**Université de Montréal**

# L'ORDINATEUR QUANTIQUE: COMPLEXITÉ ET STABILISATION DES CALCULS

par

**André Berthiaume**

Département d'informatique et recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

**Université de Montréal**
Faculté des études supérieures

Cette thèse intitulée

**L'ORDINATEUR QUANTIQUE: COMPLEXITÉ ET STABILISATION DES CALCULS**

présentée par

**André Berthiaume**

a été évaluée par un jury composé des personnes suivantes :

*Geňa Hahn*
(président-rapporteur)

*Gilles Brassard*
(directeur de recherche)

*Claude Crépeau*
(membre du jury)

*William K. Wootters*
(examinateur externe)

Thèse acceptée le :

# SOMMAIRE

L'ordinateur quantique [9] est un modèle de calcul physiquement réalisable (selon les régles de la mécanique quantique) utilisant les superpositions d'états quantiques pour se donner accès à un parallélisme exponentiellement grand sur une quantité finie de matière. En contrepartie à cette fantastique capacité de parallélisation, la théorie de la mécanique quantique limite très sévèrement la quantité et le type d'information qu'il est possible d'extraire de ces calculs fait en parallèle par superposition quantique. Malgré ces contraintes, Deutsch donne un problème spécifique dont la solution demande un temps linéaire à toute machine de Turing (déterministe ou probabiliste) et seulement un temps logarithmique pour l'ordinateur quantique. Cet écart exponentiel entre les deux modèles de calcul se limite-t-il à ce seul problème ou bien est-ce que la hiérarchie de complexité quantique est fondamentalement différente de la hiérarchie classique? La présente thèse explore les possibilités qu'offre le modèle quantique.

La classe de complexité **EQP** est la classe des problèmes décidables en temps polynomial exact sur un ordinateur quantique. Le première partie de cette thèse explore la position de **EQP** dans la hiérarchie de complexité relativisée. Nous démontrons entre autre qu'il existe des oracles $X$,$Y$ et $Z$ relativement auxquels on a les inclusions suivantes: $\mathbf{P}^X \subsetneq \mathbf{EQP}^X$, $\mathbf{EQP}^Y \not\subseteq \mathbf{NP}^Y \cup \textbf{co-NP}^Y$, $\mathbf{EQP}^Z \not\subseteq \mathbf{ZPP}^Z$, $\mathbf{EQP}^Z \not\subseteq \mathbf{RP}^Z \cup \textbf{co-RP}^Z$. Ces inclusions relativisées permettent de croire que l'ordinateur quantique est effectivement plus puissant qu'une machine de Turing.

Le modèle quantique étant physiquement réalisable *et* réellement plus puissant qu'une machine de Turing, on a toutes les raisons du monde de le construire.

Malheureusement, les obstacles technologiques sont de taille. L'un de ces problèmes (et non le moindre) est de maintenir un état quantique stable pendant un certain temps. Peu importe les soins que nous mettons à l'isoler, un système quantique subit toujours de légères perturbations causant des erreurs de calcul. Les techniques classiques pour réduire la propagation de ces erreurs (redondance et vote majoritaire, par exemple) ne sont pas applicables dans le cas quantique car elles détruiraient les superpositions. Deutsch propose donc une méthode quantique de stabilisation des calculs. La seconde partie de cette thèse s'interroge sur l'efficacité de cette méthode. La première constatation est que le comportement de la méthode de stabilisation de Deutsch varie énormément en fonction du type de perturbations subi par l'ordinateur quantique. Nous avons donc choisi un modèle d'erreur très simple ayant le double avantage de permettre une analyse complète et détaillée de l'efficacité de la méthode de stabilisation des calculs tout en étant suffisamment complexe pour étudier les diverses possibilités de celle-ci. Nous concluons que sous ce modèle d'erreur, la méthode de stabilisation des calculs proposée par Deustch est très efficace et donne ainsi espoir que cette conclusion tient aussi sous un modèle d'erreur plus réaliste.

# Remerciements

Avant de considérer faire carrière de chercheur, il faut d'abord avoir la passion pour le domaine de recherche. Dans mon cas, cette passion fut éveillée à la suite de deux cours d'informatique théorique enseignés par Gilles Brassard. La qualité de son enseignement et sa propre passion pour l'informatique théorique furent les éléments qui m'ont convaincu à me diriger vers la recherche. Mais c'est vraiment en tant que directeur de recherche que j'ai vraiment pu apprécier Gilles. Ses conseils judicieux, ses critiques éclairées, son aide financière, la chance de m'avoir fait rencontrer de nombreux chercheurs et surtout la confiance en moi qu'il a su m'inspirer dans les moments difficiles sont autant de raisons pour lesquelles je lui suis éternellement reconnaissant.

Je tiens à remercier Geňa Hahn et Claude Crépeau d'avoir accepté d'être membres du jury d'évaluation de cette thèse, et aussi William K. Wootters d'avoir accepté d'agir à titre d'examinateur externe. Claude et William ont également ma reconnaisance pour leurs nombreux conseils et suggestions tout au long de cette recherche.

J'aimerais aussi remercier Artur Ekert et David Deutsch de m'avoir permis de faire un stage dans leur groupe de recherche en cryptographie et calcul quantique de l'université d'Oxford ainsi que Umesh Vazirani pour son invitation à l'Indian Institute of Technology à Delhi. Les connaissances acquises au cours de ce stage furent d'une valeur inestimable pour la rédaction de cette thèse.

Grâce à l'aide financière du Conseil de recherches en sciences naturelle et en génie du Canada, soit sous forme de bourses ou par l'entremise des fonds de mon directeur de recherche, j'ai pu exécuter mes travaux de recherche sans trop de soucis financiers.

Merci aussi à mes amis pour leur présence dans les moments difficiles. Je tiens toutefois à remercier plus spécialement Isabelle Harpin et Sophie Laplante, deux amies inestimables dont le support moral m'a permis de rester à peu près sain d'esprit au cours des quatre dernières années.

Merci également à mon grand ami Marc K. Potters qui, bien malgré lui, m'ouvrit les yeux sur les liens entre la physique et l'informatique en me recommendant le livre "*The Emperor's New Mind*" de Roger Penrose. Il fut également celui qui m'initia aux rudiments de la mécanique quantique.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# CHAPITRE 1

## Introduction

Peut-on simuler l'univers physique sur un ordinateur? C'est avec cette question que Richard Feynman [12] inaugura en 1981 le premier atelier sur la physique du calcul. Selon Feynman, alors qu'un ordinateur classique ne peut probablement pas exécuter une telle simulation *en temps réel*, il est possible qu'un ordinateur quantique en soit capable. Mais ceci équivaut à dire qu'un ordinateur quantique peut accomplir certaines tâches significativement plus rapidement qu'un ordinateur classique. Trois ans plus tard, David Deustch formalise la notion d'un ordinateur quantique et pose la question suivante: "Du point de vue de la complexité du calcul, est-ce que l'ordinateur quantique est plus puissant qu'une machine de Turing?".

### 1. Thème de la thèse

Dans cette thèse, nous donnons des arguments qui portent à croire que l'ordinateur quantique est plus puissant que la machine de Turing. Pour ce faire, nous étudierons les classes de complexités relativisées. Mais montrer l'existence d'un modèle de calcul plus puissant qu'une machine de Turing n'est pas en soi un très grand exploit. De tels modèles, nous en connaissons plusieurs (modèle non-déterministe, modèle à oracle, etc.). Pour que le modèle soit vraiment intéressant, il doit être *physiquement réalisable*. Deustch a argumenté la faisabilité théorique de son modèle, mais il souligne en même temps certains problèmes à résoudre. L'un de ceux-ci est qu'il est impossible de garder un système quantique parfaitement stable. Le second volet de cette thèse considère une solution possible à ce problème.

## 2. Plan de la Thèse

Le plan de la thèse se présente comme suit. Le chapitre 2 donne la description formelle de l'ordinateur quantique et passe en revue les travaux de Deustch et Jozsa sur la complexité quantique.

Le chapitre 3 présente les résultats de complexité relativisée. Nous définissons quelques classes de complexités (classique et quantique) pour terminer par une série de quatre théorèmes montrant certaines inégalités entre ces classes. Le chapitre se termine par une discussion sur la portée de ces théorèmes relativement à la question sur la puissance relative des modèles de Turing et quantique.

Le chapitre 4 aborde la question de la stabilisation de l'ordinateur quantique. Après une brève discussion sur le sujet, nous considérons trois situations de stabilisation et analysons leur efficacité en fonction du temps investi à stabiliser et la quantité d'erreurs résiduelles. Les situations proposées étant applicables seulement pour une variante du modèle quantique originale, nous concluons le chapitre avec une discussion portant sur la stabilisation du modèle général.

La conclusion rappelle l'intérêt de la démarche, souligne ses limites et propose quelques avenues de recherche futures basées sur les principaux résultats de cette thèse.

# CHAPITRE 2

## L'ordinateur quantique

Dans ce chapitre, nous définissons la notion d'ordinateur quantique telle qu'introduite par Deutsch et donnons une revue des résultats de complexité relatifs à ce modèle. Un ordinateur quantique et une machine de Turing sont très similaires tant dans leurs propriétés que dans leur fonctionnement. Tous deux possèdent un ruban semi-infini de cases contenant un caractère provenant d'un alphabet fini et une tête de lecture/écriture elle-même possédant un ensemble fini d'états. Le tout se contrôle par une fonction de transitions qui, en fonction de l'état de la tête et du caractère sous la tête, change l'état de la tête, écrit un caractère sur le ruban et déplace la tête à gauche ou à droite. La différence principale provient du type d'information contenue par chacune des cases du ruban. Supposons que l'alphabet de ruban soit l'ensemble $\{\mathbf{0}, \mathbf{1}\}$. Alors que les cases d'une machine de Turing peuvent contenir uniquement soit un $\mathbf{0}$, soit un $\mathbf{1}$, les cases d'un ordinateur quantique peuvent contenir une "mélange" de $\mathbf{0}$ et de $\mathbf{1}$ appelée *superposition quantique*. L'avantage obtenu par ces superpositions quantiques est mis en évidence par l'exemple suivant. Soit une fonction $f : \{\mathbf{0}, \mathbf{1}\} \to \{\mathbf{0}, \mathbf{1}\}$ et supposons les fonctions de transitions définies de manière appropriée pour le calcul de $f$ pour chacun des deux modèles. Le problème est de calculer les deux valeurs $f(\mathbf{0})$ et $f(\mathbf{1})$. Si le tripatouillage des fonctions de transition est interdit, la seule manière de résoudre ce problème à l'aide d'une machine de Turing est de d'abord calculer $f(\mathbf{0})$, de noter ce premier résultat sur un bout de papier, de réinitialiser la machine de Turing et de calculer $f(\mathbf{1})$. Le prix à payer est donc deux fois le temps de calcul de la fonction $f$ plus le temps nécessaire pour revenir á la configuration initiale. Par contre, si un ordinateur quantique est disponible, une économie substantielle peut être accomplie en utilisant les superpositions quantiques. Il suffit de préparer, sur la première case du ruban, une superposition de $\mathbf{0}$ et de $\mathbf{1}$, puis de laisser exécuter la fonction de transition sur cette donnée. Cette superposition a pour effet de donner à l'ordinateur une double "personnalité": l'une croit qu'elle voit un $\mathbf{0}$ sur son ruban, l'autre croit y voir un $\mathbf{1}$. Le fonctionnement de l'ordinateur quantique, décrit plus loin, est tel que chacune des "personnalités" calcule en bonne foi la valeur de $f$ sur son donnée et le résultat de ces calculs sera une superposition de $f(\mathbf{0})$ et $f(\mathbf{1})$. Donc, pour le prix d'un seul calcul de $f$, les deux réponses sont calculées. En fait, dans le cas général, si $f : \{\mathbf{0}, \mathbf{1}\}^n \rightarrow \{\mathbf{0}, \mathbf{1}\}$, les $2^n$ valeurs de $f$ peuvent être calculées en superposition (avec un ordinateur fortement schizophrénique!) en temps $\Theta(n)$ plus le temps d'un calcul de $f$.

Évidemment, une telle puissance de calcul ne vient pas gratuitement. Bien qu'un nombre exponentiel de valeurs puissent être calculées en superposition en temps linéaire, les principes de la mécanique quantique limitent fortement le type et la quantité d'information qui peuvent être extraits d'une superposition. Le but de la présente thèse est de démontrer que, malgré ces limitations, le parallélisme offert par la superposition quantique permet des gains appréciables sur la machine de Turing. La prochaine étape consiste donc à définir plus formellement le modèle ainsi que certain concepts importants de mécanique quantique.

## 1. Description du modèle

**1.1. Concepts généraux.** Tout d'abord, un point de vocabulaire: les termes 'machine de Turing' et 'ordinateur' sont interchangeable. Sans perte de généralité, toutes les machines de Turing qui seront considérées (déterministe, non-déterministe ou quantique) utiliseront l'alphabet $\{\mathbf{0}, \mathbf{1}\}$. Notons également que notre représentation de l'ordinateur quantique est fortement inspirée de [19].

La fonction de transition $\delta_1$ d'une machine de Turing a la forme suivante:

$$\delta_1 : (b, q) \rightarrow (b', q', d)$$

c'est-à-dire qu'en fonction du caractère $b$ sous la tête de lecture et de l'état interne $q$, la machine écrit le caractère $b'$, change son état interne pour $q'$ et se déplace d'une case en direction $d$. Une machine de Turing *probabiliste* est en tout point identique à la version déterministe sauf que la 'fonction' de transition $\delta_2$ aura une forme probabiliste (le terme 'fonction' est pris au sens large puisqu'une paire $(b, q)$ ne définit plus une unique transition mais bien un ensemble de transitions).

$$\delta_2 : (b, q) \rightarrow \begin{cases} (b_1, q_1, d_1) & \text{avec prob. } p_1^{bq} \\ (b_2, q_2, d_2) & \text{avec prob. } p_2^{bq} \\ \quad\vdots & \\ (b_n, q_n, d_n) & \text{avec prob. } p_n^{bq} \end{cases}$$

où $p_1^{bq}, \ldots, p_n^{bq} \in [0,1]$ et $\sum_{i=1}^{n} p_i^{bq} = 1$. Lors d'un calcul, une seule transition $(b_i, q_i, d_i)$ sera choisie avec probabilité[1] $p_i^{bq}$.

Soit $M$ une machine de Turing probabiliste utilisant $\delta_2$ comme fonction de transition. Il est possible d'exprimer le calcul de $M$ sur la donnée $x$ avec la notation des processus de Markov. Soit $S(x)$ une borne supérieure sur le nombre de cases de ruban qui seront utilisées lors du calcul de $M(x)$. Si $S(x)$ est finie, alors une configuration $\gamma_i$ de $M$ a la forme

$$\gamma_i = (q_i, t_i, \rho_i)$$

où $q_i$ est l'état interne de $M$, $t_i$ est la position de la tête et $\rho_i$ est une chaîne finie de caractères représentant le contenu du ruban. Soit $w$ suffisamment grand pour que l'ensemble

$$\Gamma = \{\gamma_1, \ldots, \gamma_w\} \tag{2.1}$$

[1]Remarque techniqque: la précision des valeurs de probabilités $\{p_1^{bq}, \ldots, p_n^{bq}\}$ doit absolument être finie, sans quoi il est possible de faire tricher le modèle de calcul. Voici un exemple d'une telle tricherie: soit une machine de Turing probabiliste $M$ ne possédant pour seule transtion que

$$(0, q_0) \rightarrow \begin{cases} (0, q_1, R) & \text{avec prob. } p_\Omega \\ (0, q_2, R) & \text{avec prob. } 1 - p_\Omega \end{cases}$$

où $p_\Omega$ est l'encodage binaire du problème d'arrêt (le nombre binaire tel que le $i$ième bit est 1 si et seulement si la $i$ième machine de Turing déterministe arrête sur la donnée $i$). Avec une probabilité d'erreur aussi petite que désirée, on peut décider du problème d'arrêt avec l'algorithme suivant: avec un nombre exponentiel (en $i$) d'essais de la machine $M$, on obtient une bonne approximation des premiers $i$ bits de $p_\Omega$ et donc on peut déterminer avec bonne probabilité si la machine $i$ arrête sur la donnée $i$.

contienne toutes les configurations possibles relativement à $M(x)$. (Note: pour la suite, on suppose un ordre standard quelconque sur $\Gamma$.) Il est facile de voir que l'on peut réécrire $\delta_2$ par rapport à $\Gamma$:

$$\Delta_2 : \gamma_i \to \begin{cases} \gamma_{j_1} & \text{avec prob. } q^i_{j_1} \\ \gamma_{j_2} & \text{avec prob. } q^i_{j_2} \\ & \vdots \\ \gamma_{j_n} & \text{avec prob. } q^i_{j_n} \end{cases}$$

En fait, toute 'fonction' $\Delta$ de la forme ci-dessus correspond à une machine de Turing probabiliste correcte en autant que pour chaque transition, les transitions $\gamma_i \to \gamma^i_{j_k}$ (pour $k$ de 1 à $n$) se fassent en une seule opération d'une machine de Turing standard. et que la somme des $q^i_{j_k}$ soit 1.

Pour représenter la machine de Turing en terme de chaîne de Markov, on doit 'convertir' les configurations $\gamma_1, \ldots, \gamma_w$ et la fonction de transition $\Delta_2$ en termes de vecteurs et d'opérateurs respectivement. Considérons $\mathbb{R}^w$, l'espace vectoriel réel de dimension $w$ et soit $\{e_1, \ldots, e_w\}$ une base de cet espace. Par convention, la configuration $\gamma_i$ est représentée par le vecteur $e_i$ dans $\mathbb{R}^w$. L'opérateur $D_{\Delta_2}$ associé à $\Delta_2$ est une matrice $w \times w$ où les données sont définies comme suit: à chaque transition de la forme ci-dessus, on a que

$$\begin{aligned} [D_{\Delta_2}]_{ij_k} &= q^i_{j_k} \qquad \text{pour } 1 \leq k \leq n \\ [D_{\Delta_2}]_{il} &= 0 \qquad \forall\, l \neq j_1, \ldots, j_n \end{aligned}$$

Soit $k_0$ tel que $\gamma_{k_0}$ soit la configuration initiale de $M(x)$. Le vecteur $v_0 = e_{k_0}$ représente l'état initial de $M$ dans l'espace réel de dimension $w$. Soit $v_1 = D_{\Delta_2} v_0$. Ce vecteur contient toute l'information de l'application de la première règle probabiliste de $\delta_2$ de $M$. En général, soit la suite $v_0, v_1, \ldots, v_m$ telle que $v_0 = e_{k_0}$ et pour tout $1 \leq i \leq m$, $v_i = D_{\Delta_2} v_{i-1}$. Les composantes de chacun des $v_i$ donnent les probabilités d'obtenir la configuration correspondante à cette étape du calcul. La chaîne des $v_i$ représente donc tous les calculs possibles par la machine $M$ sur la donnée $x$. Si $v_m$ est le vecteur correspondant à la fin des calculs, alors les composantes de ce vecteur donnent les probabilités avec lesquelles chacune des configurations finales peut être obtenue.

Mais cette représentation, bien que pratique, comporte une certaine fausseté: ces calculs ne sont que *possibles*. En réalité, un seul calcul sera exécuté. L'ordinateur quantique, décrit à la prochaine section, n'aura pas cette contrainte: tous les calculs seront exécutés en parallèle!

**1.2. Le modèle quantique: première partie.** Une machine de Turing (ou ordinateur) quantique possède les mêmes composantes qu'une machine de Turing déterministe. Tout comme avec le cas probabiliste, la différence se trouve dans la fonction de transition; plutôt que d'être pondérée par des probabilités, celle-ci sera pondérée avec des *amplitudes*.

