant la ligne X verticale joignant la plaquette à la frontière du bas. Les erreurs pures des croix sont les mêmes que celles convenues pour le code torique. En

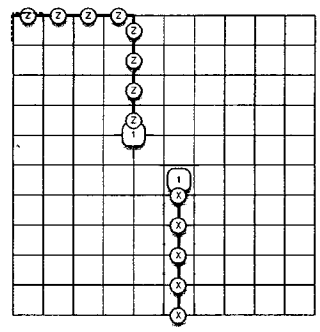


FIGURE 3.13 – Différentes erreurs pures du code de surface 2D à frontières lisses.

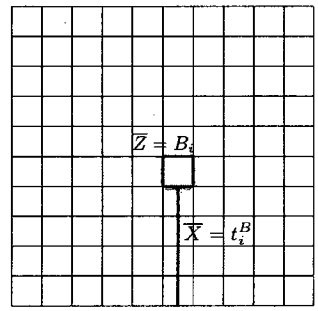


FIGURE 3.14 – Ajout d'un qubit logique au code en relâchant la contrainte sur un qubit stabilisateur.

effet, on a une croix redondante et les erreurs pures doivent connecter la croix considérée avec la croix redondante. La figure 3.13 en donne des exemple.

Pour obtenir un sous-espace plutôt qu'un état, on doit introduire des degrés de liberté. On y parvient en relaxant certaines des contraintes sur les stabilisateurs. En effet, supposons qu'on n'exige plus qu'une des plaquettes soit de valeur propre $+1$. Alors, comme le montre la figure 3.14, un des qubits effectifs stabilisateurs devient un qubit logique. Ce qubit stabilisateur était défini par un certain $\{s_i, t_i\}$ que nous interprétons maintenant comme un qubit logique défini par $\{\overline{Z}_i = s_i, \overline{X}_i = t_i\}$. En d'autres mots, l'opérateur logique $\overline{X}$ sur ce qubit sera une ligne X joignant celui-ci à la frontière et l'opérateur logique $\overline{Z}$ sera une boucle de Z encerclant la *perforation*. Puisqu'on relâche la contrainte sur une plaquette on parlera d'un qubit lisse.

On comprend intuitivement qu'on peut augmenter la robustesse de notre qubit aux erreurs locales X en l'éloignant le plus possible des frontières, ce qui rendra de plus

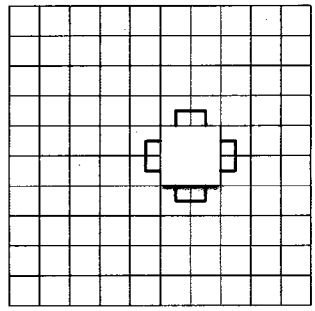


FIGURE 3.15 – En agrandissant la perforation, le nombre de qubits ne faisant plus partie du code balance le nombre de générateurs du stabilisateur qui sont relâchés pour qu'il ne reste toujours qu'un qubit logique. Certaines croix sont transformées.

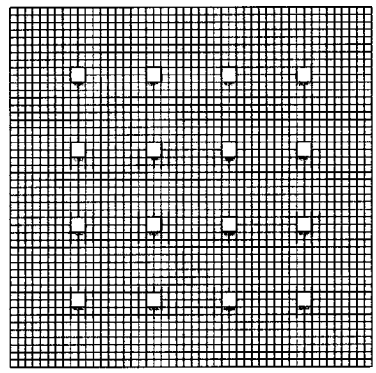


FIGURE 3.16 – Exemple de code où l'on a effectué 16 perforations de circonférence huit et distantes de huit entres elles et de la frontière.

en plus improbable l'éventualité d'une erreur connectant la frontière à la perforation. L'équivalent pour les erreurs Z est d'augmenter la surface de la perforation rendant plus improbable l'éventualité d'une ligne Z l'encerclant. Pour y parvenir, il suffit de relâcher la contrainte sur une des plaquettes voisines. En effet, le qubit appartenant à la fois aux deux plaquettes ne fera plus partie du code. On aura rejeté un qubit et relaxé une nouvelle contrainte. Les deux effets s'annulant, on ne change pas le nombre de qubits logiques. Les croix auxquelles participaient les qubits rejetés sont transformées en conséquence comme en témoigne la figure 3.15. Dans cet exemple, on a rejeté quatre qubits, alors que quatre plaquettes et une croix ont été relaxées : on a toujours un qubit logique.

On peut faire une analyse analogue dans le cas où les frontières sont rugueuses plutôt que lisses et on peut relâcher la contrainte sur une croix plutôt que sur une plaquette pour

se retrouver avec un qubit rugueux. On peut même mélanger les deux types de frontières et de qubits sur un même réseau. Comme ces cas se traitent de manière similaire à ce qui a été dit ci-haut, nous n'en discuterons pas davantage.

3.2.1 Analogie physique

Comme on l'a vu, le hamiltonien du réseau 2D initial n'est pas dégénéré. Si on veut un code contenant plusieurs qubits, il faut un sous-espace fondamental grandement dégénéré. On y parvient en pratiquant des trous dans le réseau. On peut imaginer que chacun de ces trous a deux états de base : magnétiquement chargé et neutre. Passer de chargé à neutre correspond à une inversion de charge et donc à l'opérateur logique $\overline{X}$. Pour effectuer une telle inversion de charge, on applique une chaîne de X connectant le trou à la frontière. On peut voir cela comme apporter un vortex de champ magnétique à partir de l'extérieur du réseau vers l'intérieur d'un trou. L'opérateur logique $\overline{Z}$ est plutôt constitué d'une chaîne de Z entourant un trou. On peut concevoir que cette chaîne a été produite par la création d'une paire de charges électriques qui se sont déplacées, encerclant le trou pour finalement fusionner, laissant derrière elles leur ligne de vie. Ceci a pour conséquence de procurer une phase à l'état du système si le trou est chargé via un effet Aharonov-Bohm. Pour éviter les erreurs, il faut disposer les trous loin les uns des autres ainsi que loin de la frontière du réseau pour limiter les échanges spontanés de vortex magnétiques. De plus, il faut que les trous soient de grande taille pour éviter le déphasage. La figure 3.16 présente une surface contenant 16 perforations de périmètre huit et distantes de huit les unes des autres et de la frontière.

3.2.2 Décodage

On a vu qu'on peut arriver à un code 2D qualitativement identique au code torique grâce au code de surface. De plus, celui-ci peut aussi être décodé par le décodeur à distance minimale PMA. En fait, la majorité des décodeurs décodant le code torique pourront aussi être adaptés pour décoder le code de surface, car les concepts d'opérateurs lignes et l'interprétation en termes de particules et de lignes de vie sont toujours applicables dans ce cas.

Chapitre 4

Outils de la mécanique statistique

Dans les chapitres précédents, nous avons introduit les concepts de code topologique et de décodeur logique. Dans le prochain chapitre, nous attaquerons le problème du décodage des codes topologiques à proprement parler. Pour l'instant, contentons-nous d'étudier de manière générale certaines méthodes utiles en physique statistique et profitons-en pour faire le pont entre celles-ci et des méthodes numériques utilisées en informatique qui nous seront utiles plus tard. Les deux principales méthodes dont nous userons sont inspirées des groupes de renormalisation et de la propagation de croyances. Le but de ce chapitre est d'introduire ces méthodes pour en donner la saveur, sans toutefois être trop technique.

4.1 Renormalisation

Dans cette section, on tentera de donner une impression de ce qu'est la renormalisation plutôt que de la traiter rigoureusement. Pour ce faire, considérons un exemple. Prenons le cas d'un réseau 2D de spins classiques, muni d'un hamiltonien H. On sait que la physique d'un tel système peut être comprise à travers sa fonction de partition $\mathcal{Z} = \sum_{\vec{\tau}} \mathrm{e}^{-\beta H(\vec{\tau})}$ où $\vec{\tau}$ décrit l'état des spins.

Découpons mentalement ce réseau en cases carrées 2×2 comme le montre la figure 4.1. Ensuite, sommons sur les degrés de liberté de tous les spins qui ne sont pas dans le coin supérieur gauche d'une case. On appellera $\vec{\tau}'$ le vecteur décrivant l'état du reste des spins. Ceci élimine tous les degrés de liberté sauf un par case. On se retrouve alors avec la somme suivante à effectuer : $\mathcal{Z} = \sum_{\vec{\tau}'} \left(\sum_{\vec{\tau} \setminus \vec{\tau}'} \mathrm{e}^{-\beta H(\vec{\tau})} \right)$. On tire un hamiltonien effectif, H', pour le nouveau réseau renormarlisé en posant $\mathcal{Z} = \sum_{\vec{\tau}'} \mathrm{e}^{-\beta H'(\vec{\tau}')}$. En effet, en

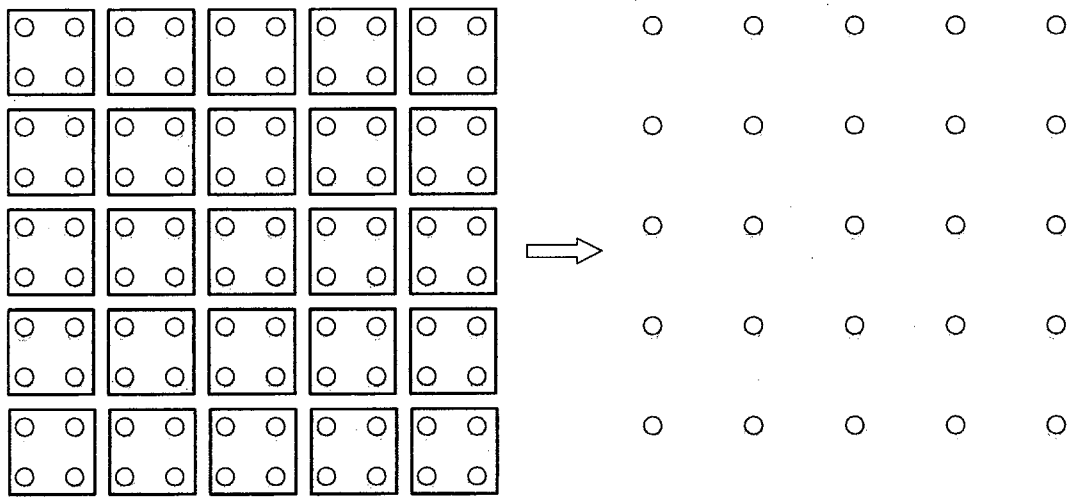


FIGURE 4.1 – Renormalisation : On découpe le réseau de spins en régions carrées. Puis on somme sur certains degrés de liberté. Ceci résulte en un réseau et un hamiltonien renormalisés.

comparant les deux expressions et en prenant le logarithme des arguments, on obtient : $-\beta H'(\vec{\tau}') = \log(\sum_{\vec{\tau}\setminus\vec{\tau}'} \mathrm{e}^{-\beta H(\vec{\tau})})$.

La renormalisation concerne l'étude de cette évolution du hamiltonien à mesure qu'on somme sur les degrés de liberté du système. Cela permet d'en extraire les propriétés physiques. Par exemple, si H ne change pas, on dit qu'il s'agit d'un point fixe de la renormalisation. Dans ce cas, le hamiltonien est invariant d'échelle et cela implique que la longueur de corrélation pour H est infinie. En effet, étant donné n'importe quels deux points d'un réseau infini, il existe une échelle où ceux-ci sont voisins.

L'aspect qui nous intéressera le plus de la renormalisation est cette possibilité de considérer le réseau à plusieurs échelles différentes. En effet, pensons au code torique. On sait que les qubits logiques sont encodés dans des opérateurs s'étendant sur tout le réseau, donc à grande échelle. On s'est inspiré de la renormalisation pour inventer une méthode où on peut renormaliser de petites régions du code de manière itérative pour distiller toujours plus d'information à propos du réseau à grande échelle, où vivent les opérateurs logiques. Cette méthode, présentée au chapitre 5, permet de passer, par plusieurs étapes, de notre réseau de $2\ell^2$ qubits à un réseau effectif de seulement huit qubits, ceux-ci caractérisant la topologie du réseau et donc, l'information qui y est encodée.

4.2 Propagation de croyance

Pour bien comprendre ce qu'est la propagation de croyance, il est plus pratique de débuter avec un exemple connu et compris d'où émergera naturellement la méthode. Notre exemple est tiré de la physique statistique. En effet, considérons une chaîne finie de n spins $1/2$ classiques. Le hamiltonien de notre chaîne est simple : une somme de termes locaux, $h_i(\tau_i)$, et de couplages aux plus proches voisins, $h_{i,i+1}(\tau_i, \tau_{i+1})$. On note par τ_i la valeur ± 1 représentant l'état *up* ou *down* du spin i et par $\vec{\tau}$ le vecteur contenant la valeur de l'ensemble des n spins. On a

$$H(\vec{\tau}) = \sum_{i=1}^{n} h_i(\tau_i) + \sum_{i=1}^{n-1} h_{i,i+1}(\tau_i, \tau_{i+1}). \tag{4.1}$$

Pour caractériser ce système, il est nécessaire de calculer sa fonction de partition $\mathcal{Z}$. Or, on sait que celle-ci vaut de manière générale $\mathcal{Z} = \sum_{\vec{\tau}} \mathrm{e}^{-\beta H(\vec{\tau})}$. Typiquement, il faut sommer sur les 2^n configurations possibles pour le vecteur $\vec{\tau}$, ce qui rend la fonction de partition incalculable en général. Toutefois, en explicitant le hamiltonien décrit ci-haut dans cette formule et en exploitant les propriétés de l'exponentielle, on trouve

$$\mathcal{Z} = \sum_{\vec{\tau}} \mathrm{e}^{-\beta H(\vec{\tau})} = \sum_{\vec{\tau}} \mathrm{e}^{-\beta \sum_i h_i(\tau_i) + h_{i,i+1}(\tau_i, \tau_{i+1})} = \sum_{\vec{\tau}} \prod_i \mathrm{e}^{-\beta(h_i(\tau_i) + h_{i,i+1}(\tau_i, \tau_{i+1}))}. \tag{4.2}$$

Or, par distributivité du produit, on peut récrire cette dernière expression comme suit

$$\sum_{\tau_n} \mathrm{e}^{-\beta h_n(\tau_n)} \sum_{\tau_{n-1}} \mathrm{e}^{-\beta(h_{n-1}(\tau_{n-1}) + h_{n-1,n}(\tau_{n-1}, \tau_n))} \cdots \sum_{\tau_i} \mathrm{e}^{-\beta(h_i(\tau_i) + h_{i,i+1}(\tau_i, \tau_{i+1}))} \cdots \sum_{\tau_1} \mathrm{e}^{-\beta(h_1(\tau_1) + h_{1,2}(\tau_1, \tau_2))}. \tag{4.3}$$

Une fois cette forme atteinte, on note déjà que le calcul est factorisé, c'est-à-dire qu'on peut d'abord sommer sur τ_1, puis τ_2, etc., jusqu'à τ_n. On a alors à calculer $2n$ valeurs plutôt que 2^n. Pour mieux voir cette factorisation, définissons les fonctions suivantes,

qu'on appellera messages.

