POLYTECH ANGERS

université angers

Master Systèmes Dynamiques et Signaux

Mémoire de master

Conception de détecteurs quantiques optimaux via le calcul par intervalles

Auteur :
M. Pierre Engelstein

Encadrants :
Dr. Nicolas Delanoue
Pr. François Chapeau-Blondeau

Jury :
Pr. Laurent Hardouin
Dr. Nicolas Delanoue
Pr. François Chapeau-Blondeau
Pr. Sébastien Lahaye
Dr. Mehdi Lhommeau
Pr. David Rousseau

Version du
15 juin 2021

Remerciements

Je remercie Dr. Nicolas Delanoue et Pr. François Chapeau-Blondeau pour leur encadrement sur ce travail. Je remercie également mes parents pour les encouragements et l'aide apportés sur cette année de Master.

Table des matières

Table des figures

Chapitre 1

Introduction

L'informatique quantique, application des théories quantiques développées depuis le début du vingtième siècle à la théorie de l'information puis spécifiquement au calcul, est aujourd'hui en plein développement, théorique avec la découverte de nouveaux algorithmes, mais aussi pratique avec l'intérêt porté par les différents industriels. On voit alors l'apparition de nouvelles plateformes basées sur des processeurs quantiques, permettant de mettre en place les différentes avancées théoriques.

L'informatique quantique offre une accélération de certains traitements permettant théoriquement d'effectuer des calculs qui seraient infaisables en des temps raisonnables sur nos calculateurs classiques, par exemple le cassage des clés cryptographiques dans l'ordre de minutes au lieu de dixaines d'années.

Avec ce nouveau champ, de nombreuses questions se posent sur nos infrastructures actuelles, notamment en termes de cybersécurité et il est alors important de comprendre les capacités qu'offre l'informatique quantique.

Ce travail présente en première partie les notions fondamentales à la compréhension de l'informatique quantique. On y montre ce qu'est un qubit, puis on explique le mécanisme de mesure qui vient apporter de la probabilité, et enfin les principes d'évolution de systèmes quantiques permettant de construire des systèmes de calcul.

La deuxième partie de ce travail présente un tableau des mises en œuvres expérimentales au travers des processeurs quantiques et des simulateurs, développés par les différents industriels tels que Microsoft, IBM, Google et Atos depuis le début des années 2010.

Enfin, la dernière partie présente trois algorithmes majeurs au développement de l'informatique quantique, qui notamment illustrent les apports spécifiques du quantique, avec des performances de traitement de l'information inaccessibles en classique. On explique tout d'abord l'algorithme de Deutsch-Jozsa, sur la parallélisation d'évaluation de fonction ; puis l'algo-

rithme de Grover, sur la recherche de base de données ; et enfin l'algorithme de Shor sur la factorisation en nombres premiers.

Chapitre 2

Informatique quantique : éléments de base

Les notions de base d'informatique quantique sont décrites dans plusieurs ouvrages de référence, notamment dans [1, 2]. On présente ici un résumé des notions fondamentales à connaître pour la suite du rapport.

2.1 Postulats de base

On pose 3 postulats, servant de base aux raisonnements qui suivront. Ces postulats sont confirmés jusqu'à présent par les expériences.

2.1.1 Etat d'un système quantique

Un système quantique peut être représenté par un vecteur d'état, de la même manière qu'un système physique classique. On le représente par la notation de Dirac, notée de la forme $|\psi\rangle$. Ce vecteur d'état est nécessairement de norme 1 (la somme des modules au carré vaut 1). On peut distinguer deux types d'états pour un système quantique : les états de base, formant une base orthonormée d'un espace vectoriel complexe, et les états superposés. Ces états superposés correspondent à une combinaison linéaire des états de base. On peut écrire généralement un état quantique de la façon suivante :

$$|\psi\rangle = \sum_i c_i \, |k_i\rangle \, , \tag{2.1}$$

avec les $|k_i\rangle$ états de base, et les c_i respectant $\sum_i |c_i|^2 = 1$ pour la normalisation du vecteur d'état.

Dans le cadre de l'informatique quantique, on utilise le système quantique le plus simple, appelé **qubit**. Ce système quantique est composé de deux états de base, $|0\rangle$ et $|1\rangle$, et des états superposés. Similairement à l'informatique classique, où on travaille sur le système physique le plus élémentaire - le

bit - en quantique on travaille sur le système physique quantique élémentaire - le qubit. On dispose des mêmes états de base, mais l'informatique quantique apporte les états *intermédiaires* superposés. Dans la base canonique $\{|0\rangle, |1\rangle\}$, on note un qubit de la façon suivante : $|\psi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle$.