Soit $M$ une machine de Turing quantique, alors sa fonction de transition $\delta$ aura la forme suivante:

$$\delta : (b,q) \rightarrow \begin{cases} (b_1, q_1, d_1) & \text{avec amplitude } \alpha_1^{bq} \\ (b_2, q_2, d_2) & \text{avec amplitude } \alpha_2^{bq} \\ & \vdots \\ (b_n, q_n, d_n) & \text{avec amplitude } \alpha_n^{bq} \end{cases}$$

Ou encore, par rapport à l'ensemble $\Gamma$ (voir section précédente):

$$\Delta : \gamma_i \rightarrow \begin{cases} \gamma_{j_1} & \text{avec amplitude } \beta_{j_1}^i \\ \gamma_{j_2} & \text{avec amplitude } \beta_{j_2}^i \\ & \vdots \\ \gamma_{j_n} & \text{avec amplitude } \beta_{j_n}^i \end{cases}$$

Encore une fois, les transitions $\gamma_i \rightarrow \gamma_{j_1}, \ldots, \gamma_i \rightarrow \gamma_{j_n}$ doivent s'exécuter en une seule opération d'une machine de Turing. De manière à forcer notre modèle a suivre les règles de la mćanique quantique, dit que si $M$ est dans l'état $\gamma_i$ alors une application de $\Delta$ mettra $M$ en *superposition* des états $\gamma_{j_1} \ldots \gamma_{j_n}$: $M$ est *en même temps* dans chacune des configurations $\gamma_{j_1}, \ldots, \gamma_{j_k}$. Les amplitudes $\beta_{j_1}^i, \ldots, \beta_{j_n}^i$ associées aux états $\gamma_{j_1}, \ldots, \gamma_{j_n}$ sont des nombres complexes tels que si l'utilisateur tente d'observer $M$, il verra *un seul* $\gamma_{j_k}$ avec probabilité $\left\|\beta_{j_k}^i\right\|^2$. On serait tenté de croire qu'à l'instar des machines de Turing probabilistes, la seule contrainte dictée par la mécanique quantique sur les amplitudes soit que leur norme au carré somme à 1, mais cela ne suffit pas. Pour être valide, ces transition doivent être globalement *unitaire*. Cette notion d'unitarité sera définie à la prochaine section. Toutefois, une conséquence directe est que chaque transition de $\delta$ doit être au moins *réversible*: à tout moment, la machine doit posséder toute l'information nécessaire pour renverser la vapeur et 'décalculer' ce qui fut calculé. Cette contrainte en apparence pénible à vérifier correspond à une notion très simple lorsque l'on considère la représentation vectorielle de la machine de Turing. Avant de décrire cette représentation, il est toutefois nécessaire d'ouvrir une parenthèse sur la notation qui sera utilisée.

1.2.1. *Espace de Hilbert et notation de Dirac.* La méthode de la section 1.1 pour convertir la fonction de transition $\Delta_2$ en un opérateur $D_{\Delta_2}$ sur les réels peut être appliquée directement sur la fonction de transition de la machine quantique pour obtenir cette fois un opérateur sur les complexes.[2]

En physique quantique, l'état d'un système se représente par un vecteur dans un espace vectoriel complexe, dit espace de Hilbert. Une notation introduite par Dirac pour manipuler plus aisément les vecteurs et les opérateurs d'un espace de Hilbert peut être trouvée dans [20]. Nous en présentons ici que les grandes lignes.

Soit $\mathcal{H}$ un espace de Hilbert de dimension $n$ et soit

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}$$

un vecteur normalisé de $\mathcal{H}$ en terme de ses composantes selon une base quelconque. Ce vecteur s'écrit avec un *ket* ($|\cdot\rangle$) comme suit

$$|\varphi\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}$$

Le même vecteur $\varphi$ écrit dans un *bra* ($\langle\cdot|$) note le vecteur conjugué transposé

$$\langle\varphi| = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*)$$

[2] Il est possible de représenter une machine de Turing quantique par un opérateur sur les réels (voir [3]), mais cette représentation est un peu plus lourde à manipuler.

La juxtaposition d'un bra et d'un ket est le produit matriciel des deux vecteurs. Par exemple, soit

$$|\varphi'\rangle = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix}$$

alors

$$\begin{aligned} \langle\varphi|\varphi'\rangle &= (\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*) \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} \\ &= \sum_{i=1}^{n} \alpha_i^* \beta_i \end{aligned}$$

C'est donc dire que $\langle\cdot|\cdot\rangle$ est le produit scalaire des vecteurs $\varphi$ et $\varphi'$. Le produit d'un ket et d'un bra

$$\begin{aligned} |\varphi\rangle\langle\varphi'| &= \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} (\beta_1^*, \beta_2^*, \ldots, \beta_n^*) \\ &= \begin{pmatrix} \alpha_1\beta_1^* & \alpha_1\beta_2^* & \ldots & \alpha_1\beta_n^* \\ \alpha_2\beta_1^* & \alpha_2\beta_2^* & \ldots & \alpha_2\beta_n^* \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_n\beta_1^* & \alpha_n\beta_2^* & \ldots & \alpha_n\beta_n^* \end{pmatrix} \end{aligned}$$

est un opérateur qui envoie un vecteur $|\xi\rangle$ dans $|\varphi\rangle\langle\varphi'|\xi\rangle$. Un cas particulier de cette classe d'opérateur est lorsque $|\varphi\rangle = |\varphi'\rangle$. Ces opérateurs sont dits *opérateurs de projection orthogonale* car ils envoient un vecteur $|\xi\rangle$ dans le vecteur de longueur $\langle\varphi|\xi\rangle$ dans la direction $|\varphi\rangle$.

Nous pouvons maintenant établir la correspondance entre un état de la machine de Turing quantique et sa représentation vectorielle. Soit $|e_1\rangle, |e_2\rangle, \ldots$ une base de $\mathcal{H}$ et soit $n = w$, $w$ étant la taille de l'ensemble $\Gamma$. On associe à chaque configuration $\gamma_i$ le vecteur de base $|e_i\rangle$ et on pose pour tout $1 \leq i \leq w$

$$|\gamma_i\rangle = |e_i\rangle$$

En général, une machine de Turing quantique se trouve en superposition de plusieurs (voire toutes) ses configurations. Létat $|\varphi\rangle$ qui représente cette superposition est un vecteur normé de la forme

$$\begin{aligned} |\varphi\rangle &= \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} \\ &= \sum_{i=1}^{n} \alpha_i |\gamma_i\rangle \end{aligned}$$

où $\alpha_i$ est l'amplitude associée à la configuration $\gamma_i$.

**1.3. Le modèle quantique: deuxième partie.** Nous en étions à convertir la fonction de transition $\delta$ en un opérateur. Par la même méthode qu'à la section 1.1; soit l'opérateur $D_\Delta$ sur les complexes où les données sont définies comme suit: pour chaque transition on a que

$$\begin{aligned} [D_\Delta]_{ij_k} &= \beta^i_{j_k} \qquad \text{pour } 1 \leq k \leq n \\ [D_\Delta]_{il} &= 0 \qquad \forall\, l \neq j_1, \ldots, j_n \end{aligned}$$

La théorie quantique stipule que l'évolution d'un système quantique doit être *unitaire*: l'opérateur agissant sur l'état du système doit être un opérateur unitaire (on rappelle qu'un opérateur $A$ est dit unitaire si $A^{-1} = A^\dagger$ où $A^\dagger$ est le conjugué transposé de $A$). Il s'ensuit que la contrainte sur les amplitudes est que l'opérateur $D_\Delta$ soit unitaire. Le fait que l'évolution d'un processus soit unitaire rend celui-ci entièrement réversible, d'où la remarque à la fin de la section 1.2. En apparence, cette contrainte sur les algorithmes pourrait diminuer la puissance de calcul du modèle quantique, mais Charles Bennett [2] et Yves Lecerf [17] ont démontré que tout algorithme peut être converti en algorithme réversible sans augmenter significativement le temps d'exécution.

Comme précédemment, soit $k_0$ tel que $\gamma_{k_0}$ est la configuration initiale de $M$ sur la donnée $x$. $|\varphi_0\rangle = |\gamma_{k_0}\rangle$ représente donc l'état initial de $M(x)$ dans l'espace de Hilbert. Soit la suite de vecteurs $|\varphi_0\rangle, |\varphi_1\rangle, \ldots, |\varphi_n\rangle$ telle que pour tout $1 \leq i \leq n$, $|\varphi_i\rangle = D_\Delta|\varphi_{i-1}\rangle$. Si $|\varphi_n\rangle$ est l'état final du calcul de $M$ sur la donnée $x$ et si

$$|\varphi_n\rangle = \sum_{i=1}^{w} \beta_i |\gamma_i\rangle$$

alors comme dans le cas du modèle probabiliste, l'interprétation de ce vecteur est que si l'usager tente de lire la configuration de $M$, il verra la configuration $\gamma_i$ avec probabilité $\|\beta_i\|^2$.

Pour mettre en évidence la différence de fonctionnement du modèle quantique par rapport au modèle probabiliste, deux notions doivent être présentées: les interférences quantiques (dues aux amplitudes) et le concept d'observation. Ces sujets seront traités dans les deux sections suivantes.

**1.4. Interférence quantique.** Il est important de souligner ici la différence entre *probabilité* et *amplitude*. L'exemple suivant montre à quel point ces amplitudes peuvent engendrer des situations contre-intuitives. Soit un ordinateur quantique dont l'opérateur $U$ associé à la fonction de transition est tel que $U_{ik} = \alpha_1$ et $U_{jk} = \alpha_2$. Soit l'état initial $|\varphi\rangle$ la superposition ${}^1\!/\!\sqrt{2}\,(|\gamma_i\rangle + |\gamma_j\rangle)$. Le problème est de déterminer avec quelle probabilité $p$ l'état $q_k$ peut être atteint (après une application de $U$). Cette probabilité est la norme au carré de l'amplitude d'atteindre $q_k$. D'après la définition de $U$ et de l'état initial, cette amplitude est $\frac{1}{\sqrt{2}}(\alpha_1 + \alpha_2)$. Donc $p = {}^1\!/_2\|\alpha_1 + \alpha_2\|^2$. Considérons deux cas extrêmes. D'abord, si $\alpha_1 = \alpha_2$ alors $p = 2\|\alpha_1\|^2$, ce qui est deux fois plus que la probabilité "classique" intuitive. Le second cas est encore plus surprenant. Les amplitudes n'étant pas limitées à être positives, considérons le cas où $\alpha_1 = -\alpha_2$. Dans ce cas $p = 0$ et malgré qu'il y ait *deux* voies possibles pour atteindre l'état $\gamma_k$, *il est impossible de trouver l'ordinateur dans cette configuration.* Ce genre d'effet d'interférence, jumelé avec la notion d'observable de la section suivante, semble donner à l'ordinateur quantique un avantage sur la machine de Turing.

**1.5. Observation.** Lorsqu'il est nécessaire de regarder le résultat d'une machine de Turing, aucun problème particulier ne se pose. Par contre, pour un ordinateur quantique dont le ruban (et donc par extension l'ordinateur lui-même) peut se trouver en superposition quantique, l'observation n'est plus aussi simple. Par exemple, si une seule case du ruban est en superpostion de **0** et de **1**, l'ordinateur au complet se trouve en superposition de deux configurations. Pire: si le ruban contient $n$ cases en superposition, cela représente possiblement une superposition de $2^n$ configurations! Le volume même de l'information en rend la lecture difficile. De plus les principes fondamentaux de la mécanique quantique limitent fortement le type d'information qui peut être extraite d'une telle superposition. C'est ici qu'entre en jeu la notion d'*observable*. Un observable $\mathcal{O}$ est un partition de l'espace de calcul en sous-espaces $E_1, \ldots, E_k$ deux à deux orthogonaux. Soit un vecteur d'état $|\varphi\rangle$. Ce vecteur a une représentation unique de la forme

$$|\varphi\rangle = \sum_{i=0}^{k} \alpha_i |\varphi_i\rangle$$

où $|\varphi_i\rangle$ est la projection de $|\varphi\rangle$ dans $E_i$. Les principes de la mécanique quantique stipulent que l'acte d'observer le système $|\varphi\rangle$ avec l'observable $\mathcal{O}$ a pour effet de choisir *un* des $|\varphi_i\rangle$ avec probabilité $\|\alpha_i\|^2$. La seule information qui transpire de l'observation est $i$ (le sous-espace choisi). L'état résiduel après l'observation est $|\varphi_i\rangle$ (normalisé). Toute autre information se trouvant dans $|\varphi\rangle$ est irrévocablement détruite.

Il existe une notion plus générale d'observable, dites *mesure généralisée* (*Positive-Operator-Valued measurements* ou POM) dont nous omettons la description par souci de clarté. Soulignons seulement que les résultats contenus dans cette thèse demeurent valides sous les mesures généralisées. Pour plus de détail, consulter [14].

## 2. Résultats de complexité quantique

Il est possible maintenant de présenter quelques tâches pour lesquelles l'ordinateur quantique possède un net avantage sur la machine de Turing. En fait, le problème suivant est même impossible pour celle-ci. La tâche est de définir une opération appelée $\sqrt{\mathit{Non}}$. Lorsqu'appliquée *deux fois* sur un bit $b$, cette opération retourne le bit $\neg b$ (*non* $b$). Il est aisé de voir qu'une telle opération ne peut être exécutée sur une machine de Turing n'utilisant qu'un seul bit. Par contre, on peut très facilement définir une fonction de transition (c'est-à-dire, un opérateur pour $\sqrt{\scriptstyle Non}$) ayant exactement la propriété voulue. Préalablement, les états utilisés doivent être définis. Il est commun de nommer ces derniers par le contenu du ruban et d'ignorer l'état et la position de la tête. Cette convention permet d'alléger la notation. Ici, puisque l'opération $\sqrt{\scriptstyle Non}$ n'affecte qu'un seul bit (donc une seule case du ruban), les états nécessaires sont $|\mathbf{0}\rangle$ et $|\mathbf{1}\rangle$. L'espace d'états du calcul est $\mathbb{C}^2$ et l'opérateur pour $\sqrt{\scriptstyle Non}$ (appelé lui aussi $\sqrt{\scriptstyle Non}$ pour simplifier la notation) est donné par

$$\sqrt{\scriptstyle Non} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

Dans cette notation, les états $|\mathbf{0}\rangle$ et $|\mathbf{1}\rangle$ correspondent aux vecteurs

$$|\mathbf{0}\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{et} \qquad |\mathbf{1}\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Si l'ordinateur est initialisé avec un $\mathbf{0}$ sur son ruban (c'est-à-dire dans l'état $|\mathbf{0}\rangle$), la double application de l'opérateur $\sqrt{\scriptstyle Non}$ produit l'effet suivant:

$$\begin{aligned} \sqrt{\scriptstyle Non}\,[\sqrt{\scriptstyle Non}|\mathbf{0}\rangle] &= \sqrt{\scriptstyle Non}\left[\frac{1}{\sqrt{2}}|\mathbf{0}\rangle + \frac{1}{\sqrt{2}}|\mathbf{1}\rangle\right] \\ &= {}^1\!/_2(|\mathbf{0}\rangle + |\mathbf{1}\rangle) + {}^1\!/_2(-|\mathbf{0}\rangle + |\mathbf{1}\rangle) \\ &= |1\rangle \end{aligned}$$

Lorsque l'ordinateur est initialisé dans l'état $|\mathbf{0}\rangle$ il termine dans l'état $|\mathbf{1}\rangle$ après deux applications de $\sqrt{\scriptstyle Non}$. De la même manière, il termine dans l'état $-|\mathbf{0}\rangle$ s'il est initialisé dans $|\mathbf{1}\rangle$. Dans les deux cas, une observation de l'état final donne avec probabilité 1 la négation du bit original.

Cet exemple a pour but premier de souligner le jeu des interférences: soit $b$ le bit en donnée, alors l'opérateur $\sqrt{\scriptstyle Non}$ est construit de telle sorte que la *seconde* application détruit exactement la composante selon $|b\rangle$ créée lors première application. Ce jeu d'interférences constructives et destructives est à la base d'un autre problème intéressant, appelé *problème de Deutsch-Jozsa.*

**2.1. Problème de Deutsch-Jozsa.** Le problème ci-dessus n'étant défini que pour une seule taille d'entrée (1 bit), il ne peut constituer un argument pour démontrer l'avantage que possède l'ordinateur quantique sur la machine de

Turing. Il faudrait au minimum en étudier la généralisation à $n$ bits. Deutsch et Jozsa (dans [11]) présentent toutefois un problème bien plus amusant et défini sur des données de taille arbitraire. En voici l'énoncé:

**Donnée:** une chaîne de $n$ bits $b = b_1, \ldots, b_n$ ($n = 2^k$, $k > 0$).

**Problème:** Des deux affirmations suivantes, en donner une qui est vraie à propos de la chaîne $b$.

1- La chaîne $b$ est **non constante**

2- La chaîne $b$ est **non équilibrée**

où **non constante** signifie que $b$ contient au moins un $\mathbf{0}$ *et* un $\mathbf{1}$ et **non équilibrée** signifie que $b$ ne contient pas le même nombre de $\mathbf{0}$ que de $\mathbf{1}$.