$$\begin{aligned}
m_{1\to 2}(\tau_2) &= \sum_{\tau_1} \mathrm{e}^{-\beta(h_1(\tau_1)+h_{1,2}(\tau_1,\tau_2))}, \\
m_{2\to 3}(\tau_3) &= \sum_{\tau_2} m_{1\to 2}(\tau_2)\mathrm{e}^{-\beta(h_2(\tau_2)+h_{2,3}(\tau_2,\tau_3))}, \\
&\vdots \\
m_{i\to i+1}(\tau_{i+1}) &= \sum_{\tau_i} m_{i-1\to i}(\tau_i)\mathrm{e}^{-\beta(h_i(\tau_i)+h_{i,i+1}(\tau_i,\tau_{i+1}))}, \\
&\vdots \\
m_{n-1\to n}(\tau_n) &= \sum_{\tau_{n-1}} m_{n-2\to n-1}(\tau_n - 1)\mathrm{e}^{-\beta(h_{n-1}(\tau_{n-1})+h_{n-1,n}(\tau_{n-1},\tau_n))}.
\end{aligned}$$

On voit qu'à l'aide de ces définitions, on a l'identité suivante :

$$\mathcal{Z} = \sum_{\tau_n} m_{n-1\to n}(\tau_n)\mathrm{e}^{-\beta h_n(\tau_n)}. \tag{4.4}$$

Dans le cas de la chaîne 1D, la propagation de croyance se réduit à la méthode de la matrice de transfert. Grâce au fait que le hamiltonien est constitué de termes locaux 1D, on a pu factoriser le calcul de la fonction de partition qui en général croît de manière exponentielle avec n. De plus, grâce aux fonctions messages qu'on a définies, on peut donner une interprétation de ce calcul qui se décompose en une séquence de calculs simples. Dans un premier temps, on calcule le message concernant le spin d'un bout de la chaîne, i.e. on calcule $m_{1\to 2}$. Cette information est en fait un message transmis au deuxième site. On peut ensuite l'utiliser pour calculer l'impact du deuxième spin. Ceci résultera dans la création du message $m_{2\to 3}$, qui sera transmis au spin suivant, etc. On peut s'imaginer que chaque site est muni d'un processeur effectuant un calcul. Le processeur i reçoit un message du processeur $i-1$, à l'aide duquel il calcule le message $m_{i\to i+1}$ qu'il enverra au processeur $i+1$. On voit naturellement apparaître ce que l'on appelle la propagation de croyance (*belief propagation*) [35].

Comme on le sait de la physique statistique, une fois la fonction de partition calculée, on peut avoir accès à plusieurs quantités utiles, telles les distributions de probabilité marginales sur n'importe quel spin ou paire de spins, etc.

On généralise la méthode à un graphe quelconque. On associe toujours un processeur à chaque site. Chaque processeur reçoit un message de chacun de ses voisins. À partir de

ceux-ci, il calcule de nouveaux messages selon la règle suivante, généralisation naturelle du cas 1D,

$$m_{i\to j}(\tau_j) = \sum_{\tau_i} \mathrm{e}^{-\beta(h_i(\tau_i)+h_{i,j}(\tau_i,\tau_j))} \prod_{j'\in v(i)\setminus j} m_{j'\to i}(\tau_i). \tag{4.5}$$

où $v(i)$ est l'ensemble des sites connectés à i. On peut montrer que cet algorithme sera exact tant que l'on considère un hamiltonien pouvant être représenté par un arbre, i.e. ne contenant aucun cycle [35]. La chaîne 1D est un cas particulier d'arbre.

La méthode de la propagation de croyance est exacte pour des arbres, mais qu'en est-il des graphes quelconques ? Dans ce cas, il n'existe aucune garantie que cet algorithme fonctionne. Dans ce contexte, il s'agit d'une méthode heuristique. Toutefois, cette méthode heuristique a été utilisé à plusieurs reprises et a souvent donné de bons résultats [25]. De plus, on peut comprendre en termes plus physiques ce qu'elle fait. Prenons encore une fois un exemple de la physique statistique qui nous aidera à comprendre.

Considérons un système de spins, disons un réseau 2D carré. Comme dans le cas de la chaîne, le hamiltonien du système est constitué de termes locaux et de couplages plus proches voisins.

$$H = \sum_i h_i(\tau_i) + \sum_{\langle i,j\rangle} h_{i,j}(\tau_i,\tau_j), \tag{4.6}$$

où $\langle\cdot,\cdot\rangle$ signifie « voisins ».

On s'intéresse à minimiser son énergie libre de Gibbs, $F(P) = E(P) - TS(P)$, par rapport à la distribution $P(\vec{\tau})$ sur tous les spins du réseau. Étant donné que le hamiltonien n'est constitué que de termes locaux, il est toujours simple de calculer l'énergie du système, c'est-à-dire en ne considérant que les probabilités marginales à un et deux spins, comme dans le cas de la chaîne discuté ci-haut. Toutefois, l'entropie est une fonctionnelle plus complexe de $P(\vec{\tau})$ et en général, on ne peut pas en factoriser le calcul comme pour l'énergie. L'énergie libre est alors très difficile à calculer explicitement. En effet, on connaît déjà la solution en principe, il s'agit de la distribution de Gibbs : $P(\vec{\tau}) = \frac{1}{Z}\mathrm{e}^{-\beta H(\vec{\tau})}$. C'est de la calculer explicitement qui est difficile. Par exemple, si on désirait obtenir la distribution marginale sur le spin i, $P(\tau_i) = \frac{1}{Z}\sum_{\vec{\tau}\setminus\tau_i} \mathrm{e}^{-\beta H(\vec{\tau})}$, la solution générale (Gibbs), ne nous aide pas, car on doit faire la somme sur un nombre de termes exponentiellement grand par rapport à n, le nombre total de spins. En 1D, la propagation de croyance nous

donne la réponse directement et en général, on a

$$P(\tau_i) = \frac{1}{\mathcal{Z}} \mathrm{e}^{-\beta h_i(\tau_i)} \prod_{j \in v(i)} m_{j \to i}(\tau_i). \tag{4.7}$$

Dans le but d'arriver efficacement à une approximation de F, on peut la remplacer par une expression contenant seulement une approximation de l'entropie. Celle-ci ne devrait dépendre elle aussi que des probabilités marginales sur de petites régions, permettant ainsi la factorisation de son calcul. Il s'agit de l'énergie libre de Bethe,

$$F^*(P) = E(P) - TS^*(P), \qquad S^* = \sum_{r \in R} S_r^*(P_r), \tag{4.8}$$

où on a découpé notre réseau en un ensemble de régions, $\{R\}$, significativement plus petites que la taille du réseau considéré. On définit alors S_r^* comme étant l'entropie restreinte à la région r. Ces nouvelles entropies ne font intervenir que les probabilités marginales sur les régions, simplifiant grandement le problème de minimisation de l'énergie libre. Pour simplifier la discussion, on prend à partir de maintenant R comme étant l'ensemble des paires de spins voisins. Les $S_{i,j}^*(P_{i,j})$ sont alors les entropies restreintes à toutes les paires (i,j) de spins voisins. $P_{i,j}$ représente la distribution de probabilité sur les spins i et j. Alors, F^* ne fait intervenir que des probabilités à un ou deux corps ; elle est donc plus facile à manipuler. On peut tenter de minimiser cette dernière plutôt que de s'attaquer à F directement :

$$\min_P E(P) - TS^*(P) = \min_{\{P_{i,j}\}} \sum_{\langle i,j \rangle} \left(E(P_{i,j}) - TS_{i,j}^*(P_{i,j}) \right). \tag{4.9}$$

C'est alors qu'intervient la propagation de croyance. Yedidia *et al.* ont montré qu'en utilisant un algorithme de propagation de croyance comme celui présenté ci-haut, on pouvait trouver l'ensemble des probabilités marginales minimisant l'énergie libre de Bethe [35]. En effet, si les messages convergent, alors ce point fixe de l'algorithme conduit à un minimum de F^*.

De plus, ce même résultat nous assure une auto-cohérence des probabilités marginales obtenues. Les régions du type $S_{i,j}^*$ et $S_{j,k}^*$ se chevauchent sur le spin j. Après minimisation, chaque région obtient une marginale optimale qui lui est propre, $p_{i,j}$ et $p_{j,k}$, et leur restriction respective au spin partagé, p_j^i et p_j^k (l'indice du haut représente le spin sur

lequel on a sommé), seront en général différentes. La propagation de croyance nous assure que cela ne sera pas le cas, i.e. $p_j^i = p_j^k$, pour toutes paires de régions partageant un spin. Cette auto-cohérence améliore d'emblée la qualité de l'approximation. En termes plus physiques, on peut dire que cela revient à imposer des conditions de type champ moyen entre les régions, là où elles se chevauchent.

4.3 Décodeur physique

D'après ce qui a été expliqué ci-haut, on peut comprendre pourquoi la propagation de croyance est une bonne méthode heuristique pour calculer des probabilités marginales sur des qubits ou des régions. Or, dans le cadre plus général des codes stabilisateurs et de leur décodage, on a vu que calculer des probabilités marginales sur des qubits ou des régions consistait en du décodage physique. La propagation de croyance peut donc être utilisée comme décodeur physique.

Dans ce contexte, cette méthode consiste à permettre aux générateurs du stabilisateur et aux qubits sur lesquels ils agissent de s'échanger des messages, que l'on nomme croyances (*beliefs*), au sujet d'erreurs potentielles afin de déterminer l'erreur marginale sur chaque qubit : $\{P_i(e)\}$ où $e \in \mathcal{P}_1$.

Au départ, la croyance des qubits, q, est limitée à $P_q(e)$ comme définie par le modèle de bruit. Chaque qubit, q, enverra un message, m, aux stabilisateurs, s, agissant sur lui (cf. Fig. 4.2),

$$m_{q\to s}(e) = P_q(e). \tag{4.10}$$

Ensuite, les stabilisateurs, s, utilisent ces messages pour calculer leur propre croyance qu'ils enverront aux qubits sur lesquels ils agissent (cf. Fig. 4.3),

$$m_{s\to q}(e_q) \propto \sum_{\{e_{q'}:q'\in v(s)\setminus q\}} \delta_{t_i,\prod_{q'\in v(s)}\{s,e_{q'}\}} \prod_{q'\in v(s)\setminus q} m_{q'\to s}(e_{q'}), \tag{4.11}$$

où $v(s)$ est l'ensemble des qubits sur lesquels agit s. Le delta de Kronecker sert à imposer la contrainte que les erreurs doivent être en accord avec le syndrome observé, $t_i \in \{-1, 1\}$.

Puis, les qubits peuvent recalculer leur croyance à la lumière des nouveaux messages

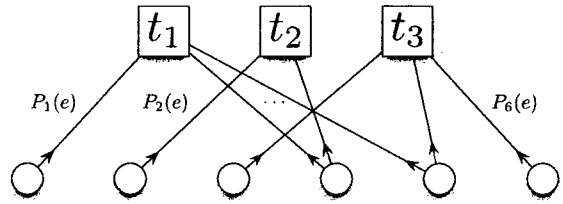


FIGURE 4.2 – Les qubits envoient $P_q(e)$ aux stabilisateurs. Les boîtes représentent les générateurs du stabilisateur et les cercles, les qubits. Les arêtes mettent en relation les générateurs et les qubits sur lesquels ils agissent.

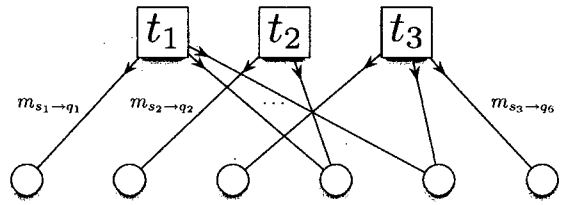


FIGURE 4.3 – Les générateurs du stabilisateur calculent leur croyance et l'envoient à leurs qubits.

reçus,

$$m_{q \to s}(e) \propto P_q(e) \prod_{s' \in v(q) \setminus s} m_{s' \to q}(e), \tag{4.12}$$

où $v(q)$ est l'ensemble des générateurs du stabilisateur auquel participe q de manière non-triviale.

On peut répéter cet échange autant de fois que nécessaire. Pour terminer, les qubits calculent leur croyance finale qui tient compte du modèle de bruit et de tous leurs messages entrants,

$$b_q(e) \propto P_q(e) \prod_{s \in v(q)} m_{s \to q}(e). \tag{4.13}$$

Toutefois, il n'est pas garanti que la valeur des messages converge. C'est pourquoi cette méthode n'est pas valable en général. Elle fonctionne dans certains cas, par exemple pour les codes LDPC (dont on ne discutera pas ici). Malgré qu'il ne puisse être utilisé dans le

cas général, comme nous le verrons au chapitre 5, ce décodeur peut être couplé à d'autres, un décodeur logique par exemple, pour augmenter les performances de ce dernier.

Complexité

Le décodeur logique est sous-optimal, par rapport à un décodeur logique, mais il est très efficace. En effet, il fait intervenir les m générateurs du stabilisateur plutôt que tous ses 2^m éléments. Comme chacun des messages peut être calculé indépendamment du calcul des autres, on peut paralléliser. Cela permet de réduire le temps de décodage à une constante.

Chapitre 5

Renormalisation : théorie et pratique

Dans les chapitres précédents, nous avons introduit les concepts de codes topologiques et de décodeurs logiques ainsi que quelques outils de la physique statistique. Dans ce qui suit, nous mettons ces outils à l'oeuvre et présentons un algorithme qui approxime le décodeur logique pour des codes topologiques. Pour le tester, nous nous concentrons sur le code de Kitaev, mais à priori, rien n'empêche qu'on l'applique à d'autres codes topologiques. L'algorithme dans sa forme finale est composé de trois modules dont deux sont imbriqués. Pour en faciliter la compréhension, nous analyserons dans un premier temps seulement le premier module qui constitue le coeur de l'algorithme. Ensuite, nous ajouterons successivement le deuxième puis le troisième module. Nous testerons au fur et à mesure qu'on ajoute des modules les performances du décodeur obtenu. Plusieurs de ces résultats sont déjà parus dans la publication [11].

5.1 Renormalisation du réseau

5.1.1 Intuition

Le premier module est inspiré d'un cas connu et compris, mais qui ne s'applique pas directement aux codes topologiques. C'est pourquoi la renormalisation du réseau est en fait une méthode heuristique. L'intuition derrière cette méthode est de considérer le code de Kitaev comme un code concaténé, malgré que ce ne soit pas à proprement parler le cas. Si on était doté d'un code ayant l'allure de celui dépeint à la figure 5.1, on pourrait

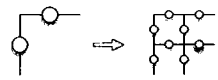


FIGURE 5.1 – Code $n = 8$ et $k = 2$ qui permettrait la concaténation. N.B. : Il s'agit d'un code de surface, il n'y a pas de conditions périodiques.