Pour simplifier, la lecture des bits de $b$ se fera en accès direct. Une première chose à constater, c'est que les affirmations 1 et 2 ne constituent pas une partition de l'ensemble des chaînes de $n$ bits. En effet, la grande majorité des chaînes sont en même temps non constantes et non équilibrées. En fait, un algorithme répondant toujours “non constante” donnerait la bonne réponse dans la presque totalité des cas, ne se trompant que si $b$ contient seulement des $\mathbf{0}$ ou seulement des $\mathbf{1}$. Mais le problème ne tolère aucune possibilité d'erreur. Il est aisé de voir que dans ces conditions, même un algorithme probabiliste ne peut garantir sa réponse sans avoir regardé au moins ${}^{n}\!/_{2}+1$ bits. Clairement, ce problème prend un temps $\Omega(n)$ en pire cas. Il existe cependant un algorithme [11] pour l'ordinateur quantique qui accomplit le même travail en temps $O(\log n)$. Les états nécessaires sont les $|i, b\rangle$ où $i$ est une chaîne de bits de longueur $\lg n$ et $b$ est un bit. L'ordinateur est initialisé dans l'état

$$|\varphi\rangle = |\underbrace{\mathbf{0}\ldots\mathbf{0}}_{\lg n}, \mathbf{0}\rangle$$

La première étape de l'algorithme consiste à créer une superposition de toutes les $n$ chaînes de longueur $\lg n$. Ceci s'accomplit avec l'opérateur

$$U_1 = \left[\bigotimes_{k=1}^{\lg n} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\right] \bigotimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(la notation $\otimes$ est le produit tensoriel et est définie au chapitre 4, section 6).

$$U_1|\varphi\rangle = {}^1\!/\!_{\sqrt{n}} \sum_{i=0}^{n-1} |i, \mathbf{0}\rangle$$

Bien sûr, le $i$ dans $|i, \mathbf{0}\rangle$ doit être pris comme la décomposition binaire de la valeur $i$. La superposition créée, l'ordinateur exécute l'opération "si l'état est $|i, \mathbf{0}\rangle$ alors lire le bit $b_i$ et l'écrire réversiblement sur le ruban". On note que l'écriture doit se faire de manière réversible. On laisse le lecteur se convaincre que remplacer le bit $\mathbf{0}$ par le bit $\mathbf{0} \oplus b_i$ est réversible. Soit $U_2$ l'opérateur correspondant, alors

$$U_2U_1|\varphi\rangle = {}^1\!/\!_{\sqrt{2^n}} \sum_{i=0}^{n-1} |i, b_i\rangle$$

La prochaine action est "$|i, b_i\rangle \rightarrow (-1)^{b_i}|i, b_i\rangle$". Si l'énumération naturelle des chaînes de $\lg n + 1$ bits est prise comme base, l'opérateur $U_3$ qui exécute cette action est une matrice diagonale avec les valeurs 1 et $-1$ en alternance. Donc

$$U_3U_2U_1|\varphi\rangle = {}^1\!/\!_{\sqrt{n}} \sum_{i=0}^{n-1} (-1)^{b_i}|i, b_i\rangle$$

Il ne reste plus que la dernière opération. La raison d'être de celle-ci apparaîtra claire plus tard. Elle n'est que l'inverse logique de la lecture faite avec $U_2$. L'opérateur correspondant est, justement, $U_2$. Le programme final est

$$U_2U_3U_2U_1|\varphi\rangle = {}^1\!/\!_{\sqrt{n}} \sum_{i=0}^{n-1} (-1)^{b_i}|i, \mathbf{0}\rangle$$

Contrairement à l'exemple $\sqrt{\mathit{Non}}$, un observable doit être défini pour extraire la réponse désirée. L'observable $\mathcal{O}$ possède deux sous-espaces propres $\mathsf{Yin}$ et $\mathsf{Yang}$ engendrés par

$$\mathsf{Yin} = {}^1\!/\!_{\sqrt{n}} \sum_{j=0}^{n-1} |j, \mathbf{0}\rangle \qquad \text{et} \qquad \mathsf{Yang} = \mathsf{Yin}^{\perp}$$

$\mathsf{Yin}$ est donc un sous-espace à une dimension le long du vecteur ci-haut et $\mathsf{Yang}$ en est le sous-espace orthogonal complémentaire. Pour déterminer le résultat de l'observation sur $|\varphi'\rangle = U_2U_3U_2U_1|\varphi\rangle$, il faut trouver les projections de $|\varphi'\rangle$ sur

$\mathsf{Yin}$ et $\mathsf{Yang}$. Le produit scalaire de $|\varphi'\rangle$ et $\mathsf{Yin}$ est

$$\begin{aligned} \langle \mathsf{Yin}|\varphi'\rangle &= {}^1\!/_n \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} (-1)^{b_i} \langle j, \mathbf{0}|i, \mathbf{0}\rangle \\ &= {}^1\!/_n \sum_{i=0}^{n-1} (-1)^{b_i} \end{aligned}$$

(Note: puisque les $|i, \mathbf{0}\rangle$ sont orthogonaux, $\langle i, \mathbf{0}|j, \mathbf{0}\rangle = 1$ si $i = j$ et 0 sinon.) Le comportement de $\mathcal{O}$ peux être analysé comme suit:

(i) Si $b$ **est constante.** Dans ce cas, $\langle \mathsf{Yin}|\varphi'\rangle = 1$ ou $-1$ selon le cas. Donc avec probabilité $\|\langle \mathsf{Yin}|\varphi'\rangle\|^2 = 1$, le résultat de l'observation est $\mathsf{Yin}$.
(ii) Si $b$ **est équilibrée.** Dans ce cas, $\langle \mathsf{Yin}|\varphi'\rangle = 0$. Donc, la probabilité d'avoir un résultat $\mathsf{Yin}$ est zéro.
(iii) Si $b$ **est ni l'une ni l'autre.** Alors il y a une certaine probabilité non-nulle que le résultat soit $\mathsf{Yin}$.

Par cette analyse, si l'observable $\mathcal{O}$ donne le résultat $\mathsf{Yin}$, il est certain que la chaîne $b$ est non équilibrée. De même, si le résultat est $\mathsf{Yang}$, il est certain que $b$ est non constante. Le programme ci-dessus résout bien le problème et la construction de chacun des opérateurs utilisés se fait en temps $O(\log n)$, comme promis.

Malheureusement, la force de ce résultat se voit légèrement diminuée par le fait que le problème de Deutsch-Jozsa ne peut se définir comme une fonction de l'ensemble des chaînes dans $\{\mathbf{0}, \mathbf{1}\}$. Mais le vrai point faible du problème de Deutsch-Jozsa est qu'il peut être résolu en temps constant sur une machine de Turing probabiliste classique si des erreurs sont tolérées. Il n'en demeure pas moins que ce problème servira de base aux résultats présentés au chapitre 3 sur le calcul quantique avec oracles. La section suivante porte sur les travaux de Jozsa [15] spécifiquement sur le calcul des fonctions.

**2.2. Calculabilité par superposition quantique.** L'ordinateur quantique peut calculer une grande quantité de valeurs en relativement peu de temps grâce au parallélisme offert par les superpositions quantiques. Le problème est de construire un observable capable d'extraire l'information utile de ces superpositions. Il faut donc déterminer quelles fonctions peuvent être transformées en un observable. La définition et le théorème suivant sont dus à Jozsa [15].

DÉFINITION 2.1. *Soit* $f : \{\mathbf{0},\mathbf{1}\}^n \to \{\mathbf{0},\mathbf{1}\}$ *et soit* $|\varphi\rangle = 2^{-n/2}\sum_{i=0}^{2^n-1}|i, f(i)\rangle$ *la superposition des différentes valeurs de* $f$. *Soit une propriété* $G$ *de* $f$ *représentée par la fonction* $G : \{\mathbf{0},\mathbf{1}\}^{2^n} \to \{\gamma_1,\ldots,\gamma_k\}$. *On dira que* $G$ *est* ***calculable par parallélisme quantique*** *s'il existe un observable* $\mathcal{O}$ *séparant l'espace de Hilbert en* $k+1$ *sous-espace* $E_{\gamma_1},\ldots,E_{\gamma_k}$ *et* $E_?$ *tel que le résultat* $R$ *d'une observation de* $|\varphi\rangle$ *avec* $\mathcal{O}$ *possède les propriétés suivantes: si* $G(f(1),\ldots,f(2^n-1)) = \gamma_{i_0}$ *alors*

(i) $\text{Prob}\left[R = E_?\right] \geq 0$
(ii) $\text{Prob}\left[R = E_{\gamma_{i_0}}\right] > 0$
(iii) $\text{Prob}\left[R \neq E_{\gamma_{i_0}} \wedge R \neq E_?\right] = 0$

*En d'autres termes,* $\mathcal{O}$ *donne avec probabilité non nulle la bonne réponse, ne donne jamais une mauvaise réponse mais peut parfois donner un réponse inconclusive (*$E_?$*).*

Il est à noter que cette définition de calculabilité est très large puisqu'elle permet que l'observable associé puisse ne pas donner de réponse conclusive (le résultat $E_?$). Malgré cette largesse, le théorème suivant permet de conclure que très peu de fonctions $G$ sont calculables par parallélisme quantique.

THÉORÈME 2.1. *Si le tripatouillage de* $f$ *est interdit (*$f$ *est dans une boite noire) alors les seules fonctions* $G : \{\mathbf{0},\mathbf{1}\}^{2^n} \to \{\gamma_1,\ldots,\gamma_k\}$ *qui soient calculable par parallélisme quantique sont*

- *la fonction constante* $\mathbf{0}$
- *la fonction projection* $(G(f(0),\ldots,f(2^n)) = f(i)$, *où* $0 \leq i \leq 2^n)$
- *le ou-exclusif de deux valeurs* $(G(f(0),\ldots,f(2^n)) = f(i) \oplus f(j)$, *où* $0 \leq i,j \leq 2^n$, $i \neq j)$

*ainsi que leur négation. De plus, la probabilité d'obtenir une bonne réponse est* $1$ *pour la fonction constante et* $2^{-n}$ *pour les autres.*

Non seulement ceci représente très peu de fonctions, mais une machine de Turing probabiliste peut faire tout aussi bien sur un bon nombre d'entre elles. Calculer une fonction constante n'a rien de difficile. Pour simuler le comportement de l'ordinateur quantique sur une projection $G_i$, une machine de Turing probabiliste n'a qu'à choisir un $x$ au hasard, calculer $G_x \equiv f_x$ et, si $x = i$, alors donner la réponse, sinon dire "*Je ne sais pas*". Par contre, cette technique de simulation ne fonctionne pas avec le ou-exclusif de deux valeurs (la bonne valeur serait obtenue avec probabilité $2^{-2n}$ au lieu de $2^{-n}$). En fait, c'est la seule fonction (avec sa négation) sur laquelle l'ordinateur quantique possède un avantage, si faible soit-il, sur la machine de Turing.

# CHAPITRE 3

## Calcul quantique avec oracles

Après avoir vu au chapitre précédent comment créer en temps linéaire une superposition d'un nombre exponentiel de valeurs, le réflexe naturel d'un théoricien en complexité est d'y voir une manière de résoudre efficacement le problème de satisfiabilité d'expressions booléennes. L'intuition est simple: soit une expression booléenne $p$ à $n$ variables. En temps $\mathrm{O}(n)$, on crée la superposition de toutes les $2^n$ assignations des $n$ variables. En temps polynomial, on évalue (par parallélisme quantique) $p$ sur ces assignations et il ne reste plus qu'à faire une observation sur l'état final pour savoir si au moins une des assignations force $p$ à être vraie. Dans la notation de Jozsa, on veut un observable $\mathcal{O}$ pour la fonction $G : \{\mathbf{0}, \mathbf{1}\}^{2^n} \to \{\mathbf{0}, \mathbf{1}\}$ où une réponse $\mathbf{1}$ indique qu'il existe une assignation donnant à $p$ la valeur booléenne vrai. Cette fonction est donc simplement la fonction booléenne OU sur $2^n$ bits. Malheureusement, le théorème de Jozsa sur les fonctions calculables par parallélisme quantique indique qu'un tel observable n'existe pas.

Malgré tout, il reste toujours le problème de Deutsch-Jozsa. Ce problème échappe aux contraintes du théorème de Jozsa puisque la propriété calculée par l'observable n'est pas une fonction mais une relation. Cette même propriété le fait sortir du cadre de la complexité du calcul. S'il était possible de transformer le problème de Deutsch-Jozsa en une fonction, tout en le soustrayant au théorème de Jozsa *et* en préservant l'avantage du modèle quantique sur la machine de Turing, alors il serait réellement possible de comparer les modèles de calcul classique (machine de Turing) et quantique du point de vue de la complexité. Ceci peut être réalisé dans le contexte des *problèmes à promesse* [11]. Sous la promesse que les chaînes en entrée soient ou bien nulles, ou bien équilibrées, alors le problème de Deustch est simplement de calculer le OU de $n$ bits. Cette promesse n'affecte pas l'analyse faite au chapitre précédent: le problème reste dans $\Omega(n)$ pour une machine de Turing et dans $\mathrm{O}(\log n)$ pour l'ordinateur quantique. Par contre, le problème étant sous forme de fonction, il peut être étudié avec la complexité du calcul.

Dans ce chapitre, nous étudierons la puissance du problème de Deutsch-Jozsa dans le contexte du calcul relativisé, c'est-à-dire en supposant l'existence d'un oracle arbitrairement complexe pouvant répondre de façon instantanée à une question précise. Les prochaines sections définiront d'abord quelques notions de complexité et nous terminerons par une série de théorèmes portant sur le calcul quantique relativisé.

## 1. Classe de complexité quantique

On suppose que les classes de complexité classiques telles **P**, **NP** et **PSPACE** sont connues. L'analogue quantique de la classe **P** est **EQP** (de l'anglais **E**xact **Q**uantum **P**olynomial-time). C'est la classe des problèmes de décision calculables en temps polynômial pire cas sur l'ordinateur quantique. On remarque que dans ce contexte, il est permis à l'ordinateur quantique d'utiliser sa pleine puissance de superposition et d'interférence pour autant qu'en aucun cas le calcul ne prenne un temps plus que polynomial ou qu'il existe une probabilité d'erreur non nulle. Benioff [1] a déjà démontré que $\mathbf{P} \subseteq \mathbf{EQP}$. La question naturelle est donc de savoir si cette inclusion est stricte. Existe-il un problème de décision calculable en temps polynomial quantique pire cas, mais nécessitant un temps super-polynomial sur une machine de Turing?

## 2. Notation et définitions

Soit $\Sigma$ l'alphabet $\{\mathbf{0}, \mathbf{1}\}$ et soit $X \subseteq \Sigma^*$. On définit le prédicat $B(X)$ est vrai si pour tout $n \in \mathbb{N}$, on a que, soit $X \cap \Sigma^n = \emptyset$, ou bien il y a exactement $2^{n-1}$ chaînes dans $X \cap \Sigma^n$. L'ensemble associé $S_X$ est défini comme

$$S_X = \{1^n \mid X \cap \Sigma^n = \emptyset\}.$$

Un *modèle de calcul à oracle* est un modèle de calcul (déterministe, probabiliste, non-déterministe ou quantique) muni d'un ensemble auquel il est possible poser des questions. L'oracle est un sous-ensemble $X$ de $\Sigma^*$ qui, sur la question "$x$?", répond "Oui" ou "Non" selon que la chaîne $x$ appartienne à $X$ ou non. Écrire une question de taille $n$ prend $n$ étapes et la réponse de l'oracle s'obtient en une seule étape. Dans le cas d'un modèle quantique, l'oracle se superpose lui aussi (chaque état superposé a accès à sa propre version de l'oracle superposé). Sous ces hypothèses, la solution au problème de Deutsch-Jozsa (voir section 2.1) implique que l'ensemble $S_X$ peut être reconnu en temps linéaire avec l'ordinateur quantique si $B(X)$ est vrai et que le modèle a accès à l'oracle $X$.

Si $M$ est un modèle de calcul et $X \subseteq \Sigma^*$, on note par $M^X$ le même modèle ayant accès à l'oracle $X$. De même, soit $C$ une classe de complexité, on note $C^X$ la classe $C$ où le modèle de calcul a accès à l'oracle $X$. Pour reformuler le théorème de Deutsch-Jozsa: $S_X \in \mathbf{EQP}^X$ pour tout $X$ tel que $B(X)$ est vrai.

Soit $\sigma$ une bijection entre $\Sigma^*$ et $\mathbb{N}^+$, par exemple $\sigma(x) = \text{bin}(\mathbf{1}x)$ où "$\mathbf{1}x$" note la chaîne $x$ précédée d'un $\mathbf{1}$ et $\text{bin}(y)$ est l'entier dont la représentation binaire est $y$. On note que pour tout $x \in \Sigma^*$, si $|x| = k$ alors $\sigma(x) < 2^{k+1}$. Cette bijection nous permet donc d'encoder l'ensemble $S_X$ de façon exponentiellement plus compacte:

$$V_X = \{x \in \Sigma^* \mid X \cap \Sigma^{\sigma(x)} = \emptyset\}$$

Ainsi, $V_X$ peut être reconnu en temps exponentiel lorsque $S_X$ est reconnu en temps linéaire puisque $x \in V_X$ si et seulement si $1^{\sigma(x)} \in S_X$. Soit $\tau$ l'inverse fonctionnel de $\sigma$.

Soit $\alpha : \mathbb{N}^+ \to \mathbb{N}^+$ une fonction quelconque telle que pour une infinité de $n$ on ait que $\alpha(n) = i$. Soit $\wp : \mathbb{N}^+ \to \mathbb{N}^+$ défini comme suit:

$$\wp(n) = \begin{cases} 1 & \text{si } n = 1 \\ 2^{\wp(n-1)} & \text{si } n > 1 \end{cases}$$

## 3. Comparaisons de puissance

Les résultats présentés dans les quatre prochaines sections se retrouvent dans [4], [5] et [6]. La forme générale des théorèmes sera la même dans chaque cas: il existe un oracle $X$ relativement auquel les temps requit pour reconnaître $S_X$ (ou $V_X$) forme une séparation entre deux classes de complexité relativisée; l'une quantique, l'autre classique. L'essentiel de la preuve se trouvera dans la construction de l'oracle $X$ qui devra permettre une application judicieuse du problème de Deustch.

**3.1. Quantique versus déterministe.** Il a déjà été vu que $S_X \in \mathbf{EQP}^X$ pour tout ensemble $X \in \Sigma^*$ pour lequel $B(X)$ est vrai. Pour obtenir une séparation entre $\mathbf{P}$ et $\mathbf{EQP}$, il suffit de trouver un oracle $X \in \Sigma^*$ pour lequel $B(X)$ est vrai et $S_X \notin \mathbf{P}^X$. Pour ce faire, la technique de diagonalisation sera utilisée. À chaque étape de la construction, une machine de Turing déterministe à oracle sera éliminée en forçant celle-ci à donner une réponse erronée à propos de $S_X$ ou à prendre trop de temps. En fait, chaque machine sera éliminée une infinité de fois. (Ce théorème étant un cas particulier du théorème 3.4, nous ne donnerons ici qu'une esquisse de preuve.)

Théorème 3.1. *Il existe un oracle $X$ relativement auquel $S_X$ peut être reconnu en temps polynomial pire cas sur l'ordinateur quantique, mais toute machine de Turing déterministe classique reconnaissant $S_X$ doit prendre un temps exponentiel sur une infinité d'entrées.*

Preuve (esquisse). Soit $M_1, M_2, \ldots$ une énumération standard des machines Turing déterministes à oracle. La construction de $X$ se fait par étapes successives. Initialement, $X_1 = \emptyset$. Pour tout $n \geq 1$, on simule la machine $M_{\alpha(n)}$ sur l'entrée

$1^{\wp(n)}$ pendant au plus $2^{\wp(n)-1}$ étapes en utilisant $X_n$ comme oracle. À la fin de cette simulation, trois cas sont possibles:

- Si $M_{\alpha(n)}$ n'a pas terminé de calculer alors fixer $X_{n+1} = X_n$.
- Si $M_{\alpha(n)}$ a terminé et rejette son entrée alors fixer $X_{n+1} = X_n$.
- Sinon, $M_{\alpha(n)}$ accepte. Soit $Q_n$ l'ensemble des questions posées à l'oracle par la machine $M_{\alpha(n)}$. Au pire, la cardinalité de $Q_n$ est $2^{\wp(n)-1}$ puisqu'il faut au moins une étape pour poser une question. Considérant qu'il y a $2^{\wp(n)}$ questions de taille $\wp(n)$, on peut former un ensemble $R_n \subset \Sigma^{\wp(n)}$ de taille $2^{\wp(n)-1}$ tel que $R_n \cap Q_n = \emptyset$. Il ne reste qu'à fixer $X_{n+1} = X_n \cup R_n$ et à passer à l'étape suivante.

Soit $X = \bigcup_{n\geq 1} X_n$. Que $S_X$ puisse être reconnu en temps linéaire sur un ordinateur quantique avec l'oracle $X$ devrait apparaître évident par construction puisque l'on prend garde d'ajouter soit la moitié, soit aucune chaîne d'une longueur donnée. Également par construction, on peut prouver que toute machine de Turing déterministe qui résout $S_X$ prendra un temps exponentiel sur une infinité d'entrées. ⊠

Corollaire 3.2. *Il existe un oracle $X$ tel que* $\mathbf{P}^X \subsetneq \mathbf{EQP}^X$.

Preuve. Immédiate du théorème 3.1 puisque qu'un polynôme ne peut avoir une valeur plus grande qu'une exponentielle, sauf sur un nombre fini d'entrées. ⊠

Corollaire 3.3. *Soit une fonction $t : \mathbb{N} \to \mathbb{N}$ quelconque strictement croissante. Il existe un oracle relativement auquel il existe un ensemble reconnaissable en temps $\mathrm{O}(t(n))$ sur un ordinateur quantique mais toute machine de Turing reconnaissant cet ensemble devra prendre un temps $\Omega(2^{t(n)})$ sur une infinité d'entrées. En particulier, il existe un oracle séparant les classes exponentielles classique et quantique.*

Preuve. La preuve du théorème 3.1 s'applique directement pour autant que l'on remplace $S_X$ par $\hat{S}_X = \{1^n \mid X \cap \Sigma^{t(n)} = \emptyset\}$ et $\wp(n)$ par

$$\hat{\wp}(n) = \begin{cases} 1 & \text{si } n = 1 \\ 2^{t(\hat{\wp}(n-1))} & \text{si } n > 1 \end{cases}$$

Les détails sont laissés au lecteur. $\boxtimes$

**3.2. Quantique versus non-déterministe.** Par la même technique, il est possible d'obtenir une séparation entre **EQP** et **NP**. La seule difficulté est que chaque machine a maintenant, sur chaque entrée, un nombre exponentiel de calculs possibles. Pour éliminer une machine sur une entrée $x$, il faudra donc s'assurer qu'aucun des calculs acceptants ne prennent un temps polynomial ou qu'ils donnent une mauvaise réponse.

THÉORÈME 3.4. *Il existe un oracle $Y$ relativement auquel $S_Y$ peut être reconnu en temps linéaire pire cas sur l'ordinateur quantique, mais toute machine de Turing non-déterministe classique reconnaissant $S_Y$ doit prendre un temps exponentiel sur une infinité d'entrées acceptées.*

PREUVE. Soit $N_1, N_2, \ldots$ une énumération standard des machines de Turing non-déterministes à oracle. La construction de $Y$ se fait par étapes successives. À la fin de la $n^{\text{ième}}$ étape, $Y$ sera défini pour toutes les chaînes de longueur inférieure à $\wp(n+1)$. Initialement, $Y_1 = \emptyset$. Pour tout $n \geq 1$, on simule tous les chemins non-déterministes de $N_{\alpha(n)}$ sur l'entrée $1^{\wp(n)}$ pendant au plus $2^{\wp(n)-1}$ étapes par chemin en utilisant $Y_n$ comme oracle. À la fin de cette simulation, trois cas sont possible:

- Si aucun des calculs de $N_{\alpha(n)}$ n'a terminé, alors fixer $Y_{n+1} = Y_n$.
- Si tous les calculs de $N_{\alpha(n)}$ rejettent l'entrée, alors fixer $Y_{n+1} = Y_n$.
- Sinon, il existe au moins un calcul de $N_{\alpha(n)}$ acceptant l'entrée en moins de $2^{\wp(n)-1}$ étapes. Considérons le calcul acceptant le plus court et soit $Q_n$ l'ensemble des questions posées à l'oracle durant ce calcul. Au pire, la cardinalité de $Q_n$ est $2^{\wp(n)-1}$ puisqu'il faut au moins une étape pour poser une question. Considérant qu'il y a $2^{\wp(n)}$ questions de taille $\wp(n)$, on peut former un ensemble $R_n \subset \Sigma^{\wp(n)}$ de taille $2^{\wp(n)-1}$ tel que $R_n \cap Q_n = \emptyset$. fixer $Y_{n+1} = Y_n \cup R_n$ et passer à l'étape suivante.

Soit $Y = \bigcup_{n\geq 1} Y_n$. On remarque que la construction de $Y$ est telle que $B(Y)$ est vrai puisqu'à chaque étape, c'est-à-dire pour chaque longueur $\wp(n)$ de chaînes, soit aucune, soit exactement la moitié des chaînes de cette taille sont ajoutées.

Clairement, $S_Y$ peut donc être reconnu en temps linéaire sur un ordinateur quantique ayant accès à l'oracle $Y$. Considérons maintenant une machine du Turing non-déterministe $N_i$ prétendant reconnaître $S_Y$ avec l'oracle $Y$. Soit un entier $n$ tel que $\alpha(n) = i$. On peut faire l'assertion suivante:

ASSERTION 3.1. *$N_i^Y$ accepte l'entrée $1^{\wp(n)}$ et doit prendre plus de $2^{\wp(n)-1}$ étapes pour le faire.*

PREUVE. Montrons d'abord que $N_i^Y$ accepte $1^{\wp(n)}$. Preuve par contradiction: supposons que $N_i^Y$ rejette l'entrée $1^{\wp(n)}$. Par l'hypothèse que $N_i^Y$ reconnaît $S_Y$, on conclut que $1^{\wp(n)} \notin S_Y$ et donc que $Y \cap \Sigma^{\wp(n)} \neq \emptyset$. Mais la seule façon pour $Y$ de contenir des chaînes de taille $\wp(n)$, c'est d'avoir été dans le troisième cas ci-dessus lors de l'étape $n$. Donc $N_i$ accepte $1^{\wp(n)}$ en moins de $2^{\wp(n)-1}$ étapes en ayant accès à l'oracle $Y_n$. Considérons le calcul acceptant qui fut utilisé durant cette étape. $R_n$ ne contient que des questions non posées lors de ce calcul. De plus, aucune de ces questions ne peut avoir une taille supérieure à $2^{\wp(n)-1}$ car le temps aurait manqué pour formuler une si grande question. Puisque $Y$ et $Y_n$ ne diffèrent sur les chaînes de taille inférieure à $\wp(n+1) = 2^{\wp(n)}$ uniquement par le fait que les chaînes de $R_n$ sont dans $Y$ mais pas dans $Y_n$, le même chemin acceptant existe lorsque $N_i$ utilise $Y$ comme oracle plutôt que $Y_n$. Or, il suffit d'un seul chemin acceptant pour que $N_i^Y$ accepte, donc $N_i^Y$ accepte l'entrée $1^{\wp(n)}$. $\bowtie$

Montrons maintenant que $N_i^Y$ accepte $1^{\wp(n)}$ en plus de $2^{\wp(n)-1}$ étapes. Preuve par contradiction: supposons que $N_i^Y$ accepte $1^{\wp(n)}$ en au plus $2^{\wp(n)-1}$ étapes. Par l'hypothèse que $N_i$ reconnaît $S_Y$, on conclut que $1^{\wp(n)} \in S_Y$ et donc que $Y \cap \Sigma^{\wp(n)} = \emptyset$. Ceci entraîne que $Y$ et $Y_n$ s'accordent sur les chaînes de taille inférieure à $\wp(n+1) = 2^{\wp(n)}$ et donc que $N_i$ a un comportement identique sur $1^{\wp(n)}$ peu importe qu'elle utilise $Y$ ou $Y_n$ comme oracle. En particulier, le plus court chemin acceptant de $N_i^Y$ sur l'entrée $1^{\wp(n)}$ existe aussi pour $N_i^{Y_n}$. Par conséquent, à l'étape $n$ de la construction, nous étions dans le troisième cas et donc $Y \cap \Sigma^{\wp(n)} = R_n \neq \emptyset$. $\bowtie$

Il ne reste qu'une option: $N_i^Y$ accepte l'entrée $1^{\wp(n)}$ et prend plus de $2^{\wp(n)-1}$ étapes pour le faire. $\boxtimes$

Mais l'assertion ci-dessus est vraie pour tout $n$ tel que $\alpha(n) = i$. Or, par définition de $\alpha$, il existe une infinité de tel $n$. Donc, toute machine de Turing non-déterministe à oracle reconnaissant $S_Y$ avec l'oracle $Y$ devra prendre un temps exponentiel sur une infinité d'entrées acceptées. $\boxtimes$

Un problème est dans **co-NP** si et seulement si son complément est dans **NP**. On voit aisément que $S_Y$ est dans $\mathbf{co\text{-}NP}^Y$. Le corollaire suivant montre que la puissance de l'ordinateur quantique va au-delà du non-déterminisme.

COROLLAIRE 3.5. *Il existe un oracle $Y$ tel que $\mathbf{EQP}^Y \not\subseteq \mathbf{NP}^Y \cup \mathbf{co\text{-}NP}^Y$.*

PREUVE. Soit $Y$ l'oracle utilisé au théorème 3.4. On a vu que $S_Y \in \mathbf{EQP}^Y$. Puisque toute machine de Turing non-déterministe reconnaissant $S_Y$ doit prendre un temps exponentiel sur une infinité d'entrées, il s'ensuit que $S_Y \notin \mathbf{NP}^Y$ et donc $\mathbf{EQP}^Y \not\subseteq \mathbf{NP}^Y$. Puisque **EQP** est fermé sous toutes les opérations booléennes, le complément de $S_Y$ est dans $\mathbf{EQP}^Y$ mais pas dans $\mathbf{co\text{-}NP}^Y$ et donc $\mathbf{EQP}^Y \not\subseteq \mathbf{co\text{-}NP}^Y$. Pour démontrer le corollaire, il suffit de trouver un ensemble dans $\mathbf{EQP}^Y$ qui n'est simultanément ni dans $\mathbf{NP}^Y$ ni dans $\mathbf{co\text{-}NP}^Y$. L'ensemble $\hat{S}_Y = S_Y \cup \{\mathbf{0}^n \mid \mathbf{1}^n \notin S_Y\}$ possède cette propriété. $\boxtimes$

**3.3. Quantique versus déterministe: plus difficile.** Bien qu'une machine de Turing classique, qu'elle soit déterministe ou non-déterministe, doive prendre une temps exponentiel pour reconnaître l'ensemble $S_Y$ lorsque l'oracle $Y$ est disponible (voir la preuve du théorème 3.4), ce n'est vrai que pour un ensemble d'entrées infini mais très peu dense. Toute entrée $x$ contenant au moins un $\mathbf{0}$ est automatiquement hors de $S_Y$ ainsi que toute chaîne $1^m$ où $m$ n'est pas dans l'image (déjà très peu peuplée) de $\wp$. Seules les rares chaînes de la forme $1^{\wp(n)}$ exigent un travail réel. Pour obtenir un résultat plus fort, on doit monter plus haut dans la hiérarchie.

THÉORÈME 3.6. *Il existe un oracle $Z$ relativement auquel $V_Z$ peut être reconnu en temps exponentiel sur l'ordinateur quantique, mais toute machine de Turing* *déterministe reconnaissant $V_Z$ doit prendre un temps* doublement *exponentiel sauf sur un nombre fini d'entrées.*

Preuve. On remarque que l'ensemble reconnu à partir de $Z$ est la version compacte $V_Z$. La construction de l'oracle s'inspire de celle utilisée par Manuel Blum dans la preuve de son théorème de compression [7]. La construction de $Z$ se fait par étapes successives. À la fin de la $n$ième étape, $Z$ sera défini pour toutes les chaînes de longueur inférieure ou égale à $n$. Un élément important cette fois est que nous devrons maintenir à jour la liste des machines qui sont *supprimées* (intuitivement, une machine supprimée est une machine qui ne reconnaît pas correctement l'ensemble $V_Z$ avec l'oracle $Z$). Une machine non supprimée est dite *active*.

Soit $M_1, M_2, \ldots$ une énumération standard des machines de Turing déterministes à oracle. Initialement $Z_3 = \emptyset$ (l'énumération débute à l'étape 3 pour des raisons techniques). Pour tout $n \geq 3$, on simule les machines $M_1, \ldots, M_n$ sur chaque entrée $\tau(m)$ (pour $1 \leq m \leq n$) pendant $2^{n-1}/n^2$ étapes en utilisant $Z_n$ comme oracle (on rappelle que $\tau$ est la bijection inverse de $\sigma$). Soit $Q_n$ l'ensemble de toutes les questions posées à l'oracle durant toutes ces simulations. Notez que $Q_n$ contient au plus $n \times n \times (2^{n-1}/n^2) = 2^{n-1}$ questions.

- Si aucune des machines actives ne s'arrête, fixer $Z_{n+1} = Z_n$ et passer à l'étape suivante.
- Sinon, soit $i$ le plus petit indice tel que $M_i$ est active et s'arrête en moins de $2^{n-1}/n^2$ étapes.
  - Si $M_i^{Z_n}$ rejette $\tau(m)$, fixer $Z_{n+1} = Z_n$, supprimer la machine $i$ et passer à l'étape suivante.
  - Sinon, soit $R_n$ un sous-ensemble de $\Sigma^n$ de taille $2^{n-1}$ tel que $Q_n \cap R_n = \emptyset$. Fixer $Z_{n+1} = Z_n \cup R_n$, supprimer la machine $i$ et passer à l'étape suivante.

Soit $Z = \bigcup_{n \geq 3} Z_n$. Comme dans les preuves précédentes, il est clair que $B(Z)$ est vrai. Par conséquent, un ordinateur quantique utilisant l'oracle $Z$ peut reconnaître $S_Z$ en temps linéaire et donc $V_Z$ en temps exponentiel.

Soit $M_i$ une machine de Turing déterministe à oracle reconnaissant $V_Z$ avec l'oracle $Z$.

Assertion 3.2. *$M_i$ ne peut être supprimée lors de la construction ci-dessus*

Preuve. Preuve par contradiction: supposons que $M_i$ est supprimée à l'étape $n$. La construction de $Z$ est telle que $M_i$ en utilisant l'oracle $Z_n$ rejette l'entrée $\tau(n)$ si et seulement si $Z$ ne contient aucune chaîne de longueur $n$ et donc que $\tau(n) \in V_Z$. Par conséquemment, $M_i$ ne reconnaît pas $V_Z$ lorsqu'elle utilise l'oracle $Z_n$. Or, puisque toutes les questions posées par $M_i$ sur l'entrée $\tau(n)$ avec l'oracle $Z_n$ se retrouve dans $Z$, $M_i$ ne reconnaît pas $V_Z$ même en utilisant l'oracle $Z$ (l'énumération débute à 3 pour s'assurer que $2^{m-1}/m^2 \geq 2^{n-1}/n^2$ pour tout $m \geq n \geq 3$). Contradiction. ⊠

Soit $n_0$ suffisamment grand pour que toutes les machines d'indice inférieur à $n_0$ qui auront à être supprimée l'aient été une fois la construction parvenue à l'étape $n_0$. Considérons un $n > \max\{i, n_0\}$. L'assertion 3.2 dit que $M_i$ ne peut être supprimée à la $n^{\text{ième}}$ étape. Puisqu'aucune machine d'indice inférieur à $i$ ne peut être supprimée à cette étape, c'est donc que $M_i$ prend plus de $2^{n-1}/n^2$ étapes sur l'entrée $\tau(n)$ en utilisant l'oracle $Z_n$. Mais par un argument similaire á celui pour la preuve de l'assertion, $M_i$ a le même comportement en utilisant l'oracle $Z$. Or, la longueur de $\tau(n)$ est logarithmique en $n$ donc $2^{n-1}/n^2$ est doublement exponentiel en $\tau(n)$. Puisque $\tau$ est une bijection sur $\Sigma^*$, on conclut que si $M_i^Z$ reconnaît $V_Z$, elle doit prendre un temps double exponentiel dans la longueur de $n$ pour tout $n$ suffisamment grand. ⊠

Une remarque intéressante, due à Lev Levitin, est qu'une simulation avec parallélisme classique d'un tel calcul quantique requérait une quantité doublement exponentielle de processeurs puisque un nombre exponentiel de processeurs tournant pendant un temps exponentiel ne peuvent accomplir qu'une quantité exponentielle de travail.

**3.4. Quantique versus probabiliste.** Les sections précédentes montrent comment l'ordinateur quantique limité à un temps polynomial arrive à supplanter une machine de Turing (déterministe ou non-déterministe) dans un contexte relativisé. Il est généralement admis, par contre, que l'accès à une source aléatoire permet à une machine de Turing classique d'accomplir beaucoup plus de tâches dans le même temps. Avant de formaliser ce fait, introduisons d'abord quelques classes de complexité probabiliste classiques [13].

Les définitions suivantes sont tirées de [8]. Soit un langage $L$ et une machine de Turing probabiliste $M$. On dit que $M$ *décide* $L$ si pour tout $x$, $M(x)$ accepte si et seulement si $x \in L$. Mais puisque que $M$ est probabiliste, le temps de calcul requis pour une entrée donnée ainsi que son acceptation ou son rejet ne sont pas fixes; ils dépendent des choix probabilistes fait par $M$ lors du calcul. Considérons les machines de Turing probabilistes qui terminent dans un temps polynomial *espéré*. On définit quatre classes de machines (les probabilités sont prises sur les choix de $M$):

**PP:** La réponse donnée par $M(x)$ est correcte avec probabilité strictement plus grande que $1/2$. En particulier, cette probabilité peut être exponentiellement proche de $1/2$, ce qui est à peine meilleur qu'un simple choix à pile ou face!

**BPP:** (*Bounded error probability poly-time*) Il existe un $\varepsilon \in (0, 1/2]$, indépendant de $x$, tel que la réponse donnée par $M(x)$ est correcte avec probabilité au moins $1/2 + \varepsilon$. De tels algorithmes sont souvent appelés *Monte Carlo*. On peut diminuer la probabilité d'erreur arbitrairement près de 0 en itérant un nombre polynomial de fois $M$ sur $x$ et en prenant la réponse majoritaire (Cette propriété des algorithmes **BPP** les rend tout aussi concluants en pratique que les algorithmes de **P**.)

**RP:** Si $x \notin L$, alors avec certitude, $M(x)$ rejette. Par contre, si $x \in L$, $M(x)$ accepte avec probabilité plus grande ou égale à $1/2$. La classe **co-RP** est l'ensemble des problèmes dont le complément est dans **RP**.

**ZPP:** (*zero error probability in poly-time*) Ici, on doit introduire un troisième état final pour les machines de Turing probabilistes. Nous avions déjà "accepte" et "rejette", maintenant nous incluons "je ne sais pas". Un algorithme de type **ZPP** ne donne jamais une réponse incorrecte, mais peut parfois répondre "je ne sais pas". Autrement dit: si $x \in L$, $M(x)$ accepte avec probabilité plus grande ou égale à $1/2$ et avec probabilité complémentaire, $M(x)$ répond "je ne sais pas". Si $x \notin L$ alors $M(x)$ rejette avec probabilité plus grande ou égale à $1/2$ et avec probabilité complémentaire $M(x)$ répond "je ne sais pas". De tels algorithmes sont souvent appelés *Las Vegas*.

On peut voir que $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$ et que $\mathbf{RP} \subseteq \mathbf{NP}$ mais aucune preuve n'existe que ces inclusions sont strictes. Ces inclusions sont également valables dans un contexte relativisé. On voit aussi que $\mathbf{ZPP} = \mathbf{RP} \cap$ **co-RP** puisqu'un appel à un algorithme de **RP** suivi d'un autre (sur la même entrée) à un algorithme de **co-RP** permet soit de donner la bonne réponse avec probabilité $\geq 1/2$, soit de répondre "je ne sais pas".

De manière similaire, les analogues quantiques **ZQP**, **RQP**, **BQP** et **PQP** de ces classes peuvent être définis. Tout comme **BPP** dans le cas classique, **BQP** est sans doute la classe de problèmes la plus importante pour le modèle quantique.

Théorème 3.7. *Il existe un oracle $Y$ relativement auquel $\hat{S}_Y$ peut être reconnu en temps linéaire pire cas sur l'ordinateur quantique, mais aucune machine de Turing polynomiale de type* **ZPP** *ou* **RP** *ne peut reconnaître ni $\hat{S}_Y$ ni son complément. Donc*

$$\begin{aligned} \mathbf{EQP}^Y &\not\subseteq \mathbf{ZPP}^Y \\ \mathbf{EQP}^Y &\not\subseteq \mathbf{RP}^Y \cup \textbf{co-RP}^Y \end{aligned}$$

Preuve. Directe à partir du corollaire 3.5 puisque $\mathbf{ZPP}^Y \subseteq \mathbf{RP}^Y \subseteq \mathbf{NP}^Y$. ⊠

## 4. Discussion

**BPP** étant la classe qui caractérise le mieux les algorithmes utiles en pratique, il serait intéressant de trouver une séparation entre celle-ci et **EQP** (ou **BQP**). Malheureusement, les ensembles construits aux sections précédentes sont tous reconnus par un algorithme de **BPP** relativisé. Pour cette raison, les théorèmes 3.1, 3.4, 3.6 et 3.7 s'en trouvent amoindris.