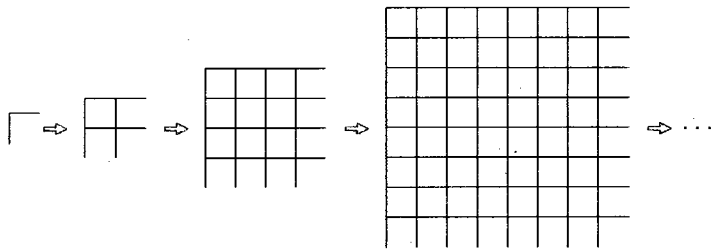


FIGURE 5.2 – Intuition derrière l'approche renormalisation du réseau. Chaque cellule unitaire, représentant deux qubits, est ré-encodée en utilisant le code montré à la figure 5.1 et ce, de manière itérative.

peut-être constuire un code torique par concaténation. Remarquons que le code présenté à la figure 5.1 n'est pas un code de Kitaev. Il s'agit bien d'un code topologique, mais il s'agit d'un code de surface. Il n'y a pas de conditions de frontières périodiques. Les stabilisateurs ne sont pas nécessairement les mêmes que pour le code de Kitaev, i.e. pas nécessairement des plaquettes et des croix. La figure 5.2 illustre l'idée de bâtir le code de Kitaev comme une concaténation du code de surface discuté. À chaque passe, on encode chacune des cellules unitaires à la l'aide du code de surface. Chacune d'elles passe de deux qubits à huit. Au total, le nombre de qubits est multiplié par quatre à chaque passe. Si après un certain nombre de niveaux d'encodage la résultante de la concaténation était équivalente à un encodage avec un code de Kitaev au ℓ approprié, on pourrait alors utiliser un décodeur logique de codes concaténés (cf. Chapitre 2) qui nous permettrait de faire le chemin inverse, décodant le code torique de manière itérative comme le montre la figure 5.3. Bref, on aurait une procédure pour décoder efficacement le code torique.

Toutefois, la structure présentée ci-haut ne donnera jamais un code de Kitaev, car elle ne produit pas les stabilisateurs désirés. En effet, on voit que deux plaquettes et deux croix ne sont contenues que partiellement dans notre code à huit qubits comme le montre la figure 5.4.

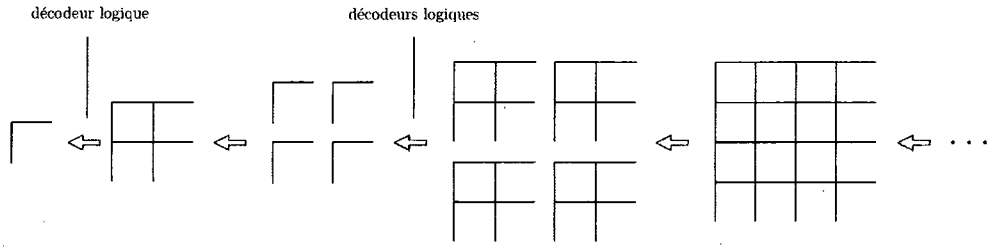


FIGURE 5.3 – Décodeur potentiel pour le code torique vu comme un code concaténé. Les décodeurs logiques remontent en sens inverse le schéma de la figure 5.1 passant de huit qubits physiques à deux qubits logiques pour chaque cellule unitaire.

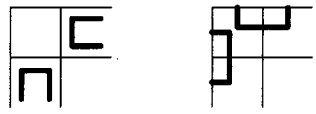


FIGURE 5.4 – Certains stabilisateurs du code de surface sont incompatibles avec le code de Kitaev, car ceux-ci sont incomplets. En effet, on voit qu'un qubit est manquant à deux plaquettes (gauche) ainsi qu'à deux croix (droite). Les traits verts représentent des opérateurs Z et les traits bleus, des opérateurs X.

5.1.2 Sous-Code

D'après la structure présentée ci-haut, nous désirons un protocole permettant de débuter avec huit qubits et d'en distiller deux de façon à obtenir une structure globale invariante d'échelle. Il faut donc six contraintes. Or, la cellule unitaire qu'est le code de surface ne compte que deux stabilisateurs complets comme le montre la figure 5.5. Nous choisissons donc d'ajouter à notre cellule unitaire quatre qubits additionnels afin de compléter les quatre débuts de stabilisateurs illustrés à la figure 5.4. En conséquence, nous obtenons des cellules unitaires qui se recouvrent ; les quatre qubits ajoutés sont partagés entre deux cellules. Chaque sous-code emprunte un qubit à chacun de ses quatre voisins en les dupliquant (cf. Fig. 5.9). Le code de surface ainsi obtenu est illustré à la figure 5.6 et on l'appellera sous-code. Toutefois, cela brise la structure en arbre caractéristique des codes concaténés et nous n'en avons plus exactement un, car les sous-codes ne sont plus indépendants.

Avant d'aller plus loin au sujet de l'algorithme de renormalisation, introduisons les détails du code de surface à 12 qubits qu'est le sous-code. Les 10 générateurs du stabilisateur ainsi que leur erreur pure respective sont présentés au tableau 5.1 et illustrés à

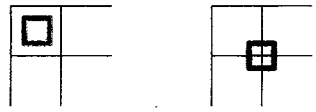


FIGURE 5.5 – stabilisateurs du code de surface compatibles avec le code de Kitaev : une plaquette (gauche) et une croix (droite). Les traits verts représentent des opérateurs Z et les traits bleus, des opérateurs X.

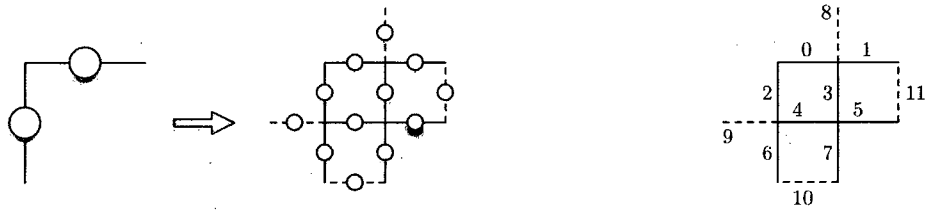


FIGURE 5.6 – À gauche, complétion du code présenté à la figure 5.1 dans le but d'obtenir les stabilisateurs désirés. Les qubits complémentaires sont représentés en pointillé. À droite, on voit la convention utilisée dans le texte pour parler des qubits du sous-code.

la figure 5.7. Il s'agit essentiellement du code à huit qubits introduit ci-haut auquel sont greffés les quatre qubits nécessaires à la complétion des croix et des plaquettes. On a donc $n = 12$ qubits physiques, $k = 2$ qubits logiques et $m = 10$ qubits stabilisateurs. Les opérateurs logiques sont présentés au tableau 5.2 et illustrés à la figure 5.8.

	1	2	3	4	5	6
s	$Z_0Z_2Z_3Z_4$	$Z_1Z_3Z_5Z_{11}$	$Z_4Z_6Z_7Z_{10}$	$X_3X_4X_5X_7$	$X_2X_4X_6X_9$	$X_0X_1X_3X_8$
t	X_3X_5	X_5	X_7	Z_0Z_3	Z_2	Z_0

	7	8	9	10
s	Z_{10}	Z_{11}	X_8	X_9
t	X_7X_{10}	X_5X_{11}	Z_0Z_8	Z_1Z_9

TABLEAU 5.1 – stabilisateurs et erreurs pures du sous-code

5.1.3 Décoder

Un sous-code étant de taille finie et raisonnable, on peut le décoder à l'aide d'un décodeur logique doux qui calculera, de façon exacte par force brute, l'énergie libre de chaque classe logique de manière cohérente avec le syndrome (cf. chapitre 2). Ce décodeur

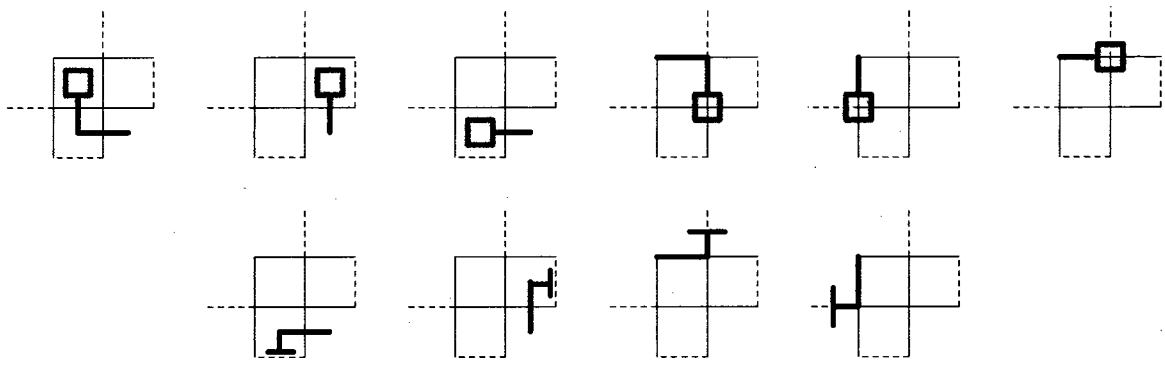


FIGURE 5.7 – Le sous-code est spécifié par 10 stabilieurs et 10 erreurs pures décrits au tableau 5.1. Les blocs du haut présentent trois plaquettes et trois croix analogues à celles vues dans le cas du code de Kitaev ainsi que leur erreur pure respective. Les blocs du bas illustrent les générateurs de l'algèbre des qubits empruntés. Comme ils ne sont là que pour permettre la complétion des plaquettes et des croix, on gardera une base locale des stabilisateurs sur ces qubits. Ils ne participent qu'indirectement à l'élaboration des qubits logiques délocalisés. Les traits verts représentent des opérateurs Z et les traits bleus, des opérateurs X.

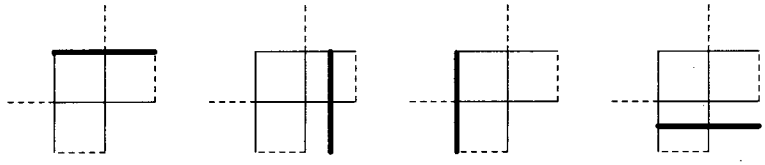


FIGURE 5.8 – Le sous-code contient deux qubits logiques dont voici les générateurs. Ils sont décrits au tableau 5.2. Les traits verts représentent des opérateurs Z et les traits bleus, des opérateurs X.

	0	1
$\overline{Z}$	Z_0Z_1	Z_2Z_6
$\overline{X}$	X_1X_5	X_6X_7

TABLEAU 5.2 – Opérateurs logiques du sous-code

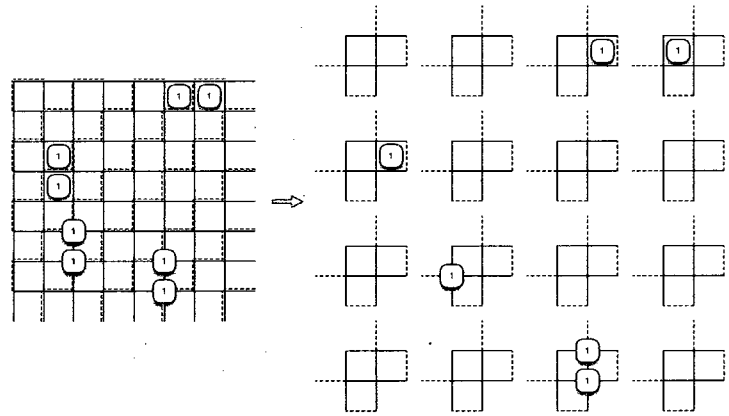


FIGURE 5.9 – Découpe du code torique en plusieurs petits codes de surfaces. Les qubits en pointillé sont dupliqués, ils apparaissent déjà dans un code voisin. Le syndrome soumis à chaque sous-code est une restriction du syndrome total aux stabilisateurs contenus dans le sous-code en question. L'information contenue dans les syndromes manquants sera utilisée dans les étapes de renormalisation subséquentes, comme le montre les figures 5.10, 5.11, 5.12 et 5.13.

prendra en entrée le modèle de bruit et le syndrome restreint au sous-code (cf. Fig. 5.9). Notons que les stabilisateurs des qubits empruntés ne sont pas mesurés dans le cadre du code de Kitaev. L'erreur pure associée à ces stabilisateurs n'est pas spécifiée. On notera T_e le groupe généré par les erreurs pures de qubits empruntés : $T_e = \langle t_7, t_8, t_9, t_{10} \rangle$. Le décodeur logique doit donc aussi sommer sur les différentes erreurs pures de ces quatre qubits, i.e. ils participent aussi à l'entropie de l'erreur d'un sous-code J :

$$S_J(t_J, l) = -\sum_{s \in \mathcal{S}, t_e \in T_e} P(t_J t_e l s) \ln(P(t_J t_e l s)), \tag{5.1}$$

où t_J est le syndrome mesuré restreint au sous-code J.

Après avoir calculé l'énergie libre des différentes classes logiques, le décodeur donne en sortie la distribution de probabilité sur les deux qubits logiques contenus dans le sous-

code. On est donc passé de huits qubits (quatre plus huit partagés entre deux sous-codes) à deux qubits, d'où la renormalisation. Ce procédé est l'inverse de celui illustré à la figure 5.6. L'ensemble des sorties sert ensuite de canal de bruit effectif au réseau renormalisé qui peut lui-même être décodé de manière analogue et ainsi de suite, comme on l'a vu pour des décodeurs de codes concaténés.

Après suffisamment d'étapes de renormalisation, nous obtenons un réseau de taille 2×2. À ce point, il ne reste plus qu'à appliquer un décodeur logique dur au code de Kitaev de taille deux pour obtenir la classe logique d'erreurs la plus probable sur les deux qubits qui correspondent aux deux qubits encodés au départ dans le code torique, complétant le décodage.

5.1.4 Syndrome

Étant motivés par la structure de renormalisation du réseau présentée ci-haut, nous ne mesurerons pas les générateurs habituels du stabilisateur. Certains d'entre eux seront modifiés pour convenir à notre structure. En effet, notons que les plaquettes du code torique qui correspondent au coin inférieur droit d'un sous-code et les croix correspondant au coin supérieur gauche n'apparaissent pas dans le sous-code (cf. Fig. 5.9). L'information contenu dans ces stabilisateurs ne sera utilisée qu'à une étape subséquente de la renormalisation. Ceux-ci seront donc modifiés en conséquence. La figure 5.13 illustre ce changement. À chaque couche du décodeur correspond une taille de plaquettes et de croix (e.g. 1,2,4,8, etc.). Les figures 5.10, 5.11 et 5.12 illustrent cette idée en poursuivant l'instance introduite à la figure 5.9 : la première couche contient l'information des stabilisateurs de taille un, alors que la deuxième couche contient l'information des stabilisateurs de taille deux et la troisième, celle des stabilisateurs de taille quatre.

5.1.5 Simulations numériques

Pour savoir si l'algorithme proposé est bien un décodeur, on doit le tester pour vérifier s'il présente une transition de phase topologique pour un certain niveau de désordre. Cela se traduit en l'observation d'un seuil qui est la force du bruit à laquelle se produit la transition *phase désordonnée - phase topologique* si elle existe.

Pour tester notre algorithme, on utilise une approche Monte Carlo. On génère à l'aide d'un générateur de nombres pseudo-aléatoires une erreur correspondant au canal de bruit d'intérêt, ici le canal dépolarisant. À partir de cette erreur, on calcule ce que devrait être

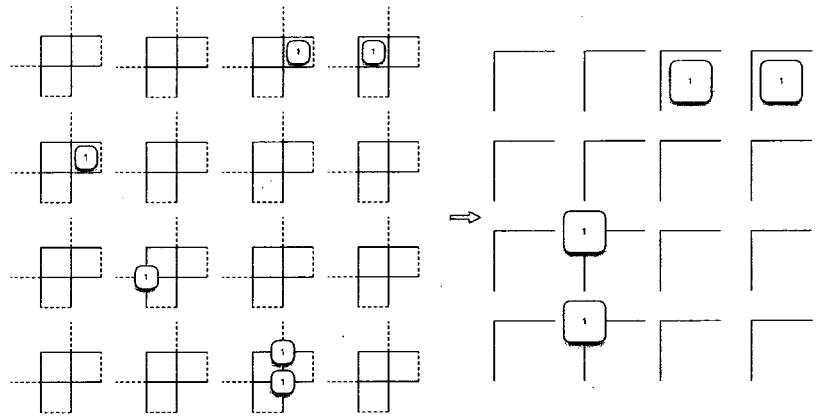


FIGURE 5.10 – Premier niveau de décodage

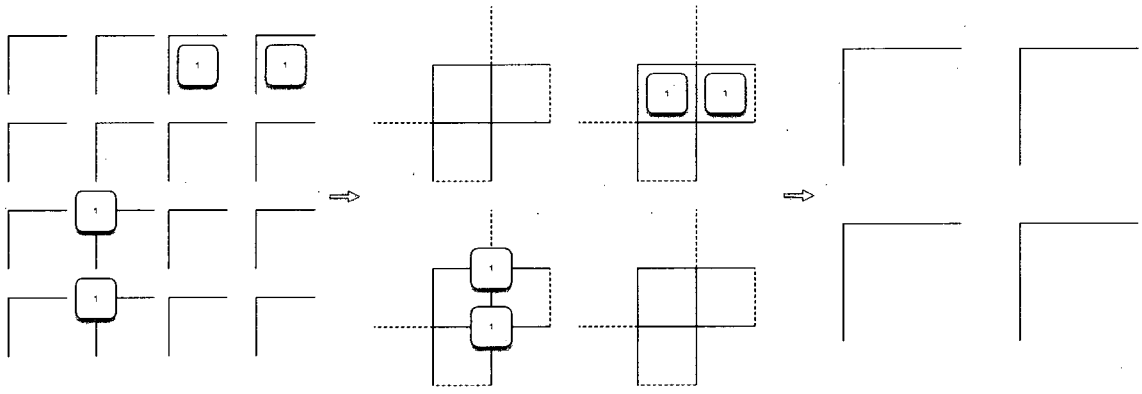


FIGURE 5.11 – Deuxième niveau de décodage

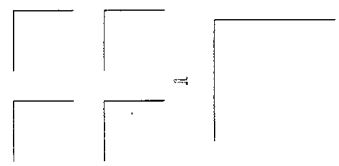


FIGURE 5.12 – Dernier niveau de décodage. On applique un décodeur logique au code de Kitaev 2×2 pour obtenir une distribution sur les deux qubits logiques encodés au départ.

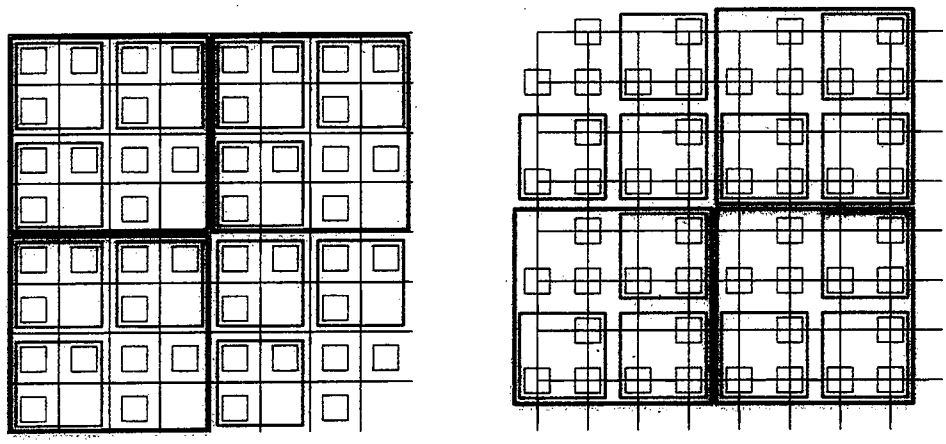


FIGURE 5.13 – Nouvel ensemble de générateurs du stabilisateur à mesurer pour le décodeur de renormalisation. Les stabilisateurs de taille 2^k seront utilisés dans le niveau k du décodage. N.B. : On peut mesurer l'ensemble habituel et, à partir des résultats, calculer ce qui aurait été obtenu si on avait plutôt mesuré ce nouvel ensemble. Il ne s'agit que d'un changement de base.

le syndrome et on le soumet au décodeur. Celui-ci transmet en sortie la classe d'erreur qu'il juge la plus probable. On vérifie ensuite si l'erreur produite appartient à cette classe. Si oui, c'est un succès, sinon, c'est une erreur de décodage.

On répète cette procédure un grand nombre de fois (10^4 ou plus) afin d'accumuler des statistiques. On divise ensuite le nombre de mauvaises corrections par le nombre d'essais, ce qui nous donne une estimation de la probabilité d'erreur du décodeur.

Après quelques simulations préliminaires, la méthode de renormalisation ne présente pas de seuil bien défini, comme le montre la figure 5.14. En effet, les trois courbes ne se croisent pas au même point, qui aurait été le taux de bruit à partir duquel encoder dans des réseaux de plus en plus grands améliore la tolérance au bruit.

On observe plutôt un effet de taille important, qui fait en sorte que les points de croisement fuient vers la gauche. Il est concevable que cet effet de taille sature et que pour les grands réseaux, la probabilité d'erreur de décodage diminue lorsque le taux de bruit est en dessous d'un seuil bien défini. Nous n'avons pas poussé dans cette direction puisque nous pouvons améliorer le décodeur sans augmenter sa complexité de façon significative. Si on poursuivait dans cette voie, on s'attendrait à trouver le seuil autour d'une force du canal dépolarisant de 7%.

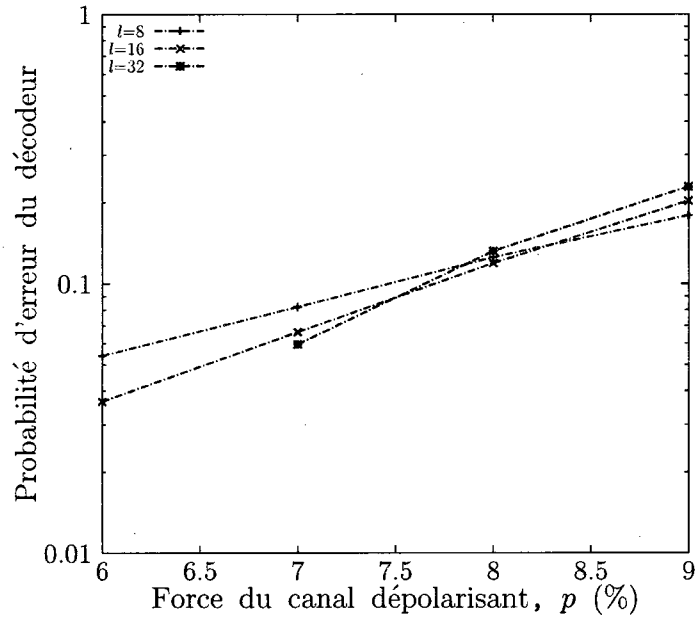


FIGURE 5.14 – Probabilité d'erreur du décodeur de renormalisation en fonction de la force du canal dépolarisant pour les tailles de réseau $\ell = 8$, 16 et 32. Chaque point a été déterminé par un banc d'essai de 10^4 erreurs aléatoires. Le seuil n'est pas bien défini, dû à des effets de taille marqués : les points de croisement fuient vers la gauche.

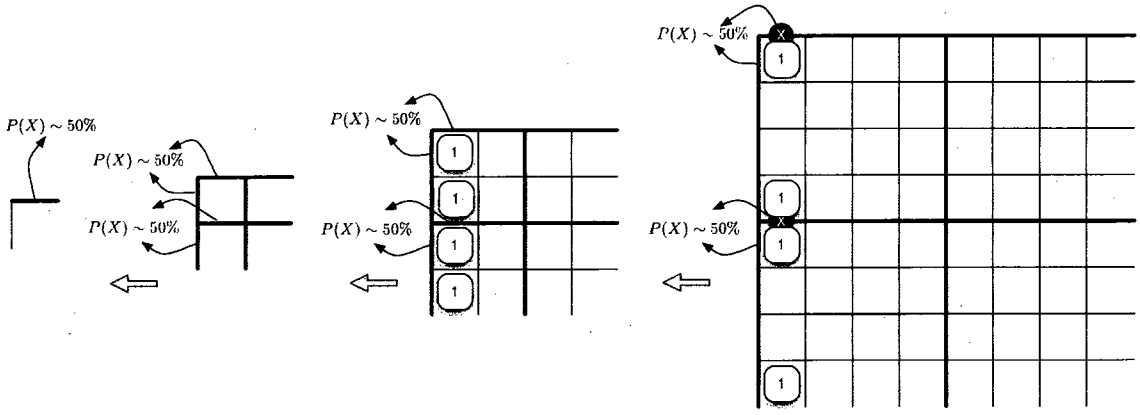


FIGURE 5.15 – Erreur de poids deux déjouant la renormalisation. Si des défauts se concentrent autour des coins marqués par les lignes épaisses, le décodeur n'arrive pas à décoder adéquatement. Augmenter la taille du réseau n'y change rien. Ici, le décodeur n'ayant accès qu'aux syndromes et pas aux erreurs et étant donné la configuration, il n'arrive pas à décider où pourrait se situer l'erreur : l'information est perdue.

De plus, on peut identifier des configurations de défauts de poids fini empêchant le décodage comme le montre la figure 5.15. Évidemment, si une erreur de poids fini empêche le décodeur de fonctionner adéquatement, il est alors inutile d'agrandir le réseau pour s'en prémunir.

Malgré ces problèmes, comme nous le verrons à la section suivante, on peut aisément améliorer cet algorithme significativement.

5.2 Propagation de croyance généralisée

Dans cette section, on utilisera l'algorithme de renormalisation de réseau introduit à la section précédente et on lui ajoutera une nouvelle composante. On permettra aux différents sous-codes d'échanger des messages avant de procéder au décodage logique. On verra que cet ajout a un impact spectaculaire sur les performances du décodeur.

Un problème évident de la méthode de renormalisation est le manque de cohérence dans l'utilisation des qubits partagés. En effet, ceux-ci correspondent à une même entité, le qubit partagé, mais on les traite indépendamment. Rappelons qu'à l'autre extrême, le traitement idéal est celui où on décode en considérant des configurations sur tout le réseau simultanément, mais comme on le sait, ceci conduit à un décodeur logique de

complexité exponentielle.

La solution envisagée est un compromis entre ces deux extrêmes. Nous imposerons des conditions champ moyen aux qubits partagés, i.e. on imposera que la distribution marginale sur les qubits partagés entre blocs voisins soit la même pour chacun desdits blocs, et ce, à chaque étape de renormalisation. Ceci résulte en un ensemble d'équations champ moyen d'auto-cohérence à résoudre.

5.2.1 Résoudre les équations champ moyen

Au chapitre 4, on a vu que la propagation de croyance [35] est une méthode permettant de s'approcher d'une solution aux équations champs moyens (cf. section 4.2 et éqs. 4.8, 4.9). On cherche à modifier les $P_J(t_J ls)$ (J spécifie un sous-code) qui à priori sont données par le modèle de bruit ou la sortie du niveau précédent pour qu'elles soient telles qu'elles minimisent l'énergie libre de Bethe tout en étant cohérentes là où elles se chevauchent, i.e. sur les qubits partagés. Comme cette méthode est heuristique, sa capacité à résoudre le problème ne peut être garantie, mais on sait qu'elle fonctionne dans plusieurs cas. Comme elle est aussi très rapide, on se contentera de la BP (*belief propagation*) pour approximer la probabilité marginale d'un qubit partagé.

Dans ce contexte, les sous-codes s'échangeront des messages correspondants aux probabilités marginales sur les qubits partagés. Le formalisme et les concepts utilisés dans cette section sont très similaires à ceux présentés à la section 4.3. Tout d'abord chaque sous-code calcule l'ensemble des ces probabilités marginales,

$$P_{i,J}(\sigma, t_J) = \sum_{t_e,l,s:(t_J t_e ls)_i=\sigma} P_J(t_J t_e ls), \tag{5.2}$$

où σ représente les différents opérateurs de Pauli à un qubit.

Puis, les sous-codes envoient ces messages à leurs voisins respectifs. La figure 5.16 illustre l'ensemble des messages qu'un sous-code échange avec ses voisins. Les messages reçus par un sous-code seront ensuite utilisés pour modifier sa distribution P_J comme suit,

$$P_J(t_J ls) \to P_J(t_J ls) \prod_i m_{i,J}^{(t_J ls)_i}, \tag{5.3}$$

où les $m_{i,J}$ sont les messages reçus par le sous-code J concernant le qubit i, i.e les

probabilités marginales telles que calculées par les sous-codes voisins. Intuitivement, on peut comprendre qu'un sous-code ajuste le « poids » de ses décisions en fonction des messages qu'il a reçus de ses voisins. Cette nouvelle distribution est ensuite utilisée pour recalculer de nouveaux messages qui seront aussi échangés et cette procédure peut être répétée plusieurs fois. En vérité, l'expression donnée à l'équation 5.2 pour les $P_{i,J}(\sigma, t_J)$ n'est valable que pour la première passe de BP d'une étape de renormalisation. Pour les passes subséquentes, on doit enlever l'information que l'on vient de recevoir d'un voisin avant de lui envoyer un nouveau message pour éviter un effet de *feed-back* pervers,

$$P_{i,J}(\sigma, t_J) = \sum_{t_e,l,s:(t_J t_e l s)_i=\sigma} P_J(t_J t_e l s)/m^{\sigma}_{i,J}. \tag{5.4}$$

Pour donner une intuition du rôle de la BP et de ce qu'elle permet, un exemple informel est illustré à la figure 5.17. Au départ, le sous-code 3 ne connaît que le défaut qu'il contient et ne sait pas où se situe l'erreur. Pour $p \ll 1$, il y a grossomodo une chance sur deux que l'erreur se soit produite au-dessus ou à gauche du défaut. Il en va d'une manière similaire pour le sous-code 1. Toutefois, le sous-code 2 ne contient pas de défauts. La probabilité qu'il soit traversé par une chaîne d'erreurs X est alors $\mathcal{O}(p^2)$. Les blocs s'échangent ensuite ces informations et leur P_J est modifiée en conséquence. À ce moment, le qubit à gauche du défaut du sous-code 3 voit sa probabilité d'erreur diminuer d'ordre $\mathcal{O}(p^2)$ par rapport à celui du dessus. Cet exemple, quoiqu'incomplet, nous donne une bonne intuition de l'effet de la BP, i.e. le sous-code 3 a pu partir d'information partielle le laissant aux prises avec deux principales possibilités et, grâce à la BP, a atteint une quasi-certitude sur ce qui s'est produit.