La première percée au-delà de **BPP** fut accomplie par Ethan Berstein et Umesh Vazirani [3] en prouvant l'existence d'un oracle $W$ tel que $\mathbf{EQP}^W \not\subseteq \mathbf{BPP}^W$. Peu de temps après, Daniel Simon [19] montre l'existence d'une différence *exponentielle* entre l'ordinateur quantique et les algorithmes classiques de type Monte Carlo. Toutefois, on sort maintenant de **EQP** pour travailler avec les algorithmes quantiques terminant en temps polynomial *espéré*, soit **ZQP**. Le résultat de Simon est donc qu'il existe un oracle relativement auquel il existe un ensemble reconnu en temps polynomial espéré avec probabilité d'erreur nulle sur l'ordinateur quantique, mais toute machine de Turing probabiliste reconnaissant le même ensemble avec une probabilité d'erreur bornée devra prendre un temps exponentiel espéré sur une infinité d'entrées.

Évidemment, toutes ces séparations relativisées ne prouvent pas *réellement* que le modèle quantique est plus puissant que le modèle de Turing lorsque le temps de calcul est limité. Au mieux, ils donnent des raisons de le croire, sans plus. Mais une preuve de $\mathbf{P} \subsetneq \mathbf{EQP}$ impliquerait une preuve que $\mathbf{P} \subsetneq \mathbf{PSPACE}$ et cette question demeure toujours sans réponse malgré les nombreux efforts de la communauté scientifique. Toutefois, le modèle quantique possède au moins l'avantage d'offrir une voie inexplorée pour résoudre $\mathbf{P} \subset \mathbf{PSPACE}$. Si cette séparation risque de demeurer non prouvée pour encore quelque temps, un récent résultat en algorithmique quantique permet de supposer fortement que le modèle quantique possède un réel avantage. Peter Shor a prouvé [18] qu'il est possible de factoriser en temps polynomial quantique, tâche toujours impossible classiquement même après plus de 400 ans d'effort!

# CHAPITRE 4

## Stabilisation des calculs quantiques

Le chapitre précédent nous donne de bonnes raisons de croire que le modèle quantique proposé par Deutsch possède un avantage sur les machines de Turing du point de vue de la complexité du calcul. Cependant, même dans le cas idéal où ces arguments constitueraient une preuve formelle de l'avantage quantique, le résultat n'aurait que peu d'intérêt. On retrouve dans la littérature plusieurs modèles de calcul plus puissant qu'une machine de Turing pour lesquels il existe des preuves complètes démontrant leur supériorité sur les machines de Turing. Là où le modèle quantique se démarque des autres modèles, c'est que celui-ci est *physiquement réalisable*: rien dans les principes de la mécanique quantique n'interdit la construction d'un ordinateur quantique. Il ne saurait en être de même pour une machine de Turing à oracle où celui-ci est arbitrairement complexe par exemple. Cette combinaison d'être à la fois puissants (du moins, en apparence) et réalisable rend l'étude du modèle quantique particulièrement intéressante.

### 1. Nécessité de la stabilisation

Deustch a déjà discuté dans [9] et [10] des détails théoriques relatifs à la réalisation d'un ordinateur quantique. Malheureusement, la technologie nécessaire pour effectivement construire une telle machine est hors de notre portée pour l'instant. Un des obstacles à surmonter (et non le moindre) est de maintenir la cohérence de l'ordinateur quantique suffisamment longtemps pour que l'algorithme ait le temps de s'exécuter. Le problème est le suivant: tout système quantique possède un ensemble d'états (dits *propres*) dans lesquels le système est parfaitement stable; dans tout autre état, le système a tendance à se dissiper. Cette règle s'applique en particulier à l'ordinateur quantique. Les états propres de celui-ci sont les configurations de $\Gamma = \{\gamma_1, \ldots, \gamma_w\}$ (voir 2.1, page 5) et, en général, l'état $|\varphi\rangle$ d'un ordinateur quantique est une superposition quelconque de ces configurations, et donc instable. Cette instabilité inhérente introduira inévitablement des erreurs lors du calcul.

Dans le contexte d'un ordinateur classique, des erreurs dues à une forme d'instabilité des circuits existent aussi, mais plusieurs techniques sont disponibles pour garder le taux d'erreur à un niveau négligeable (la plus simple consistant à tripler tous les systèmes et procéder par vote majoritaire). Pour des raisons qui seront expliquées plus loin, ces techniques ne sont pas utilisables dans un contexte quantique. Le but du présent chapitre est d'analyser la première technique de correction d'erreurs efficace sur l'ordinateur quantique (proposée par Deutsch) et d'analyser l'efficacité de celle-ci sur un type d'erreur très précis.

## 2. Définition du modèle d'erreur

Soit $A$ une matrice unitaire et soit $|\varphi_0\rangle$ l'état représentant la configuration initiale d'une machine de Turing quantique $M$ sur entrée $x$. On a vu au chapitre 2 que le calcul de $M(x)$ est en fait une suite de vecteurs d'état tels que

$$\forall\, 1 \leq i \leq m: \quad |\varphi_i\rangle = A|\varphi_{i-1}\rangle.$$

Ceci du moins dans un contexte idéal. Considérons maintenant le cas où l'ordinateur quantique est légèrement instable: entre deux applications de $A$, l'état de l'ordinateur aura dévié quelque peu. Dans ce cas, la suite des états représentant le calcul de $M(x)$ a la forme:

$$\forall\, 1 \leq i \leq m: \quad \begin{array}{rl} \text{temps mort:} & |\varphi_{i-1}\rangle \rightsquigarrow |\varphi'_{i-1}\rangle \\ \text{opération:} & |\varphi_i\rangle = A|\varphi'_{i-1}\rangle \end{array}$$

Bien sûr, le processus engendrant la déviation n'est pas réellement discret en fonction du temps. Ce que ce modèle d'erreur indique, c'est que dans le 'temps mort' entre deux opérations $A$, l'état $|\varphi_{i-1}\rangle$ sera perturbé légèrement en $|\varphi'_{i-1}\rangle$

et donc lorsque que $A$ est appliquée, ce sera sur ce dernier. L'erreur réfère à la déviation $|\varphi_{i-1}\rangle \rightsquigarrow |\varphi'_{i-1}\rangle$. Nous supposons que le processus engendrant ce type d'erreur possède les propriétés suivantes:

(i) **Purement Aléatoire:** l'action du processus ne dépend pas des états précédents ou courants de l'ordinateur. En particulier, les erreurs systématiques ne sont pas considérées.
(ii) **Continu:** la déviation $|\varphi_{i-1}\rangle \rightsquigarrow |\varphi'_{i-1}\rangle$ se fait de manière continue dans le temps et non pas par sauts instantanés.
(iii) **Indépendant du temps:** La déviation $|\varphi_{i-1}\rangle \rightsquigarrow |\varphi'_{i-1}\rangle$ se fait par le chemin le plus direct.

Nous ferons également l'hypothèse que le processus d'erreur soit tel que

$$\forall\, 0 \leq i \leq m: \ \ \|\langle\varphi_i|\varphi'_i\rangle\|^2 = 1 - \varepsilon^2$$

Nous reviendrons sur la propriété (i) à la section 9. L'idée derrière les propriétés (ii) et (iii) est qu'entre deux opérations, la direction de la déviation est constante mais lorsque qu'une opération est appliquée, ceci perturbe suffisamment le système pour que la direction de déviation de l'étape suivante soit non corrélée avec la précédente. On suppose que tout ce processus se fait de manière unitaire.

Demander que le processus d'erreur possède ces propriétés limite énormément la généralité du modèle d'erreur. Malgré tout, l'étude de ce modèle d'erreur simpliste nous permettra d'avoir une bonne idée des problèmes reliés à la stabilisation des calculs quantiques.

## 3. Définition du problème

Il est intéressant de noter que le modèle d'erreur utilisé ici est invariant par rapport à l'algorithme (soit la matrice unitaire $A$) et aussi invariant par rapport à la configuration initiale. Sans perte de généralité, nous supposerons donc que $A$ est en fait la matrice identité. Pour simplifier, nous limiterons aussi les ordinateurs quantiques à n'utiliser qu'un seul bit de mémoire. (À la section 9, nous discuterons du cas où cette contrainte sera levée.) Toujours sans perte de généralité, la configuration initiale sera $|\mathbf{0}\rangle$.

Considérant toutes ces propriétés et ces contraintes, la situation est la suivante: soit $M$ un ordinateur quantique à un seul bit, initialement dans la configuration $|\mathbf{0}\rangle$, calculant la fonction identité pendant $m$ étapes. Ce calcul subit malheureusement de légères déviations déterminées par le paramètre $\varepsilon$ selon le modèle d'erreur ci-haut. Lorsque les $m$ étapes sont terminées, on observe $M$ avec l'observable $\mathcal{O} = \{|\mathbf{0}\rangle, |\mathbf{1}\rangle\}$. Le problème est, d'une part, de déterminer quelle est la probabilité espérée d'obtenir la bonne réponse (soit $|\mathbf{0}\rangle$) après ces $m$ étapes et, d'autre part, de trouver des méthodes pour maximiser cette probabilité.

## 4. Premier cas: aucune correction

Soit $M$ un ordinateur quantique défini tel qu'indiqué à la section 3. Soit $|\varphi_0\rangle, \ldots, |\varphi_m\rangle$ la suite des vecteurs d'état lors du calcul de $M(\mathbf{0})$ (en particulier, $|\varphi_0\rangle = |\mathbf{0}\rangle$). On a que

$$\forall\, 1 \leq i \leq m: \quad \begin{array}{rl} \text{temps mort:} & |\varphi_{i-1}\rangle \rightsquigarrow |\varphi'_{i-1}\rangle \\ \text{opération:} & |\varphi_i\rangle = I|\varphi'_{i-1}\rangle = |\varphi'_{i-1}\rangle \end{array}$$

avec la relation

$$\forall\, 1 \leq i \leq m: \ \ \|\langle\varphi_{i-1}|\varphi_i\rangle\|^2 = 1 - \varepsilon^2$$

Soit $X$ la variable aléatoire du résultat de l'observation de $M$ avec l'observable $\mathcal{O}$ après $m$ étapes. La probabilité que $X = |\mathbf{0}\rangle$ est $\|\langle\mathbf{0}|\varphi_m\rangle\|^2$. Cette probabilité exprime le fait qu'en général, le vecteur final $|\varphi_m\rangle$ forme un certain angle $\theta$ par rapport au vecteur initial $|\mathbf{0}\rangle$ et le résultat de l'observation est probabiliste en fonction de cet angle. Mais cet angle $\theta$ est lui-même aléatoire: deux vecteurs finaux $|\varphi_m^1\rangle$ et $|\varphi_m^2\rangle$ résultant de deux exécutions de $M$ seront généralement différents. On doit donc calculer l'espérance de la probabilité que $X = |\mathbf{0}\rangle$. Soit $\Theta$ la variable aléatoire de l'angle entre $|\mathbf{0}\rangle$ et $|\varphi_m\rangle$. Considérons le cercle unité (dans les complexes) et considérons les points sur ce cercle déterminés par les vecteurs $|\varphi_0\rangle, \ldots, |\varphi_m\rangle$. Cette suite de points n'est rien d'autre qu'une marche aléatoire sur le cercles unité complexe centrée au point correspondant à $|\mathbf{0}\rangle$ ([16]). Si $m$ est suffisamment grand (tout en maintenant $\varepsilon m$ petit), on peut approximer la fonction de densité $f_\Theta$ de $\Theta$ par ($f_\Theta$ étant symétrique sur le cercle unité complexe, nous l'exprimons en fonction de l'angle $y$ par rapport à la position $|\mathbf{0}\rangle$)

$$f_\Theta(x) = \frac{2}{m}\,\boldsymbol{e}^{-\frac{y^2}{m}}$$

On peut exprimer la probabilité que $X = |\mathbf{0}\rangle$ en fonction de la variable aléatoire $\Theta$ puisque $\|\langle\mathbf{0}|\varphi_m\rangle\|^2 = \cos^2\Theta$. L'espérance de cette probabilité est

$$\begin{aligned}\operatorname{Esp}\left[\cos^2\Theta\right] &= \int_0^\infty \cos^2(\varepsilon m) f_\Theta(y)\,dy \\ &\approx \cos^2(\varepsilon\sqrt{m}) \\ &\approx 1 - \varepsilon^2 m \qquad \text{si } \varepsilon\sqrt{m} \ll 1\end{aligned}$$

La réponse au premier volet de la question posée à la section 3 est que si rien n'est fait pour compenser pour les erreurs, la probabilité d'obtenir une réponse correcte est $\approx 1 - \varepsilon^2 m$. Dans les prochaines sections, nous considérons deux méthodes pour améliorer cette probabilité de succès.

## 5. Second cas: vote majoritaire

La méthode la plus simple pour hausser la fiabilité d'une réponse est de procéder par vote majoritaire: on répète le calcul un certain nombre de fois et le résultat majoritaire est jugé comme étant la bonne réponse.

Soit $R$ ordinateurs quantiques $M_1, \ldots, M_R$ différents et indépendants tels que décrits à la section 3 (on suppose $R$ impair). Si ces ordinateurs sont observés après leurs $m$ étapes de calcul, chacun donnera une réponse correcte avec probabilité[1] $1 - \varepsilon^2 m$ (voir section 4). Soit $X$ la variable aléatoire correspondant au résultat majoritaire. La probabilité qu'un vote majoritaire donne la bonne réponse est

$$\begin{aligned}\operatorname{Prob}[X = |\mathbf{0}\rangle] &= 1 - \operatorname{Prob}[X = |\mathbf{1}\rangle] \\ &= 1 - \sum_{i=\lceil R/2\rceil}^{R} \mathrm{C}_i^R (1-\varepsilon^2 m)^{R-i}(\varepsilon^2 m)^i \\ &\approx 1 - \mathrm{C}_{\lceil R/2\rceil}^R (1-\varepsilon^2 m)^{\lceil R/2\rceil}(\varepsilon^2 m)^{\lceil R/2\rceil} \\ &\approx 1 - \mathrm{C}_{\lceil R/2\rceil}^R (\varepsilon^2 m)^{\lceil R/2\rceil}\end{aligned}$$

[1] Pour la suite du texte, à moins d'avis contraire, le mot *probabilité* signifiera implicitement *probabilité espérée*.

Ceci représente une nette amélioration par rapport à la situation de la section 4. Malheureusement, cette méthode de correction d'erreur n'est pas entièrement satisfaisante puisqu'elle n'est applicable que si le résultat final du calcul de $M_i(\mathbf{0})$ est garanti de ne pas être en superposition (dans le cas où aucune erreur n'est générée). En général, l'algorithme utilisé par les machines $M_1, \ldots, M_R$ est tel que pour toute entrée $x$, l'état final $M_i(x)$ est une superposition $|\varphi_i\rangle$ de la forme

$$|\varphi_i\rangle = \alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle$$

où $\alpha, \beta \in \mathbb{C}$ et $\|\alpha\|^2 + \|\beta\|^2 = 1$. Même dans le cas où il n'y a aucune erreur, une observation avec $\mathcal{O}$ détruirait cette superposition. En particulier, si $M_i$ servait de sous-routine à une autre machine quantique, le vote majoritaire détruirait la superposition. Si l'état $|\varphi_i\rangle$ est connu d'avance, une observation non-destructive peut être faite, mais pourquoi perdre notre temps à calculer $|\varphi_i\rangle$ dans ce cas? Ces considérations expliquent aussi pourquoi la méthode du vote majoritaire ne peut pas servir *à l'intérieur* d'un calcul. La prochaine section présente une méthode de correction d'erreur à la fois efficace et utilisable même durant un calcul.

## 6. Troisième cas: symétrisation

Le problème principal de la méthode du vote majoritaire est qu'elle force une observation potentiellement destructive de l'ordinateur quantique. On sait qu'une observation non destructive d'une superposition $|\varphi\rangle$ est possible à condition de bien connaître cet état, mais $|\varphi\rangle$ est justement ce que nous voulons apprendre. La situation semble sans issue.

Pas entièrement! Soit $R$ machines quantiques $M_1, \ldots, M_R$ et une entrée $x$. Ignorons pour l'instant les erreurs. En général, l'usager ne possède aucune information sur l'état final du calcul de $M_i(x)$, pour tout $i$. Cependant, quelque soit cet état final pour l'une de ces machines, *toutes les autres doivent être dans le même état*. Cette constatation se trouve à la base de la méthode de symétrisation présentée ici. Toutefois, avant d'en décrire le fonctionnement, nous devons introduire certains concepts clés.

**6.1. L'effet watch-dog.** L'effet watch-dog peut s'exprimer ainsi: plus un état quantique est observé, moins il a tendance à changer. Considérons, par exemple, le cas d'un système à deux états $|\mathsf{Yin}\rangle$ et $|\mathsf{Yang}\rangle$. Au temps $t_0$, l'état $|\varphi_0\rangle$ du système est $|\mathsf{Yin}\rangle$. À mesure que le temps passe, $|\varphi_0\rangle$ subit une 'rotation' constante l'éloignant de son état initial. Au temps $t_1$, le système est dans l'état $|\varphi_1\rangle$ tel que l'angle entre ce dernier et l'état initial est $\theta$. Formellement:

$$\begin{array}{ll} \text{au temps } t_0\text{:} & |\varphi_0\rangle = |\mathsf{Yin}\rangle \\ \text{au temps } t_1\text{:} & |\varphi_1\rangle = \boldsymbol{e}^{-\boldsymbol{i}\alpha} \cos\theta |\mathsf{Yin}\rangle + \boldsymbol{e}^{-\boldsymbol{i}\beta} \sin\theta |\mathsf{Yang}\rangle \end{array}$$

(les facteurs de phases $\boldsymbol{e}^{-\boldsymbol{i}\alpha}$ et $\boldsymbol{e}^{-\boldsymbol{i}\beta}$, où $\alpha, \beta \in [0..2\pi]$, apparaissant dans l'équation ci-dessus sont nécessaires. On laisse le lecteur vérifier que tout vecteur normé d'un espace vectoriel complexe de dimension 2 peut s'exprimer à l'aide des paramètres $\alpha, \beta$ et $\theta$.)

Soit $X$ la variable aléatoire représentant le résultat d'une observation de $|\varphi_1\rangle$ avec l'observable $\mathcal{Y} = \{|\mathsf{Yin}\rangle, |\mathsf{Yang}\rangle\}$. On a que

$$\text{Prob}\,[X = |\mathsf{Yin}\rangle] = \|\langle \mathsf{Yin}|\varphi_1\rangle\|^2 = \cos^2\theta$$

Supposons maintenant que nous observions le système à $n$ intervalles réguliers entre $t_0$ et $t_1$. Au temps $t_{1/n}$, l'angle de déviation sera seulement de $\theta/n$. Après observation, si $X = |\mathsf{Yin}\rangle$ alors l'état retourne à $|\mathsf{Yin}\rangle$ (voir section 1.5 du chapitre 2) et la déviation reprendra à partir de $|\mathsf{Yin}\rangle$ encore. Soit $Y$ la variable aléatoire représentant le résultat final après ces $n$ observations. On a que

$$\begin{aligned} \text{Prob}\,[Y = |\mathsf{Yin}\rangle] &= (\text{Prob}\,[X = |\mathsf{Yin}\rangle])^n \\ &= (\cos^2(\theta/n))^n \\ &\approx (1 - \theta^2/n^2)^n \qquad \text{pour } n \text{ suffisamment grand} \\ &\approx 1 - \theta^2/n \end{aligned}$$

et cette probabilité tend vers 1 lorsque $n$ tend vers l'infini.