5.2.2 Simulations numériques

Une fois la méthode de propagation de croyance introduite, nous avons testé la combinaison de la renormalisation et de la BP par Monte Carlo comme précédemment. Les résultats se trouvent à la figure 5.18. On a testé pour une, deux et trois passes de BP. Dès la première passe, on voit apparaître un seuil situé autour de 14.5%. La BP a permis d'obtenir des résultats adéquats pour les configurations de taille finie empêchant l'apparition d'un seuil net pour l'algorithme de renormalisation seul. Pour deux passes, le seuil se déplace vers 15% et pour trois passes, vers 15.2%. Ces seuils sont très comparables à celui du PMA, 15.5%, ce qui est remarquable étant donné la complexité avantageuse de

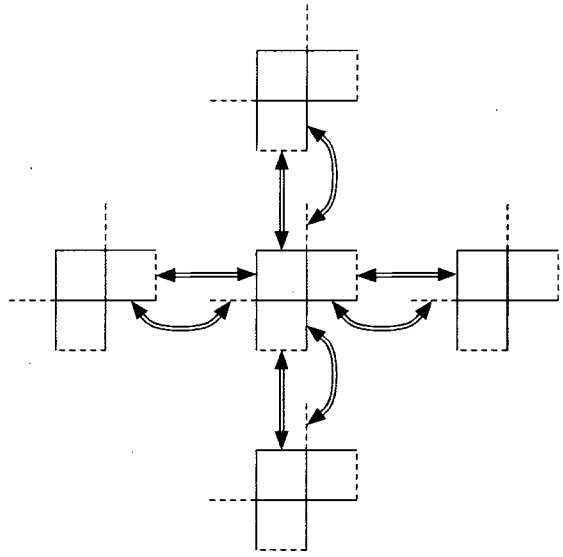
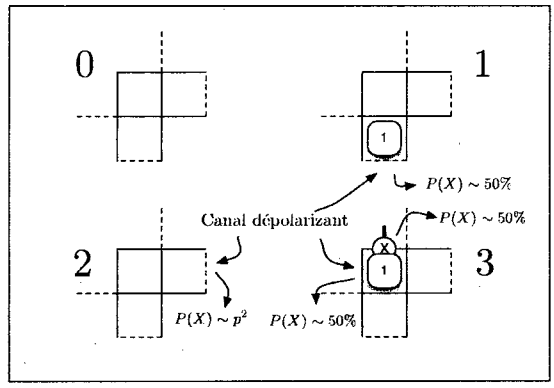


FIGURE 5.16 – Ensemble des messages échangés entre un sous-code et ses voisins : huit entrants et huit sortants.

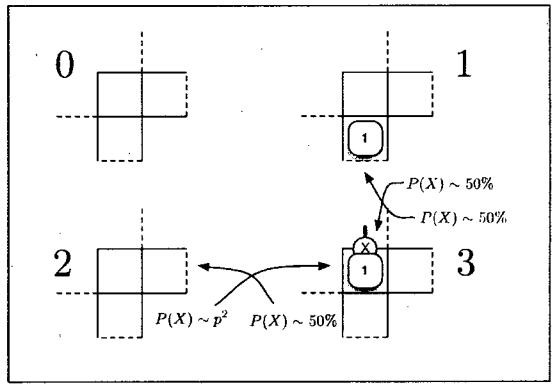
notre algorithme que nous expliciterons dans une section ultérieure.

L'amélioration due à l'augmentation du nombre de passes de BP sature très rapidement. Après trois passes de BP, on assiste à une saturation marquée du seuil. Ce phénomène peut être expliqué par les cycles effectués par l'information contenue dans les messages échangés. En effet, l'information peut revenir vers son expéditeur initial par le biais d'un cycle dans les échanges, tel qu'illustré à la figure 5.19. On a vu au chapitre 4 que l'algorithme est exact sur les arbres, mais que si notre ensemble de régions contenait des cycles, rien n'était garanti. Toutefois, si le cycle le plus court a une certaine longueur, tant que le nombre de passes de BP est inférieur à cette longueur, l'effet des boucles n'est pas apparent. Ceci pourrait expliquer le fait que la saturation se produise à trois passes de BP pour le code torique, car les cycles les plus courts sont de longueur quatre.

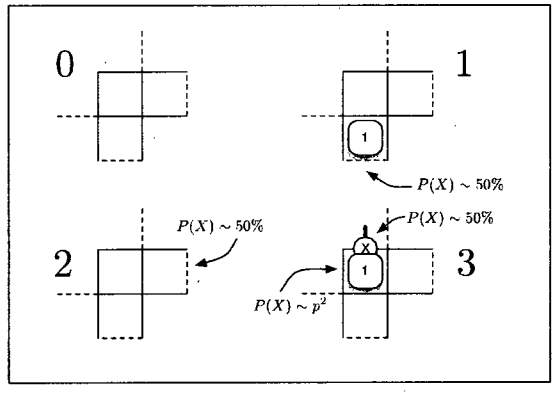
Nous verrons dans les prochaines sections comment simplifier l'algorithme pour le rendre beaucoup plus rapide sans trop en affecter le seuil, ce qui nous permettra d'étudier des tailles plus convaincantes.



(a) Calcul des probabilités marginales

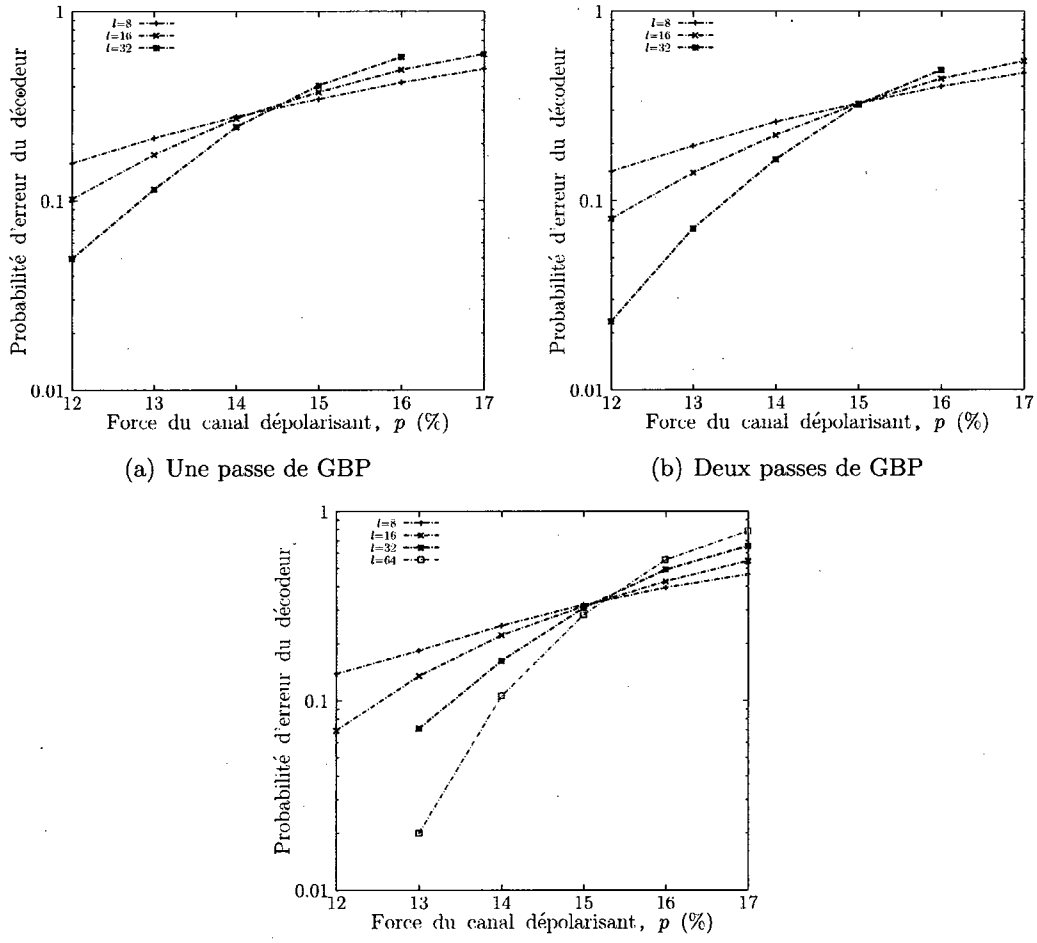


(b) Échange des messages



(c) Mise à jour des P_J

FIGURE 5.17 – Intuition derrière la BP : (a) Distribution marginale calculée à l'aide du modèle de bruit, (b) Échange des messages, i.e. des probabilités marginales sur les qubits partagés et (c) Mise à jour des P_J à l'aide des messages reçus.



(a) Une passe de GBP

(b) Deux passes de GBP

(c) Trois passes de GBP

FIGURE 5.18 – Effet de la BP sur le décodeur de renormalisation pour une, deux et trois passes de BP. Chaque point représente 10^4 essais aléatoires.

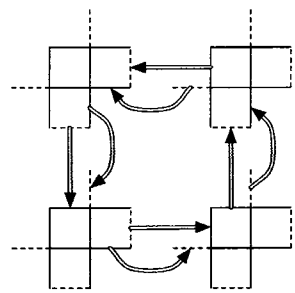


FIGURE 5.19 – Exemples de cycles de longueur quatre dans la BP.

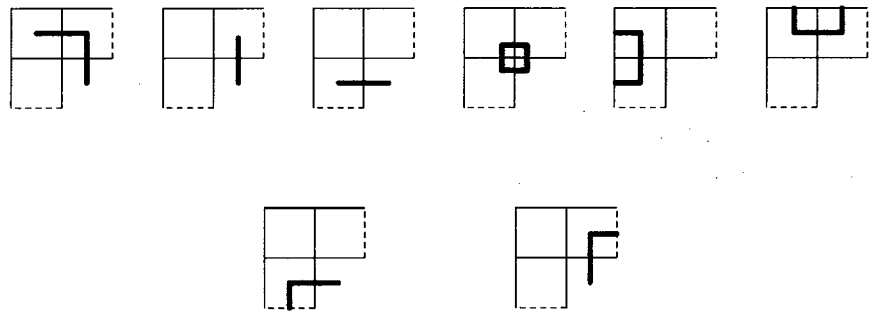


FIGURE 5.20 – Le sous-code contient maintenant 10 qubits plutôt que 12, car seules les plaquettes doivent être complètes. De plus, tous les stabilisateurs et erreurs pures Z ne sont plus nécessaires pour compléter la base X sur les qubits.

5.3 Canal d'inversion de bit

Avant de parler du dernier module qui complétera l'algorithme, arrêtons-nous pour étudier ce que nous avons construit jusqu'à maintenant dans un contexte légèrement différent. Jusqu'à présent, nous avons décodé les erreurs X et Z à la fois ce qui nous a permis de tenir compte des corrélations introduites par Y. Toutefois, nous verrons dans la section sur la complexité que cela introduit des facteurs constants importants et limitants dans la complexité de l'algorithme. Pour obtenir un algorithme que l'on pourra comparer plus directement avec PMA, nous le simplifions maintenant pour ne traiter qu'un type d'erreur. Nous choisissons les erreurs X, mais le code de Kitaev étant symétrique, cela ne fait aucune différence de choisir X ou Z.

Dans ce contexte, le modèle de bruit analogue au canal dépolarisant est le canal d'inversion de bit symétrique introduit au chapitre 2. Ce modèle de bruit génère donc des particules X, i.e. des défauts de plaquettes, ainsi que des chaînes d'erreurs X. Notre sous-code s'en trouve simplifié. D'une part, nous ne nous intéressons qu'aux syndromes plaquettes. Donc, les qubits empruntés pour compléter les croix peuvent être écartés. D'autre part, les opérateurs logiques Z peuvent aussi être écartés. Les stabilisateurs et erreurs pures du sous-code résultant sont présentés au tableau 5.3 et illustrés à la figure 5.20 et ses opérateurs logiques, au tableau 5.4 et à la figure 5.21.

	1	2	3	4	5	6	7	8
s t	X_3X_5	X_5	X_7	$X_3X_4X_5X_7$	$X_2X_4X_6$	$X_0X_1X_3$	X_7X_8	X_5X_9

TABLEAU 5.3 – stabilisateurs et erreurs pures du sous-code simplifié

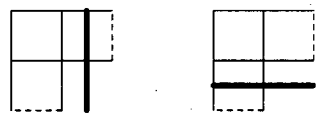


FIGURE 5.21 – Opérateurs logiques X du sous-code simplifié.

5.3.1 Simulations numériques

L'algorithme étant simplifié, il est maintenant possible de l'appliquer à de plus grandes tailles. La figure 5.22 montre des résultats analogues à ceux obtenus à la figure 5.18. Pour une passe de BP, on voit un seuil d'environ 8%, pour deux passes, un peu moins de 9% et pour trois passes, 9%. Pour explorer de manière plus convaincante la saturation, nous avons testé l'algorithme pour neuf passes de BP et le seuil ne s'est pas déplacé de manière perceptible, toujours à 9%. On confirme bien l'idée que plus de trois passes de BP est inutile dans le cas du décodeur à renormalisation.

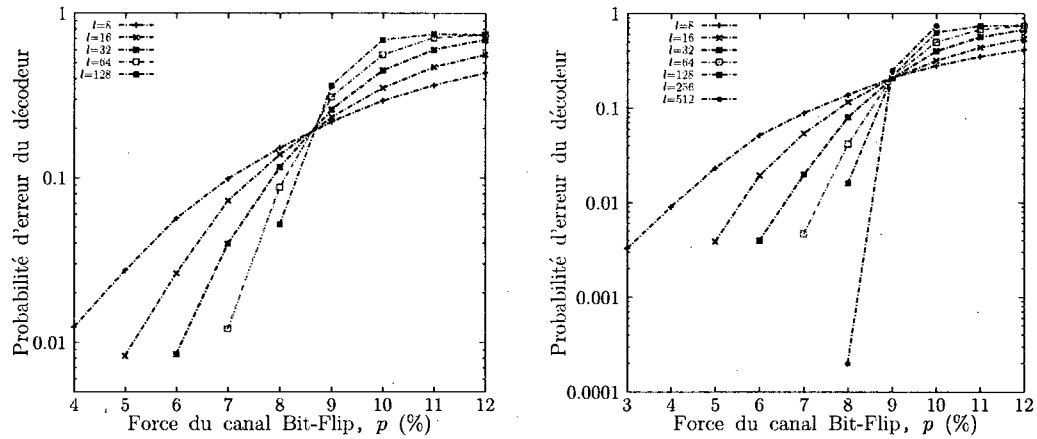
Comme il s'agit de bruit d'inversion de bit, les performances du décodeur peuvent être plus directement comparées à celles du PMA qui décode les deux types d'erreurs X et Z indépendamment. Rappelons que PMA a un seuil d'environ 10.3%. Encore une fois, étant donné la rapidité acquise par le décodeur grâce à la simplification, le seuil de 9% est très satisfaisant.

5.4 Sous-code 2×1

Dans cette section, nous verrons qu'il existe une autre manière de simplifier le décodeur qui permet toujours de décoder les erreurs X et Z à la fois, permettant de tenir compte

	1	2
$\overline{X}$	X_1X_5	X_6X_7

TABLEAU 5.4 – Opérateurs logiques du sous-code simplifié



(a) Une passe de GBP

(b) Deux passes de GBP

(c) Trois passes de GBP

(d) Neuf passes de GBP

FIGURE 5.22 – Effet de la BP sur le décodeur de renormalisation soumis à un bruit classique d'inversion de bit pour une, deux, trois et neuf passes de BP. Chaque point représente 10^4 essais aléatoires.

FIGURE 5.23 – Le sous-code contient maintenant $n = 6$ qubits plutôt que 12. Pour que le sous-code encode deux qubits, $k = 2$, on doit spécifier $m = n - k = 4$ générateurs du stabilisateur.

	1	2	3	4
s	$Z_0Z_2Z_3Z_5$	$X_0X_1X_3X_4$	Z_5	X_4
t	X_3	Z_1	X_3X_5	Z_0Z_4

TABLEAU 5.5 – stabilisateurs et erreurs pures du sous-code 2×1

des corrélations. Plus haut, nous avons introduit un sous-code de taille 2×2 permettant de « découper » le code torique. C'est ce paramètre que nous changeons ici. Nous utiliserons plutôt un sous-code 2×1. On verra que cela donne lieu à une petite subtilité dans le décodage, mais que globalement, la procédure est la même.

Spécifions d'abord ce sous-code. Comme il est deux fois plus petit que l'original, il contiendra six qubits plutôt que douze. Ses stabilliseurs et erreurs pures sont présentés au tableau 5.5 et illustrés à la figure 5.23 et ses opérateurs logiques, au tableau 5.6 et à la figure 5.24. Notons qu'il ne nécessite que deux qubits partagés.

Un sous-code 2×1 ne renormalise que dans une direction à la fois. Pour arriver à renormaliser les deux directions il faut décoder deux fois plutôt qu'une en utilisant le sous-code et son transposé de manière alternée comme le montre les figures 5.25 et 5.26. Deux fois plus d'étapes de renormalisation sont donc nécessaires. Arrivé à un réseau 2×2, on décode directement ce code torique, comme précédemment.

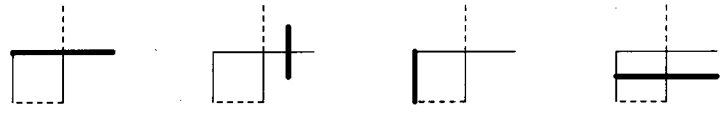


FIGURE 5.24 – Opérateurs logiques du sous-code 2×1.

	1	2
$\overline{Z}$	$Z_0 Z_1$	Z_2
$\overline{X}$	X_1	$X_2 X_3$

TABLEAU 5.6 – Opérateurs logiques du sous-code 2×1.

FIGURE 5.25 – À gauche : Le sous-code 2×1 ne permet de renormaliser que dans une direction à la fois. À droite : utilisation de la transposée du sous-code pour permettre une renormalisation dans l'autre direction.

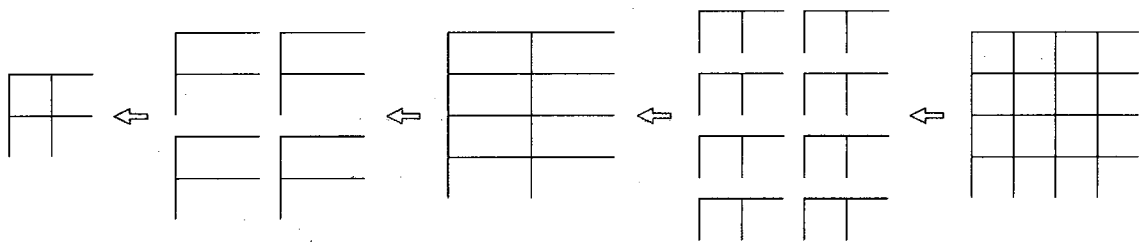


FIGURE 5.26 – Pour renormaliser dans les deux directions, il faut effectuer deux renormalisations, une avec le sous-code et l'autre avec son transposé.

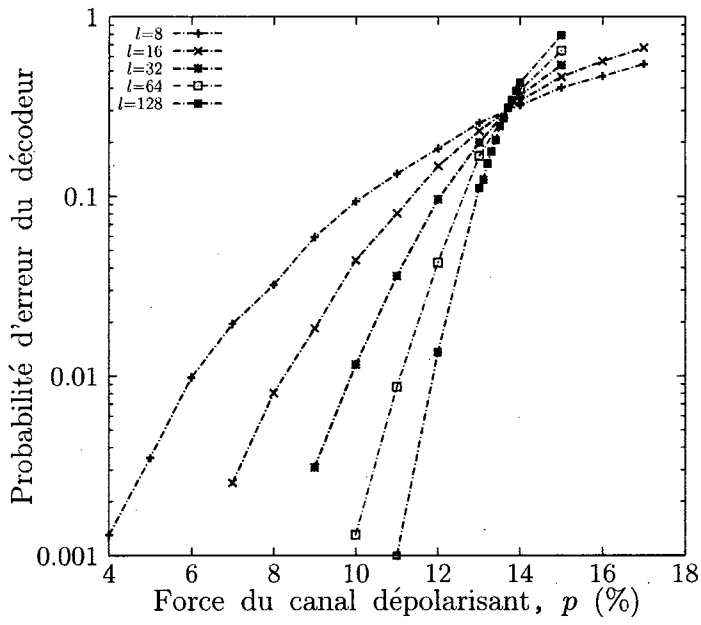


FIGURE 5.27 – Probabilité d'erreur du décodeur de renormalisation avec le sous-code 2×1 avec trois passes de BP en fonction de la force du canal dépolarisant pour les tailles de réseau $\ell = 8$, 16, 32, 64 et 128. Chaque point a été déterminé par un banc d'essai de 10^4 erreurs aléatoires. Le seuil vaut environ 13.5%.

5.4.1 Simulations numériques

Les performances de ce décodeur sont présentées à la figure 5.27. Le seuil vaut environ 13.5% ce qui est comparable au seuil du décodeur utilisant le sous-code 2×2 malgré l'importante simplification apportée. Cela prouve la grande flexibilité du décodeur. On peut ajuster les performances du décodeur en fonction de la puissance de calcul et du temps mis à notre disposition.

On peut tenter une ultime simplification en combinant les deux types de simplifications introduites ci-haut. On obtient un sous-code 2×1 ne traitant que les erreurs X. Les résultats sont présentés aux figures 5.28 et 5.29. Dans ce cas, on obtient un seuil d'environ 8.2% à comparer avec 9% pour le décodeur utilisant le sous-code 2×2 et 10.3% pour le PMA. Encore une fois, on constate la grande flexibilité du décodeur qui permet des compromis raisonnables entre le seuil et les ressources consommées. Avec ce décodeur, on a pu décoder des réseaux de taille 1024×1024, i.e. contenant plus d'un million de spins, et ce, sans même paralléliser. Comme on l'a dit plus haut, le PMA est limité en

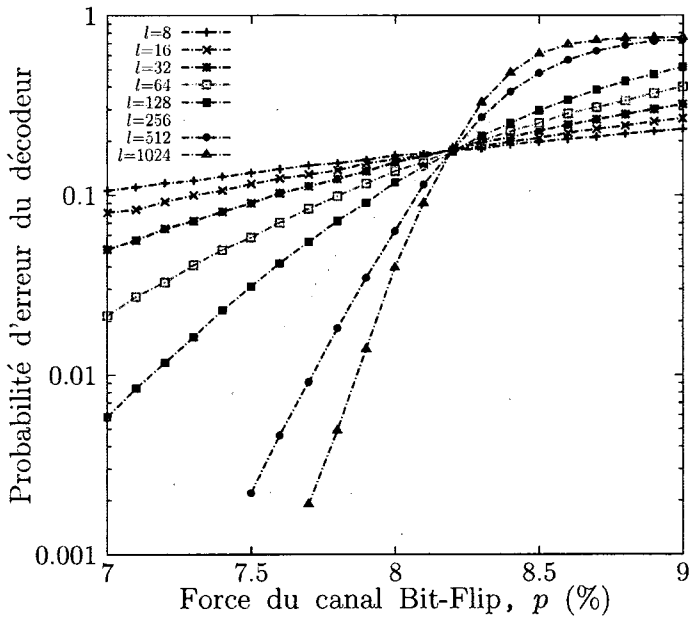


FIGURE 5.28 – Probabilité d'erreur du décodeur de renormalisation avec le sous-code 2×1 avec trois passes de BP en fonction de la force du canal d'inversion pour les tailles de réseau $\ell = 8$ à $\ell = 1024$. Chaque point a été déterminé par un banc d'essai de 10^4 erreurs aléatoires. Le seuil vaut environ 8.2%.

pratique à une taille de moins de 100×100. Avec la possibilité de paralléliser en plus, on conclut que notre décodeur est extrêmement plus efficace en pratique.

5.5 Décodeur physique préliminaire

Dans le chapitre sur le décodage, sous-section 2.3.2, on a vu en quoi les corrélations entre les défauts X et Z étaient importantes lorsqu'on considère le canal dépolarisant. Rappelons que le canal dépolarisant ne fait rien avec probabilité $1 - p$ et applique X,Y ou Z avec probabilité $p/3$. Puisque $Y = iXZ$, on peut affirmer que le canal dépolarisant est une combinaison corrélée d'un canal d'inversion de bit et d'un canal d'inversion de phase. Illustrons par un exemple en quoi ces corrélations font une différence (cf. Fig. 5.30). Un décodeur ne tenant pas compte des corrélations, comme PMA, décode les deux types d'erreurs indépendamment. Il considérera la correction du haut de la figure 5.30 comme étant celle de plus petit poids, cinq, c'est-à-dire une correction de type X de

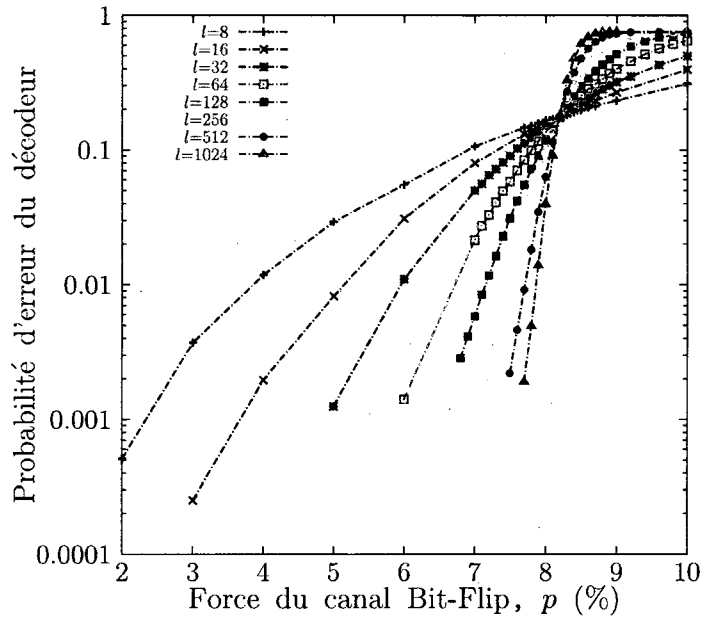


FIGURE 5.29 – Vision étendue du graphique de la figure 5.28. Tous les points de cette dernière sont contenus dans cette figure. Chaque point a été déterminé par un banc d'essai de 10^4 erreurs aléatoires, sauf les deux plus bas qui ont nécessité 10^5 essais.

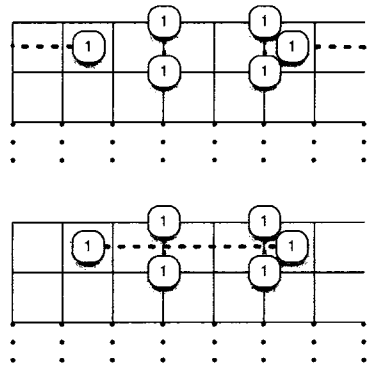


FIGURE 5.30 – En haut : correction proposée par un décodeur de type PMA. En bas : correction optimale obtenue en tenant compte des corrélations. Les traits verts représentent des opérateurs Z et les traits bleus, des opérateurs X.

poids minimal trois et une correction de type Z de poids minimal deux. Or, la correction optimale est celle présentée au bas de la figure 5.30, de poids minimal total quatre. Celle-ci ne peut être considérée comme telle que si l'on tient compte des corrélations, c'est-à-dire que si une ligne X et une autre Z se croisent, alors il faut compter une erreur Y et non pas une erreur X *et* une erreur Z.

On voit que l'effet des erreurs Y est d'introduire un croisement entre une ligne X et une ligne Z. Les corrélations introduites sont donc locales. Or, un décodeur physique comme celui introduit aux chapitres 2 et 4 traite justement les erreurs de manière locale. Il ne peut pas caractériser des opérateurs étendus comme nos opérateurs logiques, mais il pourrait traiter les corrélations qui, elles, sont locales.