Ces conclusions tiennent aussi pour un système de dimension supérieure. Avec l'effet watch-dog, il est possible d'augmenter la probabilité de succès d'une observation *à condition de connaître l'état dans lequel le système doit être*. On ne peut donc pas limiter la taille des erreurs en observant fréquemment un ordinateur quantique puisque l'on ne connaît justement pas son état. Pour accomplir cette tâche, nous avons besoin d'un second outil: l'espace symétrique.

**6.2. L'espace symétrique.** Soient $H_i$, $1 \leq i \leq R$, des espaces vectoriels complexes de dimension 2. Les vecteurs $|\mathbf{0}\rangle$ et $|\mathbf{1}\rangle$ forment la base canonique de chacun des $H_i$. Les vecteurs des espaces $H_j$ et $H_k$ ($j \neq k$) ont respectivement la forme suivante:

$$\begin{aligned} |\varphi_j\rangle &= \alpha_j|\mathbf{0}\rangle + \beta_j|\mathbf{1}\rangle \\ |\varphi_k\rangle &= \alpha_k|\mathbf{0}\rangle + \beta_k|\mathbf{1}\rangle \end{aligned}$$

Le *produit tensoriel* de $H_j$ et $H_k$, noté $H_j \otimes H_k$, est l'espace vectoriel complexe de dimension $2^2 = 4$ engendré par l'ensemble des vecteurs de la forme

$$|\varphi_j\rangle \otimes |\varphi_k\rangle = \alpha_j\alpha_k|\mathbf{00}\rangle + \alpha_j\beta_k|\mathbf{01}\rangle + \beta_j\alpha_k|\mathbf{10}\rangle + \beta_j\beta_k|\mathbf{11}\rangle \tag{4.1}$$

On remarque que ce produit n'est pas commutatif: le produit tensoriel de $H_k$ et $H_j$ est composé des vecteurs de la forme

$$|\varphi_k\rangle \otimes |\varphi_j\rangle = \alpha_k\alpha_j|\mathbf{00}\rangle + \alpha_k\beta_j|\mathbf{01}\rangle + \beta_k\alpha_j|\mathbf{10}\rangle + \beta_k\beta_j|\mathbf{11}\rangle$$

et si $\alpha_j\alpha_k = \alpha_k\alpha_j$ et $\beta_j\beta_k = \beta_k\beta_j$, il n'est pas vrai en général que les produits croisés $\alpha_j\beta_k$ et $\beta_j\alpha_k$ soient égaux à leur correspondants $\alpha_k\beta_j$ et $\beta_k\alpha_j$. Considérons le sous-espace $S$ de $H_j \otimes H_k$ engendré par les trois vecteurs normés suivant:

$$\begin{aligned} |E_0\rangle &= |\mathbf{00}\rangle \\ |E_1\rangle &= \frac{1}{\sqrt{2}}(|\mathbf{01}\rangle + |\mathbf{10}\rangle) \\ |E_2\rangle &= |\mathbf{11}\rangle \end{aligned}$$

Un vecteur $|\varphi\rangle$ de $S$ a la forme

$$\begin{aligned} |\varphi\rangle &= \nu_0|E_0\rangle + \nu_1|E_1\rangle + \nu_2|E_2\rangle \\ &= \nu_0|\mathbf{00}\rangle + \frac{\nu_1}{\sqrt{2}}|\mathbf{01}\rangle + \frac{\nu_1}{\sqrt{2}}|\mathbf{10}\rangle + \nu_2|\mathbf{11}\rangle \end{aligned}$$

On voit que $S$ est composé exactement des vecteurs de $H_j \otimes H_k$ dont les produits croisés sont égaux (voir équation 4.1). Ceci implique que le sous-espace $S$ est invariant sous la permutation des indices $j$ et $k$ dans $|\varphi_j\rangle \otimes |\varphi_k\rangle$. Nous en tirons la définition suivante:

DÉFINITION 6.1. *Soit $R$ espaces vectoriels $H_1, \ldots, H_R$ de même dimension. Soit $\mathcal{H} = \bigotimes_{i=1}^{R} H_i$. L'espace symétrique de $\mathcal{H}$, noté $\mathrm{Sym}\,(\mathcal{H})$, est le sous-espace engendré par l'ensemble des vecteurs invariants sous toute permutation de deux indices dans $\bigotimes_{i=1}^{R} H_i$.*

Le seul produit tensoriel qui sera nécessaire dans ce chapitre est $\mathcal{H} = \bigotimes_{i=1}^{R} H_i$ où chacun des $H_i$ est de dimension 2 ayant comme base les vecteurs $|\mathbf{0}\rangle$ et $|\mathbf{1}\rangle$. Il sera parfois nécessaire de référer à un vecteur de base spécifique de $\mathcal{H}$ ou de $\mathrm{Sym}\,(\mathcal{H})$. Pour ce faire, nous utiliserons la convention suivante. Soit un ordonnancement des chaînes de $R$ bits tel que pour $i$ de 0 à $R$ et $j$ de 1 à $\mathrm{C}_i^R$, $e_{ij}$ est la $j$ième chaîne contenant $i$ '**1**' et $R-i$ '**0**'. On notera $e_{ij}^k$ le $k$ième bit de la chaîne $e_{ij}$. L'ensemble des vecteurs $|e_{ij}\rangle$ forment ainsi la base canonique de $\mathcal{H}$. L'ensemble des $R+1$ vecteurs

$$|E_i\rangle = \frac{1}{\sqrt{\mathrm{C}_i^R}} \sum_{j=1}^{\mathrm{C}_i^R} |e_{ij}\rangle \qquad \text{pour } i \text{ de } 0 \text{ à } R$$

forme la base canonique (normée) de $\mathrm{Sym}\,(\mathcal{H})$. Comme exemple, si $R = 3$ on a que les vecteurs

$$\begin{aligned}
|E_0\rangle &= |\mathbf{000}\rangle \\
|E_1\rangle &= \frac{1}{\sqrt{3}}(|\mathbf{100}\rangle + |\mathbf{010}\rangle + |\mathbf{001}\rangle) \\
|E_2\rangle &= \frac{1}{\sqrt{3}}(|\mathbf{110}\rangle + |\mathbf{101}\rangle + |\mathbf{011}\rangle) \\
|E_3\rangle &= |\mathbf{111}\rangle
\end{aligned}$$

forment la base canonique de $\mathrm{Sym}\left(\bigotimes_{i=1}^{3} H_i\right)$. Le lecteur peut vérifier que chacun des $|E_0\rangle, \ldots, |E_3\rangle$ est invariant sous n'importe quelle permutation de deux indices.

**6.3. Idée générale de la méthode.** David Deutsch est le père de la méthode de correction décrite ici. La contribution apportée par la présente thèse consiste en l'analyse formelle de son efficacité.

Dans l'analyse du problème de la correction d'erreur s'accumulant lors d'un calcul quantique, nous avions déterminé qu'il est possible de garantir l'exactitude du résultat calculé à condition de connaître ce résultat. Alors que cette condition est irréalisable avec un seul ordinateur quantique, il n'en va pas de même pour un groupe d'ordinateurs calculant tous sur la même entrée. L'argument est le suivant: Soit $R$ ordinateurs quantiques $M_1, \ldots, M_R$, tous calculant sur la même entrée $x$. Soit $H_i$ l'espace de calcul de $M_i$. Supposons un instant qu'aucune erreur ne vienne perturber ces calculs. À la fin du calcul, tous les $M_i$ sont dans le même état $|\varphi\rangle$ et ce dernier est en général inconnu de l'usager. Soit $\mathcal{H} = \bigotimes_{i=1}^{R} H_i$ et considérons $\mathrm{Sym}\,(\mathcal{H})$. Soit $|\psi\rangle = |\varphi\rangle^R$ le vecteur dans $\mathcal{H}$ représentant l'état combiné des $R$ ordinateurs. Quelque soit $|\varphi\rangle$, il est clair que $|\psi\rangle$ se trouvera dans l'espace $\mathrm{Sym}\,(\mathcal{H})$.

Considérons maintenant le cas où les ordinateurs subissent certaines déviations. À la fin du calcul, les états de ces ordinateurs ne sont plus identiques: chaque $M_i$ se trouve dans l'état $|\varphi_i\rangle$ probablement différent de $|\varphi\rangle$ et ces $|\varphi_i\rangle$ sont tous aussi inconnus de l'usager. Considérons l'état combiné $|\psi\rangle$ des $R$ ordinateurs. On a que

$$|\psi\rangle = \bigotimes_{i=1}^{R} |\varphi_i\rangle$$

Si les déviations que subissent les ordinateurs sont faibles, alors quels que soient les $|\varphi_i\rangle$, le vecteur $|\psi\rangle$ est certainement peu éloigné de son état correct $|\varphi\rangle^R$. En terme des vecteurs d'état, ceci s'exprime comme

$$|\psi\rangle = \alpha|\phi\rangle^R + \beta|\eta\rangle$$

où le vecteur (normé) $|\eta\rangle$ est perpendiculaire à $|\varphi\rangle^R$ et la norme de $\beta$ est relativement proche de 0. Soit $\mathcal{O}_S$ l'observable partitionnant $\mathcal{H}$ en deux sous-espaces $S = \mathrm{Sym}\,(\mathcal{H})$ et $S^\perp = \mathrm{Sym}\,(\mathcal{H})^\perp$. Pour déterminer le résultat d'une observation de l'état combiné $|\psi\rangle$ avec $\mathcal{O}_S$, on doit calculer les projections de $|\psi\rangle$ dans $S$ et $S^\perp$. On sait déjà que $|\varphi\rangle^R = |s_1\rangle$ se trouve entièrement dans $S$. Il ne reste qu'à décomposer le vecteur $|\eta\rangle$. Puisque les erreurs provenant de chacun des ordinateurs ne sont pas corrélées entre elles, il est fort probable que $|\eta\rangle$ ne soit pas entièrement dans l'espace $S$, sans lui être perpendiculaire. Vectoriellement

$$|\eta\rangle = \nu_1|s_2\rangle + \nu_2|s^\perp\rangle$$

où $|s_2\rangle$ et $|s^\perp\rangle$ sont des vecteurs normés de $S$ et $S^\perp$ respectivement et $\|\nu_1\|^2 + \|\nu_2\|^2 = 1$. Nous pouvons donc exprimer l'état combiné des $R$ ordinateurs comme

$$|\psi\rangle = \alpha|s_1\rangle + \beta\nu_1|s_2\rangle + \beta\nu_2|s^\perp\rangle$$

Donc, avec probabilité $1 - \|\beta\nu_2\|^2$, l'état résultant $|\psi'\rangle$ de l'observation de $|\psi\rangle$ par $\mathcal{O}$ sera

$$|\psi'\rangle = \frac{1}{\sqrt{\|\alpha\|^2 + \|\beta\nu_2\|^2}}(\alpha|s_1\rangle + \beta\nu_1|s_2\rangle)$$

On remarque que le vecteur $|\psi'\rangle$ est en général beaucoup plus 'fidèle' à $|\varphi\rangle^R$ que ne l'est $|\psi\rangle$. En projetant l'état combiné des ordinateurs dans l'espace symétrique, une certaine part de l'erreur accumulée fut annulée. Tout aussi important, cette correction fut faite *sans détruire les superpositions des* $M_i$. Donc, si à la fin du calcul des $M_i$ l'observation de leur état combiné est réussie (donne le résultat $S$), alors la fiabilité de chacun des $|\varphi_i\rangle$ est augmenté. Par exemple, si en l'absence d'erreur le résultat devait être un état de base (par opposition à une superposition), alors un vote majoritaire serait d'autant plus fiable.

Il ne demeure qu'un seul problème: la probabilité de succès de la projection de $|\psi'\rangle$ dans $S$ n'est que de $1 - \|\beta\nu_2\|^2$. Si cette projection échoue, l'état de chacun des $R$ ordinateurs est irrémédiablement détruit! Il n'est pas très désirable d'avoir une méthode de correction d'erreur qui a une probabilité non négligeable de détruire ce qu'elle tente de corriger. C'est ici qu'entre en jeu l'effet watchdog: plutôt que d'observer l'état combiné des ordinateurs une seul fois à la fin du calcul, ils seront constamment sous observation. En fonction de la fréquence d'observation, la probabilité que l'état combiné demeure à l'intérieur de $S$ pourra être baissée à un niveau acceptable.

Pour être parfaitement rigoureux, la question de la réalisabilité physique de l'observable de symétrisation $\mathcal{O}_S$ doit être considérée. Ceci n'effectant pas vraiment l'analyse de l'efficacité de cette méthode de correction (traitée dans les sections suivantes), nous reporterons cette discussion à l'appendice A.

**6.4. Symétrisation.** Comme nous venons de voir, la méthode de symétrisation consiste simplement à fréquemment projeter l'état combiné de ces $R$ ordinateurs dans l'espace symétrique associé. Les deux points importants à considérer sont que, d'une part, la symétrisation peut échouer (et forcer l'usager à reprendre tous les calculs depuis début) et, d'autre part, même dans le cas où la symétrisation est réussie tout au long du calcul, une certaine erreur résiduelle demeure toujours. Pour analyser l'efficacité correctrice de cette méthode, nous devons d'abord calculer la probabilité que le processus de symétrisation réussisse tout au long du calcul et ensuite déterminer la taille de l'erreur résiduelle.

Soient $R$ ordinateurs quantiques $M_1, \ldots, M_R$ tels que décris à la section 3. On a que $H_i$ est l'espace de calcul de $M_i$. Soit $\mathcal{H} = \bigotimes_{i=1}^{R} H_i$ et soit $\mathcal{O}_S$ l'observable partitionnant $\mathcal{H}$ en deux sous-espaces orthogonaux $S = \mathrm{Sym}\,(\mathcal{H})$ et $S^{\perp}$. Pour $i$ de 1 à $R$, soit $|\varphi_0^i\rangle, \ldots, |\varphi_m^i\rangle$ les vecteurs d'état de $M_i$ pour chaque étape du calcul sur l'entrée $|\mathbf{0}\rangle$. Soient $|\psi_0\rangle, \ldots, |\psi_m\rangle$ les vecteurs d'état combinés correspondants de ces étapes. La situation se décrit comme suit:

$$\begin{array}{rl} \text{si } i = 0: & |\psi_i\rangle = |\mathbf{0}\rangle^R \\ \forall\, 1 \leq i \leq R: & |\psi_i\rangle = |\varphi_{i-1}^1\rangle ... |\varphi_{i-1}^R\rangle \end{array}$$

et nous savons qu'à chaque étape, l'erreur pour chaque machine s'exprime comme

$$\forall\, 1 \leq j \leq R,\ \ \forall\, 1 < i \leq m: \ \ \left\| \langle \varphi_{i-1}^j | \varphi_i^j \rangle \right\|^2 = 1 - \varepsilon^2$$

Donc, par rapport aux vecteurs d'état combinés, l'erreur à chaque étape est

$$\begin{aligned} \forall\, 1 < i \leq m: \ \ \| \langle \psi_{i-1} | \psi_i \rangle \|^2 &= \left\| \langle \varphi_{i-1}^1 | \varphi_i^1 \rangle \ldots \langle \varphi_{i-1}^R | \varphi_i^R \rangle \right\|^2 \\ &= \left\| \langle \varphi_{i-1}^1 | \varphi_i^1 \rangle \right\|^2 \ldots \left\| \langle \varphi_{i-1}^R | \varphi_i^R \rangle \right\|^2 \\ &= (1 - \varepsilon^2)^R \\ &\approx 1 - \varepsilon^2 R \end{aligned}$$

Nous voulons donc réduire cette probabilité à un niveau négligeable en utilisant l'effet watch-dog. Le résultat de la section 6.1 ne peut pas s'appliquer directement pour deux raisons: premièrement, après chaque observation, la direction de dérivation de l'erreur de chaque ordinateur est modifiée (voir section 2) et deuxièmement, même lorsque l'observation réussit, l'état obtenu n'est pas $|\mathbf{0}\rangle^R$: il reste encore un peu de l'erreur initiale. Une analyse plus raffinée est donc nécessaire.

Considérons d'abord la première étape du calcul. Soit $n \in \mathbb{N}$. On sépare l'intervalle de temps entre les étapes 0 et 1 en $n$ sous-intervalles égaux. Soit $|\psi_{1/n}\rangle, |\psi_{2/n}\rangle, \ldots, |\psi_{n/n}\rangle$ tels que pour tout $i$ de 1 à $n$, $|\psi_{i/n}\rangle$ est l'état combiné des $R$ ordinateurs *après* avoir projeté l'état $|\psi_{\frac{i-1}{n}}\rangle$ dans l'espace symétrique. Ces étapes de symétrisation entre deux étapes de calcul sont appelées des *micro-étapes*. Pour $j$ de 1 à $R$ et $i$ de 0 à $n$, on définit les vecteurs $|\varphi^j_{i/n}\rangle$ comme l'état de l'ordinateur $M_j$ à la micro-étape $i/n$. Soit $X_{1/n}$ la variable aléatoire du résultat de l'observation de $|\psi_{1/n}\rangle$ avec l'observable $\mathcal{O}_S$. Pour calculer la probabilité que $X_{1/n} = S$, on doit calculer la norme au carré de la composante de $|\psi_{1/n}\rangle$ selon chacun des vecteurs de base de l'espace symétrique. Ce calcul est

$$
\begin{aligned}
\text{Prob}\left[X_{1/n} = S\right] &= \sum_{i=0}^{R} \left\| \langle E_i | \psi_{1/n} \rangle \right\|^2 \\
&= \sum_{i=0}^{R} \frac{1}{\mathrm{C}_i^R} \left\| \sum_{j=1}^{\mathrm{C}_i^R} \langle e_{ij} | \psi_{1/n} \rangle \right\|^2 \\
&= \sum_{i=0}^{R} \frac{1}{\mathrm{C}_i^R} \left\| \sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \langle e^k_{ij} | \varphi^k_{1/n} \rangle \right\|^2
\end{aligned}
$$

À ce point, un problème surgit: tout ce que nous savons sur $|\varphi^k_{1/n}\rangle$ est (par les propriétés données à la section 2) que $\left\| \langle \mathbf{0} | \varphi^k_{1/n} \rangle \right\|^2 = 1 - (\varepsilon/n)^2$. La forme générale d'un vecteur normé ayant cette propriété est

$$|\varphi^k_{1/n}\rangle = (1 - \frac{\varepsilon}{n})\, \boldsymbol{e}^{-\boldsymbol{i}\alpha_k}\, |\mathbf{0}\rangle + \frac{\varepsilon}{n}\, \boldsymbol{e}^{-\boldsymbol{i}\beta_k}\, |\mathbf{1}\rangle$$

et les propriétés du modèle d'erreur nous permettent d'affirmer que $\alpha_k$ et $\beta_k$ sont choisies uniformément dans l'intervalle $[0, 2\pi]$. Soit

$$\theta^k_{ij} = \begin{cases} \alpha_k & \text{si } |e^k_{ij}\rangle = |\mathbf{0}\rangle \\ \beta_k & \text{si } |e^k_{ij}\rangle = |\mathbf{1}\rangle \end{cases}$$

On a donc que

$$\begin{aligned}
\text{Prob}\left[X_{1/n} = S\right] &= \sum_{i=0}^{R} \frac{1}{\mathrm{C}_i^R} \left\| \sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \langle e_{ij}^k | \varphi_{1/n}^k \rangle \right\|^2 \\
&= \sum_{i=0}^{R} \frac{1}{\mathrm{C}_i^R} \left(1 - {\varepsilon^2}/{n^2}\right)^{R-i} \left(\frac{\varepsilon^2}{n^2}\right)^i \left\| \sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k} \right\|^2
\end{aligned}$$