Si le bruit était bel et bien le produit d'un canal X et d'un canal Z, l'apparition d'une erreur X ne nous apprendrait rien au sujet de l'apparition d'une éventuelle erreur Z. Or, dans le canal dépolarisant, la présence d'une erreur X nous indique qu'il y a une forte chance qu'il y ait aussi une erreur Z et vice-versa, par le biais d'une erreur Y. Comme les trois erreurs de Pauli sont équiprobables, si on soupçonne la présence d'une erreur X, en réalité, il s'agit soit effectivement d'une erreur X, soit d'une erreur Y avec la même probabilité. Une erreur Y correspondant à une erreur de chaque type, on a alors les probabilités conditionnelles suivantes

$$P(Z|X) = P(X|Z) = \frac{1}{2}. \tag{5.5}$$

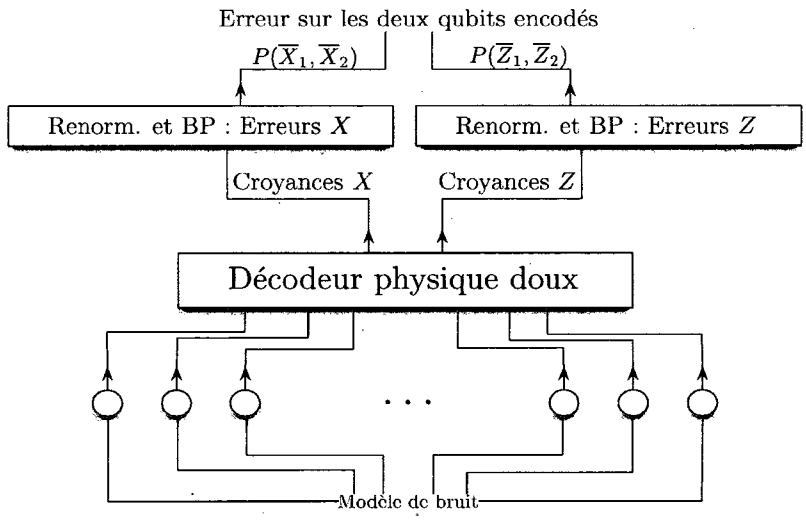


FIGURE 5.31 – Schéma du décodeur complet, du modèle de bruit à la probabilité des erreurs logiques en passant par le décodeur physique et les deux décodeurs de renormalisation simplifiés.

Nous tenterons de tenir compte de ces corrélations entre les deux types d'erreurs avant de les décoder de façon indépendante. Pour ce faire, on soumet le syndrome à un décodeur physique préliminaire qui effectue un décodage local, qubit par qubit, dont la fonction est de tenir compte des corrélations. Le résultat de ce décodage préliminaire est un nouveau modèle de bruit local que l'on peut ensuite soumettre en entrée au décodeur de renormalisation que nous avons traité jusqu'à maintenant.

Plus précisément, on a soumis le syndrome et le modèle de bruit à huit passes de décodage physique tel que présenté au chapitre 4. Une fois ces huit passes complétées on a calculé les croyances (probabilités marginales) X et Z pour chacun des qubits. On a ensuite envoyé les croyances X à un décodeur de renormalisation de sous-code 2×2 traitant les erreurs X seulement opérant trois passes de BP et on a envoyé les croyances Z à un autre décodeur identique, mais traitant les erreurs Z plutôt. Un schéma est donné à la figure 5.31.

5.5.1 Simulations numériques

On a caractérisé ce décodeur aussi par Monte Carlo. Les résultats sont présentés à la figure 5.32. On voit apparaître un seuil de 16.5% dépassant celui de PMA de 15.5%

et dépassant donc tous les seuils présentés précédemment dans ce mémoire. Donc, le décodeur physique a su faire un travail crucial et c'est au niveau des corrélations que nous croyons que cela s'est joué.

Il est intéressant de noter que cette valeur sature la borne de Hamming [29]. En théorie des codes classiques, la borne de Hamming stipule que la force du bruit p, doit respecter l'inégalité suivante,

$$0 \leq 1 - 2H(p), \tag{5.6}$$

où $H(p) = p\log_2 p + (1-p)\log_2(1-p)$ est l'entropie binaire (on omettra la base 2 à l'avenir) .

La borne s'appuie sur des arguments de comptage pour affirmer que dans la limite des longs messages, si la force du bruit respecte la borne, ceux-ci peuvent être transmis parfaitement avec grande probabilité, tandis que si l'inégalité n'est pas respectée, l'information sera typiquement complètement perdue. Bref, en dessous de cette borne il est impossible de transmettre de l'information.

Donnons une idée de la preuve dont l'intuition est géométrique. Le nombre de mots de n bits est 2^n. Si k bits logiques sont encodés dans ces n bits physiques, on a 2^k mots logiques possibles. Supposons que le système soit soumis à du bruit d'inversion de bit de force p. Pour des messages très long, on s'attend à ce que typiquement np des n bits aient été inversés. Pour être apte à corriger, il faut que tous les 2^k mots codes soient distants d'au moins $2np$. On peut donc imaginer que chacun des 2^k mots occupe une « sphère » de rayon np. Ainsi, étant donné un mot code et une erreur de poids au plus np, on peut toujours trouver une correction unique. Le volume occupé par tous les mots codes doit être au plus 2^n, le volume total de l'espace. Ceci se traduit par l'expression suivante :

$$\binom{n}{np} 2^k \leq 2^n. \tag{5.7}$$

Dans le cas qui nous intéresse, on doit tenir compte des erreurs d'inversion de bit et de phase. Il y a donc $\binom{n}{np}\binom{n}{np} = \binom{n}{np}^2$ erreurs possibles pour chaque mot codes. On a plutôt l'expression

$$\binom{n}{np}^2 2^k \leq 2^n. \tag{5.8}$$

En prenant le logarithme de cette inégalité, on a

$$2\log\binom{n}{np} + k \leq n. \tag{5.9}$$

En utilisant la formule de Stirling ($\log n! \approx n\log n - n$), on trouve pour n grand :

$$\begin{aligned}
\log\binom{n}{np} &\approx n\log n - n - np\log np + np - n(1-p)\log n(1-p) + n(1-p) && (5.10)\\
&= n\log n - np\log np - n(1-p)\log n(1-p) && (5.11)\\
&= n(-p\log p - (1-p)\log(1-p)) && (5.12)\\
&\equiv nH(p), && (5.13)
\end{aligned}$$

où $H(p)$ est l'entropie binaire. En substituant le résultat de cette approximation dans l'équation 5.9, on a

$$\begin{aligned}
2nH(p) &\leq n - k. && (5.14)\\
2H(p) &\leq 1 - R && (5.15)\\
R &\leq 1 - 2H(p), && (5.16)
\end{aligned}$$

où $R = k/n$ est le rendement du code. En notant que k et n sont toujours positifs, on retrouve bien la borne de Hamming :

$$0 \leq 1 - 2H(p), \tag{5.17}$$

La borne donne une valeur limite pour p d'environ $\sim 11\%$ pour un canal bit-flip et $\sim 16.5\%$ pour un canal dépolarisant. Or, tout décodeur n'exploitant pas les corrélations et décodant les deux types d'erreurs indépendamment ne pourra dépasser cette limite. Notre décodeur l'atteint et il n'y a pas de raisons fondamentales de croire qu'il ne pourrait pas la dépasser à l'aide d'une future amélioration de la technique, contrairement au PMA qui est fondamentalement limité à 16.5%. Tenir compte des corrélations est donc un élément clé dans l'établissement d'un haut seuil, probablement plus important encore que de tenir compte de la dégénérescence.

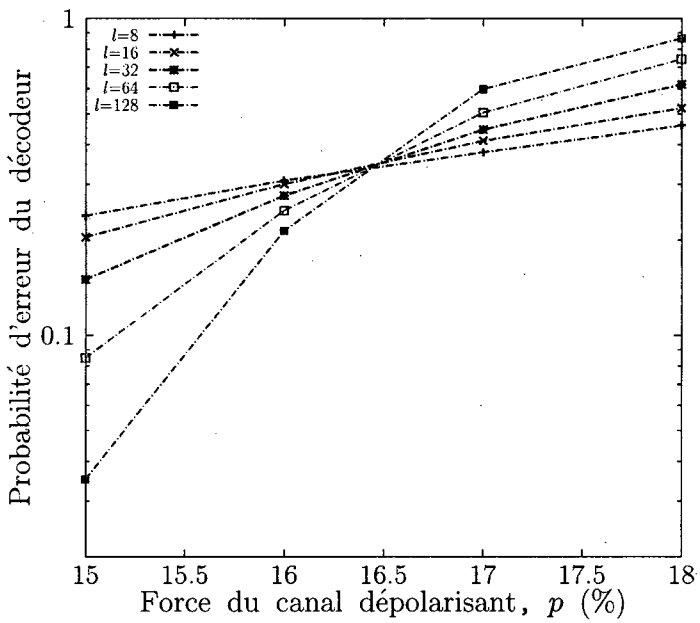


FIGURE 5.32 – Probabilité d'erreur du décodeur combiné physique-renormalisation avec le sous-code 2×2 et trois passes de BP en fonction de la force du canal dépolarisant pour les tailles de réseau $\ell = 8$ à $\ell = 128$. Chaque point a été déterminé par un banc d'essai de 10^4 erreurs aléatoires. Le seuil vaut environ 16.5% ce qui dépasse le seuil du PMA.

5.6 Complexité

Dans les sections précédentes, nous avons introduit une approche alternative au PMA, ainsi que plusieurs variantes, dont les performances sont comparables ou même supérieures en terme de tolérance au bruit. Attardons-nous maintenant à comparer leur complexité respective. D'abord, rappelons que PMA nécessite un temps de décodage qui croît comme $\mathcal{O}(\ell^6)$. Notons qu'il n'est pas parallélisable.

5.6.1 Renormalisation

Malgré qu'il ne soit pas clair que la renormalisation seule est un décodeur, il vaut la peine d'en étudier la complexité puisque cette méthode est à la base des autres introduites par la suite.

Décoder un sous-code de manière logique s'accomplit en temps constant. En effet, tous les sous-codes contiennent 12 qubits. Malgré qu'il s'agisse d'une constante, il faut reconnaître qu'elle est importante. Étant donné un syndrome, on a vu qu'on a 2^4 choix possibles pour l'erreur pure t_e sur les qubits empruntés. On sait qu'on a un choix de 2^4 classes logiques. De plus, on a 10 générateurs du stabilisateur, ce qui représente 2^{10} configurations. Au total, on a donc un ensemble de 2^{18} probabilités à calculer. Il s'agit d'un facteur limitant, mais comme on le sait maintenant, la méthode introduite est très flexible et on peut grandement varier cette constante tout en gardant le seuil à un niveau raisonnable.

Étant donné le niveau k du décodage, on a $\ell^2/2^k$ sous-codes à décoder. Toutefois, les sous-codes peuvent tous être décodés simultanément. Cette partie de l'algorithme est donc parallélisable. On peut décoder un niveau en espace $\mathcal{O}(\ell^2)$ et en temps contant.

Comme on l'a vu plus haut, à chaque niveau, la taille du réseau est divisé par deux. On en déduit qu'on a $\mathcal{O}(\log \ell)$ niveaux de décodage à accomplir avant d'atteindre une taille 2×2.

Finalement, le décodeur logique du code de Kitaev 2×2 opère aussi en temps constant. Il contient huit qubits et le syndrome est complètement spécifié, il y a donc 2^{10} configurations possibles.

En combinant tous ces résultats on déduit que, sans parallélisation, le temps nécessaire au décodage croît comme $\mathcal{O}(\ell^2 \log \ell)$, alors que si on permet la parallélisation, le temps croît comme $\mathcal{O}(\log \ell)$ et le nombre de noeuds de calcul comme $\mathcal{O}(\ell^2)$.

Bref, l'algorithme basé sur la renormalisation du réseau consomme une quantité de

ressources qui croît plus avantageusement que PMA. De plus, grâce à la parallélisation, un gain exponentiel en temps est rendu possible.

5.6.2 Renormalisation et BP

Considérons le cas où nous avons incorporé la BP à notre algorithme de renormalisation. Voyons comment cela en affecte la complexité.

Dans une passe de BP, chaque sous-code calcule, envoie et reçoit un nombre constant de messages, i.e. huit. Nous choisissons d'effectuer un nombre constant de telles passes de BP entre les étapes de renormalisation. Donc, la quantité de messages à calculer croîtra avec le nombre de sous-codes présents. Or, on sait que ce nombre vaut $\ell^2/2^k$ pour l'étape de renormalisation k. De plus, comme chaque message est calculé en n'utilisant que l'information contenue à l'intérieur du sous-code, la parallélisation est toujours possible. Donc, la BP est de complexité $\mathcal{O}(\ell^2)$ et peut potentiellement être exécutée en temps constant si on permet la parallélisation.

Si on compare à ce qui a été dit plus haut au sujet de la renormalisation, on conclut que la BP ne modifie pas la façon dont les ressources nécessaires au décodage croissent avec ℓ, i.e. le temps croît comme $\mathcal{O}(\log \ell)$ et le nombre de noeuds de calcul comme $\mathcal{O}(\ell^2)$.

5.6.3 Canal d'inversion de bit

La simplification permise en ne considérant que les erreurs X ne change pas comment les ressources consommées croissent avec la taille du réseau. Toutefois, les constantes, qui étaient un facteur limitant, seront modifiées. On peut quantifier cette différence.

Il n'y a que le décodage logique doux d'un sous-code qui est affecté. En effet, le nombre d'étapes de renormalisation et le nombre de sous-codes par étape pour un ℓ donné restent inchangés. Recalculons alors la taille du domaine des configurations pour un sous-code. On a toujours deux qubits logiques, mais puisque l'on ne traite que les opérateurs X, on a 2^2 classes logiques à considérer plutôt que 2^4. On a deux qubits partagés au lieu de quatre et donc 2^2 erreurs pures non-fixées. Aussi, on a trois générateurs du stabilisateurs plutôt que 10 et donc 2^3 stabilisateurs. Au total, on a, pour un syndrome donné, 2^7 configurations à considérer. Rappelons que le décodeur pour le canal dépolarisant devait traiter 2^{18} possibilités. L'algorithme pour le canal bit-flip est donc $2^{11} = 2048$ fois plus rapide que l'algorithme initial. Toutefois, notons qu'on néglige l'impact de la variation dans le nombre de messages échangés et de ses conséquences sur le calcul de la probabilité

de chaque configuration. Ici, on échange quatre messages plutôt que huit.

5.6.4 Sous-Code 2×1

Encore une fois, la simplification apportée ne change pas comment les ressources consommées croissent avec ℓ, elle ne modifiera que les constantes.

Commençons par analyser l'approche du sous-code 2×1 traitant X et Z à la fois. Tout d'abord rappelons que l'on doit effectuer deux fois plus de niveaux de décodages pour renormaliser dans les deux directions. On a deux qubits logiques et donc 2^4 classes logiques. On a deux qubits partagés et donc 2^2 erreurs pures non spécifiées. On a quatre générateurs du stabilisateur et donc 2^4 stabilisateurs. Au total, on doit traiter 2^{10} configurations lors d'un décodage en comparaison de 2^{18} pour le décodeur initial. On a donc un gain d'un facteur $2^7 = 128$.

En combinant les deux simplifications, on se retrouve avec 2^2 classes logiques, un qubit partagé (2^1) et un seul générateur du stabilisateur (2^1). Il faut donc considérer 2^4 configurations et le gain est un facteur de $2^{13} = 8192$ par rapport au décodeur initial et de $2^2 = 4$ par rapport au décodeur X seulement introduit à la section précédente.