Concentrons maintenant notre attention uniquement sur la norme au carré de l'équation ci-dessus. On se rappelle que nous cherchons en fait une probabilité *espérée*. Dans ce cas, ce n'est pas tant la valeur de cette norme qui nous intéresse mais sa valeur espérée. Donc

$$\begin{aligned}
\text{Esp}\left[\left\| \sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k} \right\|^2\right] &= \text{Esp}\left[\left(\sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}\right)\left(\sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{\overline{\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}}\right)\right] \\
&= \text{Esp}\left[\sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}\, \boldsymbol{e}^{\overline{\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}} + \sum_{\substack{j,l=1 \\ j\neq l}}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}\, \boldsymbol{e}^{\overline{\boldsymbol{i}\boldsymbol{\theta}_{il}^k}}\right]
\end{aligned}$$

Or, les valeurs de $\alpha_k$ et $\beta_k$ étant choisies uniformément aléatoirement dans l'intervalle $[0..2\pi]$, il en va de même pour $\theta_{ij}^k$. Dans ces conditions, l'espérance de la seconde somme est égale à zéro et puisque $\boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}\, \boldsymbol{e}^{\overline{\boldsymbol{i}\boldsymbol{\theta}_{il}^k}} = 1$, on a que

$$\begin{aligned}
\text{Esp}\left[\sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}\, \boldsymbol{e}^{\overline{\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}} + \sum_{\substack{j,l=1 \\ j\neq l}}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}\, \boldsymbol{e}^{\overline{\boldsymbol{i}\boldsymbol{\theta}_{il}^k}}\right] &= \text{Esp}\left[\sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}\, \boldsymbol{e}^{\overline{\boldsymbol{i}\boldsymbol{\theta}_{ij}^k}}\right] \\
&= \mathrm{C}_i^R
\end{aligned}$$

Nous sommes maintenant en mesure de continuer le calcul de la probabilité espérée que la symétrisation réussisse à la première micro-étape.

$$\begin{aligned}
\operatorname{Prob}\left[X_{1/n} = S\right] &= \sum_{i=0}^{R} \frac{1}{\mathrm{C}_i^R} \left(1 - \frac{\varepsilon^2}{n^2}\right)^{R-i} \left(\frac{\varepsilon^2}{n^2}\right)^{i} \left\| \sum_{j=1}^{\mathrm{C}_i^R} \prod_{k=1}^{R} \boldsymbol{e}^{-\boldsymbol{i}\theta_{ij}^k} \right\|^2 \\
&= \sum_{i=0}^{R} \left(1 - \frac{\varepsilon^2}{n^2}\right)^{R-i} \left(\frac{\varepsilon^2}{n^2}\right)^{i} \\
&\approx \frac{\left(1 - \frac{\varepsilon^2}{n^2}\right)^{R+1}}{1 - 2\frac{\varepsilon^2}{n^2}} \\
&\approx \left(1 - \frac{\varepsilon^2}{n^2}\right)^{R-1} \\
&\approx 1 - \frac{\varepsilon^2 (R-1)}{n^2}
\end{aligned}$$

On rappelle que le but est de ramener la probabilité de symétriser les $R$ ordinateurs quantiques pendant les $m$ étapes de l'algorithme aussi près de 1 que possible. Nous venons de calculer la probabilité que cette symétrisation réussisse après la première micro-étape. En supposant que cette symétrisation fut réussie, nous savons que le nouvel état combiné $|\psi_{1/n}\rangle$ se trouve d'autant plus près de $|\mathbf{0}\rangle^R$ que $n$ est grand. Pour calculer la probabilité de réussir la symétrisation à la deuxième micro-étape, nous devons d'abord déterminer comment le vecteur $|\psi_{1/n}\rangle$ évoluera, c'est-à-dire comment se comportera chacun des $R$ ordinateurs durant la seconde étape sachant qu'ils furent globalement symétrisés une première fois. Malheureusement, nous avons trop peu d'information pour connaître, à partir de $|\psi_{1/n}\rangle$, l'état exact des $M_i$ pour pouvoir calculer leur évolution subséquente. Nous pouvons néanmoins faire l'assertion suivante:

Assertion 6.1. *Le processus engendrant les erreurs étant unitaire (par hypothèse) et sachant qu'un opérateur unitaire préserve le produit scalaire, on peut affirmer que deux vecteurs initialement voisins resteront voisins suite à l'action du processus d'erreur.*

Donc, si $n$ est suffisamment grand, on peut considérer que l'évolution de $|\psi_{1/n}\rangle$ en $|\psi_{2/n}\rangle$ sera approximativement la même et si l'on considère que $|\psi_{1/n}\rangle = |\mathbf{0}\rangle^R$.

La probabilité de symétriser après la seconde micro-étape (en supposant que l'on a déjà symétrisé à la première) est donc encore $1 - \frac{\varepsilon^2(R-1)}{n^2}$. On peut appliquer le même raisonnement à chacune des $n$ micro-étapes et ainsi qu'aux $m$ étapes du calcul. Soit $Y$ la variable aléatoire de résultat d'avoir tenté de symétriser tout au long des $m$ étapes (avec $n$ micro-étapes pour chacune). La probabilité que $Y = S$ est

$$\text{Prob}\,[Y = S] \approx \left(1 - \frac{\varepsilon^2(R-1)}{n^2}\right)^{nm} \approx 1 - \frac{\varepsilon^2(R-1)m}{n} \tag{4.2}$$

On note que comme prévu, cette probabilité tend vers 1 lorsque $n$ tend vers l'infini. La probabilité ci-dessus indique que si l'on élève suffisamment la fréquence de symétrisation, on peut diminuer (en fonction de $1/n$) la probabilité que le calcul soit irrémédiablement détruit.

Avant d'analyser le second point de la méthode de symétrisation, soit la taille de l'erreur résiduelle, il est intéressant de revenir sur l'assertion 6.1.

**6.5. Considérations sur l'assertion 6.1.** S'il est aisé de calculer l'évolution des $M_j$ pour la première micro-étape jusqu'à la première symétrisation, il n'en va pas de même pour les micro-étapes subséquentes. Lorsque l'état combiné $|\psi\rangle$ a la forme $|\varphi\rangle^R$, on déduit naturellement que chaque $M_j$ se trouve dans l'état $|\varphi\rangle$. Le problème survient lorsque l'état combiné a une forme plus générale (mais toujours dans l'espace symétrique)

$$|\psi\rangle = \sum_{i=0}^{R} \alpha_i |E_i\rangle$$

L'état de chacun des $M_j$ ne s'obtient plus aussi directement à partir de $|\psi\rangle$. L'assertion 6.1 permet de contourner le problème, mais à quel point est-elle valide? La justesse de son application dépend de l'écart entre l'état combiné réel et le vecteur $|\mathbf{0}\rangle^R$, cet écart étant une fonction de $\varepsilon$, $m$ et $n$. L'évolution de l'erreur est unitaire (c'est-à-dire une rotation dans l'espace de calcul $\mathcal{H}$) mais on peut la considérer localement comme une translation si $\varepsilon m$ est petit et $n$ grand. Par analogie, la rotation de la terre sur elle-même au cours d'un intervalle de temps $t$ force Montréal, New York et Rio de Janeiro à parcourir une certaine distance.

On peut dire que Montréal et New York subissent un déplacement similaire, ce qui n'est pas vrai si l'on compare Montréal et Rio de Janeiro.

Cependant, plus $n$ est grand, plus il y aura de symétrisations par étape et, par conséquent, plus d'erreurs résiduelles de plus petite taille. L'analyse de l'erreur résiduelle devra tenir compte de ces considérations.

**6.6. Erreur résiduelle.** On se rappelle que selon le modèle d'erreur, toute action sur l'ordinateur le perturbera suffisamment pour que la direction de déviation de l'erreur soit changée. Ceci est valable pour les opérations de l'algorithme tout comme les symétrisations. On sait aussi que chaque ordinateur aura dévié d'un angle de ${}^{\varepsilon}/_{n}$ après la première micro-étape (avant symétrisation) dans une direction aléatoire et que cet angle sera probablement moindre (mais inconnu) après symétrisation. Supposons donc le pire cas que chaque symétrisation ne réduit pas l'angle de déviation, mais ne fait que changer l'angle de déviation. Ceci correspond au cas improbable que les déviations de chaque ordinateur sont toujours identiques (par étape). Par un argument de marche aléatoire similaire à celui de la section 4, on peut estimer qu'après les $nm$ symétrisations, chaque ordinateur aura dévié d'un angle espéré (en pire cas) $\frac{\varepsilon}{n}\sqrt{nm} = \varepsilon\sqrt{\frac{m}{n}}$.

Supposons que le résultat final du calcul de chacun des $R$ ordinateurs doit être un des états de base plutôt qu'une superposition quelconque. Soit $X$ et $Y$ respectivement les variables aléatoires du résultat d'un vote majoritaire et du résultat d'avoir symétrisé les $R$ ordinateurs tout au long des $nm$ micro-étapes. La probabilité que le vote majoritaire donne la bonne réponse étant donné que les $R$ ordinateurs furent toujours symétrisés est

$$\begin{aligned}
\text{Prob}\left[X = |\mathbf{0}\rangle \mid Y = S\right] &= 1 - \text{Prob}\left[X = |\mathbf{1}\rangle \mid Y = S\right] \\
&= 1 - \frac{\text{Prob}\left[X = |\mathbf{1}\rangle \wedge Y = S\right]}{\text{Prob}\left[Y = S\right]} \\
&= 1 - \frac{\sum_{i=\lceil R/2 \rceil}^{R} \left\|\langle E_i|\psi_{nm}\rangle\right\|^2}{\text{Prob}\left[Y = S\right]}
\end{aligned}$$

En utilisant le même argument qu'à la section 6.4 et le résultat de la section 4 (où l'on remplace $\varepsilon\sqrt{m}$ par $\varepsilon\sqrt{\frac{m}{n}}$), on obtient

$$\begin{aligned}\sum_{i=\lceil R/2\rceil}^{R} \|\langle E_i|\psi_{nm}\rangle\|^2 &\approx \sum_{i=\lceil R/2\rceil}^{R} \left(1-\frac{\varepsilon^2 m}{n}\right)^{R-i}\left(\frac{\varepsilon^2 m}{n}\right)^{i} \\ &\approx \left(\frac{\varepsilon^2 m}{n}\right)^{R/2}\end{aligned}$$

On sait déjà que $\mathrm{Prob}\,[Y=S] = 1-\frac{\varepsilon^2(R-1)m}{n}$. Donc

$$\begin{aligned}\mathrm{Prob}\,[X=|\mathbf{0}\rangle \mid Y=S] &= 1-\frac{\sum_{i=\lceil R/2\rceil}^{R}\|\langle E_i|\psi_{nm}\rangle\|^2}{\mathrm{Prob}\,[Y=S]} \\ &\approx 1-\frac{\left(\frac{\varepsilon^2 m}{n}\right)^{\frac{R}{2}}}{1-\frac{\varepsilon^2(R-1)m}{n}} \\ &\approx 1-\left(\frac{\varepsilon^2 m}{n}\right)^{R/2}\end{aligned}$$

Donc, si l'on possède $R$ ordinateurs tels que décrits ci-haut subissant des perturbations selon le modèle d'erreur de la section 2 et que la méthode de symétrisation est utilisée pour corriger ces erreurs, la probabilité que les $R$ calculs se rendent à terme *et* que le vote majoritaire donne la bonne réponse est

$$\mathrm{Prob}\,[Y=S]\times\mathrm{Prob}\,[X=|\mathbf{0}\rangle \mid Y=S] \approx \left[1-\frac{\varepsilon^2(R-1)m}{n}\right]\left[1-\left(\frac{\varepsilon^2 m}{n}\right)^{R/2}\right]$$

Il est intéressant de noter comment varie cette probabilité en fonction des paramètres $R$ et $n$. Si $R$ augmente, ceci améliore exponentiellement la probabilité d'un vote majoritaire correct, mais diminue linéairement la probabilité de symétriser tout au long du calcul. Ceci était prévisible car la dimension du produit tensoriel des $R$ ordinateurs croît exponentiellement en $R$ alors que l'espace symétrique associé ne croît que linéairement. Plus $R$ est grand, plus la taille relative du sous-espace dans lequel l'état combiné est projeté diminue. Par contre, on observe un curieux phénomène lorsqu'on analyse le paramètre $n$: même si $R=1$, la probabilité ci-dessus tend vers 1 a mesure que $n$ augmente! Ceci doit forcément être erroné puisque si $R=1$, le sous-espace symétrique est l'espace de calcul au complet et donc toute projection ne devrait avoir aucun effet. Comment expliquer que la fréquence de symétrisation influence le taux de correction d'erreur?

## 7. Révision de la méthode de symétrisation

Lorsque nous avons défini le modèle d'erreur à la section 2, nous avons fait l'hypothèse que la direction de déviation était modifiée de manière aléatoire à chaque étape de l'algorithme. Ceci modélisait l'idée intuitive qu'un ordinateur laissé à lui-même subit une déviation constante, mais lorsqu'une opération a lieu, celle-ci perturbe suffisamment le système pour modifier l'angle de déviation pour l'étape suivante. En toute logique, ce qui est vrai pour chaque étapes doit être tout aussi vrai pour chaque micro-étapes. C'est, du moins, l'hypothèse qui fut faite dans l'analyse de la section précédente.

Le modèle d'erreur dit qu'à chaque étape, l'état d'un ordinateur dévie d'un angle $\varepsilon$ dans une direction aléatoire. En forcant cette déviation à changer de direction à chaque micro-étapes, cela équivaut (d'un point de vue statistique) à dire que l'angle de déviation par étape est en fait de l'ordre de $\varepsilon/\sqrt{n}$. C'est ce qui explique que la fréquence de symétrisation influe sur le taux de correction d'erreur.

Le but du présent chapitre est d'analyser l'efficacité de la symétrisation et non pas l'efficacité d'un phénomène accidentel dû à l'effet watch-dog. Nous reprendrons donc l'analyse de la section précédente sous deux scénarios spéciaux qui permettront de voir le potentiel réel de la symétrisation en filtrant l'effet watch-dog. Ces scénarios sont:

- **Symétrisation sans perturbations.** Nous symétrisons à chaque micro-étape comme précédemment, mais la déviation n'est modifiée qu'à chaque étape seulement.
- **Symétrisation par étape.** Ici nous cessons complètement l'utilisation de l'effet watch-dog pour ne symétriser qu'une seule fois par étape.

Dans les deux sous-sections suivantes, on calcule la probabilité de réussir à symétriser durant tout le calcul ainsi que l'erreur résiduelle, et ce pour les deux scénarios ci-dessus. Par la suite nous comparerons leur efficacité.

**7.1. Symétrisation sans perturbations.** Nous changeons légèrement le modèle d'erreur: la direction de déviation ne sera pas affectée par la projection dans l'espace symétrique. On laisse le lecteur se convaincre que cette hypothèse ne change rien à l'équation 4.2. Par contre, le calcul de l'erreur résiduelle doit se faire de manière légèrement différente.

Si la déviation ne change pas au cours d'une étape malgré les $n$ symétrisations, l'erreur résiduelle à la fin d'une étape est simplement la somme scalaire (plutôt que vectorielle) des erreurs résiduelles de chaque micro-étape. Relativement à l'erreur résiduelle, cette situation est équivalente à avoir symétrisé une seule fois à chaque étape. Par l'argument de marche aléatoire encore une fois, la déviation finale espérée à la fin des $m$ étapes de calcul sera de $\varepsilon\sqrt{m}$. En reprenant l'analyse de l'erreur résiduelle de la section 6.6 avec ces valeurs, on trouve que la probabilité d'un vote majoritaire correct étant donné que l'on a réussi à symétriser tout au long du calcul est

$$1 - (\varepsilon^2 m)^{R/2}$$

Donc, pour ce scénario, la probabilité que les $R$ calculs se rendent à terme et que le vote majoritaire donne la bonne réponse est

$$\left[1 - \frac{\varepsilon^2(R-1)m}{n}\right]\left[1 - (\varepsilon^2 m)^{R/2}\right]$$

**7.2. Symétrisation par étape.** Pour ce scénario, nous n'utilisons plus du tout l'effet watch-dog: la symétrisation se fera une fois par étape. On laisse le lecteur vérifier que l'équation 4.2 devient, dans ces conditions

$$\text{Prob}\,[X = S] \approx 1 - \varepsilon^2(R-1)m$$

et que la probabilité d'un vote majoritaire correct étant donné une symétrisation réussie est, comme pour le scénario précédant,

$$1 - (\varepsilon^2 m)^{R/2}$$

## 8. Comparaison des méthodes de correction d'erreur

Dans ce chapitre, nous avons considéré la probabilité d'obtenir une réponse juste en utilisant un ou des ordinateurs subissant une certaine forme de perturbation. Après avoir calculé cette probabilité pour le cas de base, soit avec un seul ordinateur, une première méthode pour améliorer cette probabilité fut proposée: prendre $R$ ordinateurs en parallèle et prendre le vote majoritaire à la fin des calculs. Cette méthode possède une certaine efficacité, mais ne peut s'appliquer dans tous les cas. Nous nous sommes donc tournés vers la méthode de symétrisation proposée par Deustch. Cette dernière, plus générale, utilise également une redondance de calculs, mais d'une manière ne détruisant pas les superpositions quantiques. Pour juger de son efficacité, on peut la comparer avec la méthode du vote majoritaire (pour le type de calcul où cette dernière est applicable). Le tableau de la figure 4.1 récapitule les données. On remarque que toutes ces

| **Nombre d'ordinateurs** | **Méthode de de correction** | **Probabilité espérée (avec $\varepsilon m$ petit)** |
|---|---|---|
| 1 | aucune | $1-\varepsilon^2 m$ |
| R | Vote majoritaire | $1-\mathrm{C}^{R}_{\lceil R/2\rceil}(\varepsilon^2 m)^{\lceil R/2\rceil}$ |
| R | Symétrisation (sans perturbations) | $\left[1-\frac{\varepsilon^2(R-1)m}{n}\right]\left[1-(\varepsilon^2 m)^{R/2}\right]$ |
| R | Symétrisation (une fois par étape) | $[1-\varepsilon^2(R-1)m]\left[1-(\varepsilon^2 m)^{R/2}\right]$ |

Figure 4.1. Récapitulation des probabilités espérées en fonction du cas traité.

probabilités sont en terme de $\varepsilon^2 m$. Ces deux paramètres, la taille de la déviation et la durée de l'algorithme, sont donnés par la définition du problème. Nous sommes donc en mesure d'interpréter ces probabilités en termes d'efficacité.

En considèrant la méthode de symétrisation, on doit garder à l'esprit le fait suivant: *l'usager sait si une symétrisation échoue*, auquel cas, le travail déjà fait est perdu et il doit reprendre le calcul à nouveau. Pour autant que l'on ne doit pas recommencer trop souvent, ceci n'est pas un problème puisque si l'usager réussi à symétriser tout au long du calcul, alors il est garanti d'avoir un vote majoritaire (si applicable) correct avec une probabilité $1 - (\varepsilon^2 m)^{R/2}$. Ceci est comparable aux algorithmes de type *Las Vegas*: à chaque essai, on a une probabilité d'avoir la réponse "désolé, meilleure chance la prochaine fois", mais sinon, on sait que la réponse obtenue est relativement fiable. Comme pour les algorithmes *Las Vegas*, la symétrisation sera intéressante à la condition que le temps espéré pour symétriser tout au long d'un calcul de $m$ etapes ne soit pas exagérément long.

THÉORÈME 8.1. *Soit un événement donné $X$ se produisant avec probabilité $1-p$, $p \ll 1$ à chaque unité de temps. L'espérance du nombre d'étape à attendre pour avoir un série de $k$ étapes consécutives où $X$ s'est produit est approximativement*

$$k\left(1+\frac{kp}{2}\right)$$

PREUVE. Immédiat en considérant les termes en $p^2$ négligeables. ⊠

Le théorème 8.1 permet d'évaluer le nombre réel d'étapes de calcul qui seront nécessaires pour symétriser tout au long d'un algorithme de $m$ étapes.