Conclusion

L'encodage topologique de l'information quantique a attiré beaucoup d'attention, car c'est un modèle qui semble propice à résister aux erreurs locales.

Tout d'abord, le modèle du calcul topologique est basé sur la statistique anyonique non-abélienne universelle et sur le contrôle de ces anyons. On croit que certains systèmes, tels les gaz 2D d'électrons, pourraient présenter de tels anyons non-abéliens par le biais de l'effet Hall quantique fractionnaire pour certains facteurs de remplissage (e.g. $\nu = 5/2$ [31]). Or, les trajectoires des anyons, encodant le calcul, sont des invariants topologiques et sont donc immunisées aux erreurs locales. À $T = 0$, les processus pouvant corrompre l'information sont alors exponentiellement supprimés par la taille du système. Toutefois, à température finie, ces erreurs locales peuvent se propager librement et toute protection « naturelle » du système est perdue. On doit alors faire appel à un algorithme classique de correction qu'on appelle décodeur.

Ensuite, un tel décodeur est aussi essentiel dans le cadre des mémoires quantiques topologiques basées sur des réseaux de spins 2D. En effet, dans ce contexte, le réseau est perforé et l'information est alors contenue dans la topologie du réseau relative aux perforations, i.e. les trous sont magnétiquement chargés ou non et pour en changer l'état on doit transporter des vortex de champ de l'extérieur du réseau vers la perforation ou d'une perforation à l'autre.

Dans ces deux contextes, les erreurs se manifestent comme des particules, ou défauts. Celles-ci peuvent se propager librement dans le système. Si leur ligne de vie possède une topologie non-triviale, l'information est corrompue. Le problème du décodage est alors de fusionner toutes les particules pour retourner le système dans son état fondamental en s'assurant que tous les défauts ont une trajectoire topologiquement triviale, protégeant ainsi l'information encodée. Notons que malgré que les systèmes d'anyons ne forment pas un réseau mais un continuum, on peut quand même utiliser les idées développées dans ce mémoire pour les décoder. En effet, on peut toujours discrétiser le système en un réseau

carré. À ce moment, la présence ou non d'un anyon dans une cellule du réseau dépend de la parité du nombre d'anyons physiques présents dans cette partie du système. De plus, on peut discrétiser le probabilité sur le continuum de trajectoires puisque chaque trajectoire croise un nombre bien défini d'arêtes du réseau et peut donc être associée à une unique trajectoire sur le réseau. Bref, puisque l'information est encodée dans les grandes longueurs d'ondes, on s'attend à ce que sa correction soit peu sensible aux détails microscopiques du système.

On a aussi vu qu'on pouvait étudier de tels systèmes plus réalistes à l'aide d'un modèle jouet moins réaliste, mais plus aisé à manipuler : le code de Kitaev. L'algorithme utilisé jusqu'à maintenant pour effectuer le décodage était l'algorithme de Edmond aussi appelé *minimum weight perfect matching algorithm* (PMA) [12]. Cet algorithme est sous-optimal, car, dans un langage de physique statistique, il minimise l'énergie plutôt que l'énergie libre, i.e. il néglige la contribution entropique. De plus, il ne tient pas compte des corrélations dans le bruit introduites par le canal dépolarisant, notre modèle de bruit d'intérêt. Il n'est pas parallélisable et est limité en pratique à des réseaux de spins de taille 100×100. Le seuil qu'il atteint est de 10.3% sur le canal d'inversion de bit et de 15.5% sur le canal dépolarisant [16].

Dans ce travail, nous avons introduit une nouvelle approche basée sur la combinaison de plusieurs méthodes existantes en physique de la matière condensée et en informatique : renormalisation, propagation de croyance et décodeur physique, dans le but d'approximer le décodeur optimal, i.e. le décodeur logique de complexité exponentielle qui minimise l'énergie libre du système de manière exacte.

La renormalisation permet de traiter efficacement des degrés de liberté étendus. Elle permet de sommer sur une fraction des degrés de liberté pour obtenir un réseau toujours plus petit, permettant au bout du compte de distiller seulement l'information encodée en caractérisant la topologie du réseau. Il n'est pas clair que cette méthode à elle seule soit un décodeur, car on peut identifier certaines configurations d'erreurs de tailles finies brouillant l'information.

La propagation de croyance permet l'auto-cohérence dans la renormalisation. En effet, la renormalisation est rendue possible grâce à la duplication de certains qubits et celle-ci les traite de manière indépendante par la suite. Pour améliorer la qualité des approximations faites par la renormalisation, on impose des conditions de type champ moyen sur les qubits partagés qui assurent une auto-cohérence entre les sous-codes lors d'une étape de renormalisation.

Renorm. 2×2, 3 passes de BP	15.5%
Renorm. 2×1, 3 passes de BP	13.5%
Renorm. 2×2, 3 passes de BP, 8 passes de décodeur physique préliminaire	16.5%
Renorm. 2×2, 3 passes de BP, canal d'inversion	9%
Renorm. 2×1, 3 passes de BP, canal d'inversion	8.2%

TABLEAU 5.7 – Seuil des différentes versions du code

Quant à lui, le décodeur physique permet d'alléger les calculs en traitant de façon approximative les corrélations dans les erreurs à l'aide d'un modèle effectif, puis en les ignorant dans le processus de renormalisation. En effet, on peut traiter les corrélations en premier lieu et ce, de manière locale, à l'aide du décodeur physique. Une fois ce traitement préliminaire accompli, on peut considérer les erreurs X et Z comme étant découplées et on les décode alors indépendamment à l'aide de deux décodeurs de renormalisation assistés par de la propagation de croyance.

L'algorithme ainsi obtenu est très flexible et permet plusieurs compromis entre la rapidité d'exécution et la performance du seuil obtenu. On a vu qu'il était de complexité $\mathcal{O}(\ell^2 \log \ell)$ en série et, si on parallélise, $\mathcal{O}(\log \ell)$ en temps et $\mathcal{O}(\ell^2)$ en noeuds de calculs. Ceci est déjà un avantage en comparaison du PMA qui est de complexité $\mathcal{O}(\ell^6)$ et qui n'est pas parallélisable. Insistons sur le gain exponentiel en temps rendu possible. En effet, pour que le décodeur soit d'une quelconque utilité, il doit impérativement décoder les erreurs plus rapidement que celles-ci se produisent. De plus, en pratique, PMA est limité à des réseaux de tailles 100×100, alors que notre décodeur peut traiter des réseaux 1024×1024 sans même utiliser de parallélisme. Le tableau 5.7 présente les différents seuils atteints par les différentes versions de l'algorithme.

Travaux en cours et perspectives

Pendant la rédaction de ce mémoire, de nouvelles avenues ont été explorées. Tout d'abord, les travaux de Thomas Stace et Sean Barret [33] ont montré que le seuil de tolérance au bruit d'effacement pour le code torique était de 50%. Ils ont aussi étudié comment évoluaient les seuils lorsqu'on combinait les canaux à effacement et dépolarisant. On s'est alors intéressé à tester notre algorithme dans le contexte du canal à effacement. Ces travaux sont en cours.

De plus, comme on l'a dit, notre décodeur ne devrait pas être limité au code torique,

il devrait être en mesure de décoder tout code topologique s'inscrivant dans le cadre des codes stabilisateurs. Or, les codes de couleur [3] sont un exemple de tels codes pour lesquels PMA ne fonctionne pas. En fait, on ne connaît pas de décodeurs efficaces pour eux. Nous travaillons donc à adapter ce qui a été présenté ci-haut dans le but de concevoir le premier décodeur efficace pour de tels codes.

Ensuite, notons que nous pourrions probablement apporter plusieurs améliorations au décodeur lui-même. Par exemple, nous pourrions sûrement nous débarrasser des effets de tailles importants pour le décodeur utilisant la renormalisation seule. En effet, on peut imaginer qu'en effectuant des translations du réseau de façon judicieuse entre les étapes de renormalisation, on parviendrait à faire en sorte que les zones sensibles du décodeur (cf. Fig. 5.15) ne correspondent jamais à la même région du réseau. Ainsi, toute configuration de taille finie serait éventuellement décodée ailleurs que sur un de ces coins sensibles. Ensuite, on pourrait s'intéresser à étudier la convergence des messages échangés. En effet, on sait que s'ils convergent, ils convergent vers la bonne valeur, mais s'ils ne convergent pas, alors on ne peut rien dire. Il serait intéressant de voir si les messages convergent ou non au seuil. S'ils convergent, alors le seuil n'est pas dû à la BP, s'ils ne convergent pas, on pourrait peut-être pousser le seuil plus loin en la modifiant judicieusement. Par exemple, les messages échangés le sont par rapport à des qubits. Or, les stabilisateurs corrèlent les quatres qubits les touchant. Il serait alors peut-être plus judicieux d'échanger des messages concernant quatre qubits à la fois.

Finalement, Sergey Bravyi a montré (non publié au moment de la rédaction de ce mémoire) que pour un décodeur similaire à notre méthode de renormalisation, mais basé sur un décodage dur, on pouvait prouver l'existence d'un seuil très faible, mais rigoureux de 10^{-22}. Toutefois, des simulations semblent montrer que le seuil se situe autour de 10^{-5} pour des tailles allant jusqu'à $\ell = 16384$. On tente maintenant d'améliorer ces deux nombres en appliquant son modèle de preuve à nos méthodes.

Bibliographie

[1] D. AHARONOV AND M. BEN-OR, *Fault-tolerant quantum computation with constant error*, in STOC '97 : Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, New York, NY, USA, 1997, ACM, pp. 176–188.

[2] D. AHARONOV, W. VAN DAM, J. KEMPE, Z. LANDAU, S. LLOYD, AND O. REGEV, *Adiabatic quantum computation is equivalent to standard quantum computation*, SIAM Journal of Computing, 37 (2007), p. 166.

[3] H. BOMBIN AND M. A. MARTIN-DELGADO, *Topological quantum distillation*, Physical Review Letters, 97 (2006), p. 180501.

[4] S. BRAVYI, *Universal quantum computation with the nu=5/2 fractional quantum hall state*, Physical Review A, 73 (2006), p. 042313.

[5] S. B. BRAVYI AND A. Y. KITAEV, *Quantum codes on a lattice with boundary*, quant-ph/9811052, (1998).

[6] H. CHEN, *Some good error-correcting codes from algebraic-geometric codes*, IEEE Transactions on Information Theory, 47 (2001), pp. 2059–2061.

[7] E. DENNIS, A. KITAEV, A. LANDAHL, AND J. PRESKILL, *Topological quantum memory*, Journal of Mathematical Physics, 43 (2002), p. 4452.

[8] D. DEUTSCH, *Quantum theory, the church-turing principle and the universal quantum computer*, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 400 (1985), pp. 97–117.

[9] M. H. DEVORET, A. WALLRAFF, AND J. M. MARTINIS, *Superconducting qubits : A short review*, arXiv :cond-mat/0411174v1, (2004).

[10] D. DIEKS, *Communication by epr devices*, Physics Letters A, 92 (1982).

[11] G. DUCLOS-CIANCI AND D. POULIN, *Fast decoders for topological quantum codes*, Physical Review Letters, 104 (2009), p. 050504.

[12] J. EDMONDS, *Paths, trees and flowers.*, Canadian Journal of Mathematics, 17 (1965), pp. 449–467.

[13] A. G. FOWLER, A. M. STEPHENS, AND P. GROSZKOWSKI, *High threshold universal quantum computation on the surface code*, Phys. Rev. A, 80 (2009), p. 052312.

[14] M. H. FREEDMAN, A. KITAEV, M. J. LARSEN, AND Z. WANG, *Topological quantum computation*, Bulletin-American Mathematical Society, 40 (2001), pp. 31–38.

[15] D. GOTTESMAN, *Stabilizer codes and quantum error correction*, Arxiv preprint quant-ph/9705052, (1997).

[16] J. W. HARRINGTON, *Qecc : symplectic lattice and toric codes*, California Insitute of Technology, (2004).

[17] R. JOZSA, *An introduction to measurement based quantum computation*, Quantum Computation and Learning, (2005), p. 62.

[18] A. Y. KITAEV, *Fault-tolerant quantum computation by anyons*, Annals of Physics, 303 (2003), p. 2.

[19] E. KNILL AND R. LAFLAMME, *Concatenated quantum codes*, Arxiv preprint quant-ph/9608012, (1996).

[20] R. LAFLAMME, E. KNILL, D. G. CORY, E. M. FORTUNATO, T. HAVEL, C. MIQUEL, R. MARTINEZ, C. NEGREVERGNE, G. ORTIZ, M. A. PRAVIA, Y. SHARF, S. SINHA, R. SOMMA, AND L. VIOLA, *Introduction to nmr quantum information processing*, Arxiv preprint quant-ph/0207172, (2002).

[21] D. LOSS AND D. P. DIVINCENZO, *Quantum computation with quantum dots*, Physical Review A, 57 (1998), p. 120.

[22] M. A. NIELSEN AND I. CHUANG, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[23] J. K. PACHOS AND P. L. KNIGHT, *Quantum computation with a one-dimensional optical lattice*, Phys. Rev. Lett., 91 (2003), p. 107902.

[24] D. POULIN, *Optimal and efficient decoding of concatenated quantum block codes*, Physical Review A, 74 (2006), p. 052333.

[25] D. POULIN AND Y. CHUNG, *On the iterative decoding of sparse quantum codes*, QIC, 8 (2008), p. 987.

[26] J. PRESKILL, *Lecture notes : Quantum computation*, California Insitute of Technology, (2004).

[27] R. RAUSSENDORF, J. HARRINGTON, AND K. GOYAL, *Topological fault-tolerance in cluster state quantum computation*, New Journal of Physics, 9 (2007), p. 199.

[28] P. W. SHOR, *Scheme for reducing decoherence in quantum computer memory*, Phys. Rev. A, 52 (1995), pp. R2493–R2496.

[29] A. STEANE, *Multiple particle interference and quantum error correction*, Proc.Roy.Soc.Lond.A, 452 (1996), p. 2551.

[30] A. STEANE, *The ion trap information porecessor*, Applied Physics B : Lasers and Optics, 64 (1997), pp. 623–643.

[31] A. STERN, *Anyons and the quantum hall-effect - a pedagocial review*, Annals of Physics, 323 (2008), pp. 204–249.

[32] K. M. SVORE, D. P. DIVINCENZO, AND B. M. TERHAL, *Noise threshold for a fault-tolerant two-dimensional lattice architecture*, Quant.Inf.Comp., 7 (2007), p. 297.

[33] S. D. B. T. M. STACE, *Error correction and degeneracy in surface codes suffering loss*, arXiv :0912.1159v1 [quant-ph], (2009).

[34] W. K. WOOTTERS AND W. H. ZUREK, *A single quantum cannot be cloned*, Nature, 299 (1982), pp. 802–803.

[35] J. S. YEDIDIA, W. T. FREEMAN, AND Y. WEISS, *Constructing free energy approximations and generalized belief propagation algorithms*, IEEE Transactions on Information Theory, 51 (2005), pp. 2282–2312.