- **Symétrisation sans perturbations:** chaque symétrisation se fait avec probabilité
$$1-\frac{\varepsilon^2(R-1)}{n^2}$$
et il y a $nm$ symétrisations à faire, donc par le théorème 8.1, le nombre total espéré de symétrisations à faire est
$$nm\left(1+\frac{\varepsilon^2(R-1)m}{2n}\right)$$
et pour un $n$ suffisamment grand, ceci représente une augmentation négligeable du temps de calcul.

- **Symétrisation par étapes:** chaque symétrisations se fait avec probabilité
$$1-\varepsilon^2(R-1)$$
et il y a $m$ symétrisations à faire, donc par le théorème 8.1, le nombre total espéré de symétrisation à faire est
$$m\left(1+\frac{\varepsilon^2(R-1)m}{2}\right)$$
Puisque seul les cas où $\varepsilon^2 m \ll 1$ sont considérés ici, ceci représente une augmentation négligeable du temps de calcul au autant que
$$R<\frac{2}{\varepsilon^2 m}$$

Donc, pour les deux scénarios de symétrisation, le temps réel de calcul ne semble pas significativement augmenté par les probabilités d'échec relativement aux projections dans le sous-espace symétrique. C'est donc dire que pour une augmentation minime du temps de calcul, la symétrisation suivie d'un vote majoritaire final offre une meilleure garantie de succès (par un facteur $\mathrm{C}^R_{\lceil R/2\rceil}$) qu'un le vote majoritaire seul. Ceci à condition, bien sûr, que la réponse finale du calcul permette de faire un vote majoritaire. Sinon, non seulement la symétrisation est la seule technique (connue) qui puisse stabiliser le calcul, mais en plus, elle offre l'avantage que l'usager sait si oui ou non la stabilisation fut réussie.

## 9. Correction d'erreur sous un modèle plus général

Les résultats de la section 8 indiquent que la symétrisation est une excellente méthode de stabilisation des calculs quantique, *sous l'hypothèse* que le modèle d'erreur défini à la section 2 est valable *et* que les ordinateurs quantiques utilisés n'ont qu'un seul bit de mémoire. Cette situation est loin d'être réaliste. Qu'en est-il pour des ordinateurs quantiques plus sérieux?

**9.1. Un modèle d'erreur réaliste.** Les erreurs qui peuvent influencer le calcul d'un ordinateur prennent plusieurs formes. La déviation aléatoire simple qui fut étudiée ici est probablement le modèle d'erreur le plus simple. Notons cependant que ce dernier est trop simple pour être réaliste. Toutefois, sur le thème de la déviation, on peut aussi avoir une déviation avec un biais pour une direction donnée (dépendant ou indépendant de l'algorithme). D'autres modèles d'erreur pourraient inclure des influences de l'environnement, ces influences allant jusqu'à simuler un rayon cosmique frappant l'ordinateur quantique (transformant l'état $|\varphi\rangle$ en $|\varphi'\rangle$ de manière instantanée et imprévisible).

Pour tous ces modèles d'erreur, l'efficacité de la symétrisation pour stabiliser les calculs varie énormément: pour certains, elle est très efficace (comme nous l'avons démontré dans cette thèse); pour d'autres, elle est sans effet. Mais pour la plupart des modèles d'erreur réalistes, l'efficacité est encore inconnue. L'analyse du fonctionnement de la symétrisation fait à la section 6 nous permet toutefois d'être optimiste.

**9.2. Coût d'une symétrisation.** Dans l'analyse de la section 6, aucune restriction n'est mise sur le nombre ($n$) de symétrisations par étapes. On permet à $n$ d'être aussi grand que nécessaire. Ceci devient inacceptable si l'on doit considérer un certain coût par symétrisation. Par exemple, l'algorithme de symétrisation de Jozsa, donné à l'appendice A, prend un temps dans l'ordre de $O(R \log R)$. Le coût par symétrisation impose donc une limite sur la redondance ($R$) utilisée. Même en considérant faire une seule symétrisation par étape de calcul (le second scénario dans l'analyse de la section 7), le temps pris pour exécuter l'ensemble des symétrisations allongera de manière significative le temps de calcul. Ces considérations ne tiennent pas compte non plus que l'algorithme de symétrisation lui-même pourrait introduire des erreurs! Dans ce contexte, on pourrait imaginer faire des symétrisations encore plus éparses, une fois toutes les $k$ étapes par exemple. Une augmentation de la redondance $R$, jointe à des symétrisations moins fréquentes pourraient compenser pour le temps pris à symétriser et conserver une bonne réduction d'erreur.

**9.3. Symétriser avec des ordinateurs à $N$ bits.** Lorsque l'on passe à des ordinateurs à $N$ bits, le scénario se complique énormément. Sans compter toute la nouvelle liberté offerte par ces $N$ bits pour permettre des modèles d'erreur parfois complexes, il faut aussi considérer l'effort mis à l'acte même de symétrisation. L'algorithme décrit à l'appendice A, dans le cas d'ordinateurs à $N$ bits, aura une complexité en fonction de $R$ *et* $N$. Il est possible que ceci impose une contrainte supplémentaire sur le niveau de redondance $R$.

**9.4. Problèmes ouverts.** La symétrisation semble au moins prometteuse (sans être une panacée contre tous les types d'erreurs). Ce fait établi, on peut dorénavant aborder les questions soulevées dans les sous-sections ci-dessus pour déterminer dans chaque cas précis l'efficacité de cette méthode de stabilisation d'erreur de calcul.

# CHAPITRE 5

## Conclusion

Deutsch a présenté en 1985 un modèle de calcul basé sur la mécanique quantique. Il a démontré l'existence d'un problème pour lequel il existe un algorithme quantique exponentiellement plus rapide sur le modèle quantique que tout algorithme sur une machine de Turing. Ceci, ajouté au fait que le modèle quantique soit physiquement réalisable, justifie une étude plus appronfondie de la complexité du calcul quantique.

Nous avons concentré cette étude sur la classe des problèmes pour lesquels il existe une solution en temps polynomial sur l'ordinateur quantique; soit la classe **EQP**. Nous nous sommes intéressés à savoir où se situait **EQP** dans la hiéarchie de complexité relativisée. L'existence d'oracles $X$,$Y$ et $Z$ pour lesquels nous avons les relations $\mathbf{P}^X \subsetneq \mathbf{EQP}^X$, $\mathbf{EQP}^Y \not\subseteq \mathbf{NP}^Y \cup \textbf{co-NP}^Y$, $\mathbf{EQP}^Z \not\subseteq \mathbf{ZPP}^Z$, $\mathbf{EQP}^Z \not\subseteq \mathbf{RP}^Z \cup \textbf{co-RP}^Z$ nous permet d'esprérer que l'ordinateur quantique est plus puissant qu'une machine de Turing même dans le cas non relativisé.

La réalisation du modèle quantique, bien que théoriquement possible, pose de nombreux problèmes technologiques et conceptuels. L'un de ceux-ci est que le modèle quantique est naturellement sujet à des perturbations qui peuvent entraîner des erreurs de calculs. Deutsch propose une méthode de stabilisation des calculs et nous avons analysé l'efficacité de cette méthode sur une modèle simplifié d'erreur. Cette analyse montre que cette méthode de stabilisation, la symétrisation, se compare avantageusement avec la méthode classique de vote majoritaire (pour les cas où ces deux méthodes peuvent s'appliquer). Le vrai avantage de la symétrisation est qu'elle peut s'appliquer à toutes les situations quantiques, contrairement aux autres méthodes habituelles. Dans ce sens, la conclusion positive de l'analyse d'efficacité augure bien pour cette méthode de stabilisation d'erreur lorsqu'un modèle d'erreur plus réaliste sera considéré.

En conclusion, si beaucoup de questions demeurent encore ouvertes relativement à ce nouveau paradigme de la complexité du calcul qu'est le modèle quantique, la présente thèse offre plusieurs réponses intéressantes.

# APPENDICE A

## Algorithme de symétrisation

Les alchimistes médiévaux croyaient fermement qu'il était possible de transmuter le plomb en or. Si nous considérons que leur approche au problème était quelque peu farfelue, avec leurs incantations, leurs symboles cabalistiques et leurs potions infernales, force est d'admettre que ces alchimistes avaient raison: il est effectivement possible de transmuter le plomb en or! Prenez un atome de plomb, enlevez-lui cinq protons et cinq électrons et quelques neutrons et vous avez un atome d'or. Malheureusement, cette manipulation est très difficile en pratique. C'est l'éternel conflit entre la théorie et la pratique.

### 1. Un modèle physiquement réalisable

Lorsqu'on dit que l'ordinateur quantique est physiquement réalisable, il est nécessaire de spécifier le niveau de cette réalisation. À l'extrémité théorique du spectre, le modèle quantique est physiquement réalisable si les opérateurs en jeux sont unitaires et les observables sont biens définis. À l'opposé, du côté pratique du spectre, le modèle quantique est physiquement réalisable si le niveau technologique actuel est suffisant pour le construire. L'endroit exact dans ce spectre où l'on choisit de placer le seuil de réalisation physique est une question de point de vue.

Dans la formalisation du modèle de calcul quantique de Deutsch [9] , chaque opération élémentaire (qui sont en nombre fini) du modèle est décrite par un opérateur unitaire relativement simple. Sous l'hypothèse que ces opérateurs sont physiquement réalisables, il suffit de traduire un algorithme $A$ en une suite de ces opérations élémentaires pour conclure que $A$ est physiquement réalisable. Cette position dans le spectre de réalisation physique se trouve près du côté théorique sans toutefois être à l'extrémité.

Dans le cadre de cette thèse, nous prendrons le même point de vue, mais nous appliquons aussi ce principe aux observables.

## 2. Un observable physiquement réalisable

Un observable est une partition quelconque de l'espace $\mathcal{H}$ en sous-espaces $E_1, \ldots, E_k$. Si toutes les partitions sont permises, cela n'implique pas qu'elles sont toutes aussi aisée à exécuter. Lorsqu'on considère des observables sur un enchevêtrement d'ordinateurs comme dans le cas de la symétrisation, on risque de rencontrer des problèmes à réaliser ces observations.

Dans la méthode de symétrisation de Deutsch pour stabiliser les calculs quantiques, on utilise un observable de projection dans l'espace symétrique. Bien que cet observable soit bien défini, sa réalisation physique est loin d'être évidente. Jozsa [16] propose un algorithme qui exécute une projection dans l'espace symétrique et dont on est en droit de supposer une réalisation plus simple.

## 3. Symétrisation de deux ordinateurs

Soit $H_1$ et $H_2$ les deux espaces correspondant à deux ordinateurs quantiques tels que décrits au chapitre 4, section 3. Soit $\mathcal{H} = H_1 \otimes H_2$ l'espace combiné des deux systèmes. Pour symétriser ces deux ordinateurs, nous avons besoin de considérer un troisième système, un *ancilla*, à deux états dont l'espace $A$ engendré par les vecteurs de base $|1\rangle$ et $|2\rangle$. Cet ancilla sert de système de contrôle: la valeur de celui-ci indique quelle permutation est appliqué sur $H_1 \otimes H_2$. Les deux opérateurs principaux de cet algorithme sont $S$ et $P$: $S$ crée une superposition de toutes les valeurs possible de l'ancilla et $P$ permute les indices de $\mathcal{H}$ selon la valeur de l'ancilla. Plus précisément:

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

(on note que $S^{-1} = S$). Soit un opérateur $P$ qui permute $H_1$ et $H_2$ si l'ancilla est $|2\rangle$ et ne fait rien si l'ancilla est $|1\rangle$ (les permutations étant clairement des opérations réversibles, $P$ est unitaire). En fait, $P$ est un opérateur bloc diagonal tel que

$$\left(\begin{array}{cc} \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} & 0 \\ 0 & \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \end{array}\right)$$

La symétrisation se fait en trois étapes suivies d'une observation. Soit $|a\rangle$ et $|b\rangle$ deux états de base de $H_1$ et $H_2$ respectivement. Initialement, l'ancilla est dans l'état $|1\rangle$ et donc le système combiné des deux ordinateurs et de l'ancilla est dans l'état

$$|ab, 1\rangle$$

La première étape consiste à mettre l'ancilla en superposition de toutes ses valeurs.

$$S|ab, 1\rangle = \frac{1}{\sqrt{2}}(|ab, 1\rangle + |ab, 2\rangle)$$

Ensuite, on permute les ordinateurs en fonction de la valeur de l'ancilla.

$$PS|ab, 1\rangle = \frac{1}{\sqrt{2}}(|ab, 1\rangle + |ba, 2\rangle)$$

Maintenant, on applique l'inverse de $S$

$$\begin{aligned} S^{-1}PS|ab, 1\rangle &= \frac{1}{2}\left(|ab, 1\rangle + |ab, 2\rangle + |ba, 1\rangle - |ba, 2\rangle\right) \\ &= \frac{1}{2}\left[(|ab\rangle + |ba\rangle)|1\rangle + (|ab\rangle - |ba\rangle)|2\rangle\right] \end{aligned}$$

Finalement, on observe la valeur de l'ancilla. On rapelle que $|ab\rangle$ est un des états de base de $\mathcal{H}$. Si $|ab\rangle = |\mathbf{00}\rangle$ ou $|\mathbf{11}\rangle$ (c'est-à-dire que $|ab\rangle$ est symétrique) alors

$|ab\rangle = |ba\rangle$ et donc avec probabilité 1 la valeur lue de l'ancilla est $|1\rangle$. D'autre part, si $|ab\rangle = |\mathbf{01}\rangle$ ou $|\mathbf{10}\rangle$ alors

$$|ab\rangle = \frac{1}{\sqrt{2}}E_1 \pm \frac{1}{2}(|ab\rangle - |ba\rangle)$$

où $E_1 = \frac{1}{\sqrt{2}}(|\mathbf{01}\rangle + |\mathbf{10}\rangle)$, un des vecteurs de base de l'espace de $\mathrm{Sym}\,(\mathcal{H})$). Donc, si la valeur lue de l'ancilla est $|1\rangle$, alors l'état résiduel des deux ordinateurs combinés est leur composante dans l'espace symétrique. Par linéarité, ce qui est vrai pour les vecteurs de base est aussi vrai pour une combinaison linéaire de ceux-ci. Ainsi, pour tout état $|\varphi\rangle$ de $\mathcal{H}$ sur lequel on applique l'algorithme ci-dessus et que la valeur finale de l'ancilla est $|1\rangle$, nous sommes garantis d'avoir comme état résiduel la composante (normalisée) de $|\varphi\rangle$ dans l'espace symétrique. La probabilité de succès de cet algorithme de symétrisation (obtenir la valeur $|1\rangle$) est d'autant meilleure que $|\varphi\rangle$ est "proche" de l'espace symétrique.

## 4. Symétrisation de $R$ ordinateurs

Pour généraliser l'algorithme de symétrisation à $R$ ordinateurs, on doit modifier les opérateurs $S$ et $P$. On rapelle que l'opérateur $S$ doit mettre l'ancilla en superposition de toutes ses valeurs pour qu'ensuite l'opérateur $P$ puisse appliquer toutes les permutations possibles sur les $R$ ordinateurs. La dimension de l'espace $A$ de l'ancilla doit donc être au moins $R!$.

Pour généraliser $S$, nous utilisons les matrices de *Sylvester*. Les matrices de Sylvester forment une classe de matrices de Hadamard qui est définie récursivement comme suit:

$$\begin{aligned} S_1 &= (1) \\ S_{2i} &= \begin{pmatrix} S_i & S_i \\ S_i & -S_i \end{pmatrix} \qquad \text{pour } i > 1 \end{aligned}$$

(On note que $S_i S_i = 2^k I$.) Les matrices de Sylvester n'étant définies que pour les tailles d'une puissance de 2, on fixera la dimension de l'ancilla au plus petit entier $k$ supérieur à $R!$. Soit $I_R$ la matrice identité de dimension $R$. On pose

$$S = I_R \otimes \frac{1}{\sqrt{2^k}} S_k$$

et $P$ est tel que pour les valeurs 1 à $R!$ de l'ancilla, $P$ applique chacune des $R!$ permutations des $R$ ordinateurs. Pour les valeurs $R! + 1$ à $2^k - 1$, $P$ applique la permutation identité. On laisse au lecteur le soin de vérifier que les conclusions de la section précédente sont encore valides avec les opérateurs $S$ et $P$ ainsi définis.

# RÉFÉRENCES


[1] Benioff P.A. – Quantum mechanical hamiltonian models of turing machines. *Journal of Statistical Physics*, vol. 29, n° 3, 1982, pp. 515–546.

[2] Bennett C. H. – Logical reversibility of computations. *IBM Journal of Res. Develop.*, vol. 17, 1973, pp. 525–532.

[3] Berstein E. et Vazirani U. – Quantum complexity theory. Dans : *Proceedings of the 25st Annual ACM Symposium on the Theory of Computation*, pp. 11–20.

[4] Berthiaume A. et Brassard G. – Oracle quantum computing. Dans : *Proceedings of the Workshop on Physics and Computation – Physcomp '92.* pp. 195–199. – IEEE Press.

[5] Berthiaume A. et Brassard G. – The quantum challenge to structural complexity. Dans : *Proceedings of the 7th Annual IEEE Conference on Structure in Complexity*, pp. 132–137.

[6] Berthiaume A. et Brassard G. – Oracle quantum computing. *Journal of Modern Optics*, vol. 41, n° 12, 1994.

[7] Blum M. – A machine independant theory of the complexity of recursive functions. *Journal of the ACM*, vol. 14, n° 2, 1967, pp. 322–336.

[8] Bovet D.P. et Crescenzi P. – *Introduction to the theory of Complexity.* – Prentice Hall, 1994.

[9] Deutsch D. – Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society,* London, vol. A400, 1985, pp. 97–117.

[10] Deutsch D. – Quantum computational network. *Proceedings of the Royal Society,* London, vol. A425, 1989, pp. 73–90.

[11] Deutsch D. et Jozsa R. – Rapid solutions of problems by quantum computation. Dans : *Proceedings of the Royal Society,* London, pp. 553–558.

[12] Feynman R. – Simulating physics with computers. *International Journal of Theoretical Physics*, vol. 21, n° 6/7, 1982, pp. 467–488.

[13] Gill J. – Computational complexity of probabilistic turing machines. *SIAM Journal of Computing*, vol. 6, 1977, pp. 675–695.

[14] Helstrom C.W. – *Quantum Detection and Estimation Theory.* – Academic Press, 1976.

[15] Jozsa R. – Characterizing classes of functions computable by quantum parallelism. Dans : *Proceedings of the Royal Society,* London, pp. 563–574.

[16] Jozsa R. – Communication personnelle, 1993.

[17] Lecerf Y. – Machines de Turing réversibles. Récursive insolubilité en $n \in N$ de l'équation $u = \theta^n$ où $\theta$ est un isomorphisme de codes. Dans : *Comptes rendus de l'Académie française des sciences*, pp. 2597 – 2600.

[18] Shor P. – Algorithms for quantum computation: Discrete log and factoring. Dans : *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science.*

[19] Simon D. – On the power of quantum computation. Dans : *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science.*

[20] Sudbury A. – *Quantum Mechanics and the Particles of Nature – An Outline for Mathematicians.* – Cambridge University Press, 1986.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX

**Document Log:**

Manuscript Version 2 —
Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX — 10 April 1995

# L'ORDINATEUR QUANTIQUE: COMPLEXITÉ ET STABILISATION DES CALCULS

André Berthiaume

Université de Montréal
CP 6128 succ. centre-ville,
Montréal (Québec) H3C 3J7, Canada

*E-mail address*: berthiau@iro.umontreal.ca