2.1.2 Mesure projective

Que se passe-t-il quand on mesure un système quantique ? On a évoqué au dessus la notion de superposition des états. L'expérience montre que, lorsqu'on va mesurer un système quantique, on va mesurer au hasard un des états de base, avec comme probabilité le carré du coefficient correspondant.

Mathématiquement, la mesure effectue une projection de l'état du système sur un des états de base dont il est composé. Par exemple, si on a un qubit dans l'état $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, alors la probabilité de mesurer 0, c'est-à-dire de projeter le système dans l'état de base $|0\rangle$ est $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$; et de même pour l'état de base $|1\rangle$. On a donc exactement la même probabilité de mesurer $|0\rangle$ que de mesurer $|1\rangle$.

Il faut noter que, lorsqu'on fait la mesure, on projette réellement le système quantique dans l'état de base. Concrètement, si on a un état superposé qu'on mesure, il se place dans l'état de base qu'on mesure, et toutes les mesures successives qu'on fera sur ce qubit donneront le même résultat. La mesure fait donc perdre l'état qu'on avait auparavant.

2.1.3 Dynamique du système

Comme n'importe quel système physique, on peut faire évoluer un système quantique dans le temps. Ici apparaissent deux propriétés. Tout d'abord, il découle du premier postulat que la dynamique d'un système quantique doit conserver la norme unité. En effet, un état quantique doit, pour être valide, avoir une norme de 1, et donc l'évolution d'un système quantique d'un premier état vers un autre doit conserver cette unitarité. Cela veut dire que la matrice représentant l'évolution du système quantique doit respecter la propriété suivante :

$$U^\dagger U = U U^\dagger = I, \tag{2.2}$$

avec U la matrice d'évolution du système, $U^\dagger$ la matrice conjuguée transposée, ou adjointe, de U, et I l'identité.

Une deuxième propriété est également posée, ne découlant pas des deux postulats précédents. La dynamique d'un système quantique doit être aussi linéaire. Ainsi, on pourrait penser que n'importe quelle évolution unitaire

serait valable, mais l'expérience nous montre que non, il faut en plus qu'elle soit linéaire.

2.2 Vers une informatique quantique

À partir de ces 3 postulats de base, on peut commencer à comprendre comment se construit l'informatique quantique, et quels sont les apports sur l'informatique classique.

2.2.1 Multiples qubits

On a vu la définition d'un qubit. Cela nous permet d'étendre ce système quantique élémentaire à des systèmes composés de multiples qubits. En informatique classique, on travaille quasi systématiquement sur des mots binaires plutôt que des bits uniques ; l'équivalent est vrai en quantique. Pour cela, les systèmes quantiques, et donc les qubits, sont munis d'une opération : le produit tensoriel. Quand on veut effectuer une combinaison de deux qubits, cela revient à faire un produit tensoriel des états des deux qubits individuels. Par exemple, si nous disposons de deux qubits ayant pour valeur $|\psi_1\rangle = |0\rangle$ et $|\psi_2\rangle = |1\rangle$, alors on peut écrire le 2-qubit combinaison des deux de la façon suivante :

$$|\psi\rangle = |0\rangle \otimes |1\rangle \,, \tag{2.3}$$

qu'on écrit généralement sous la forme plus simple :

$$|\psi\rangle = |01\rangle \,. \tag{2.4}$$

Prenons un 2-qubit formé par la combinaison de 2 qubits :

$$\begin{aligned} |\psi\rangle &= (\alpha_1 \cdot |0\rangle + \beta_1 \cdot |1\rangle) \otimes (\alpha_2 \cdot |0\rangle + \beta_2 \cdot |1\rangle), \\ &= \alpha_1\alpha_2 \, |0\rangle \otimes |0\rangle + \alpha_1\beta_2 \, |0\rangle \otimes |1\rangle + \beta_1\alpha_2 \, |1\rangle \otimes |0\rangle + \beta_1\beta_2 \, |1\rangle \otimes |1\rangle \,. \end{aligned}$$

On peut donc, si on a un 2-qubit combinaison linéaire de tous les états de base, le factoriser en deux qubits séparés qu'on peut caractériser.

Considérons maintenant le 2-qubit suivant :

$$|\psi\rangle = \gamma_1 \, |00\rangle + \gamma_2 \, |11\rangle$$

Il parait évident alors qu'on ne peut pas factoriser ce 2-qubit en produit tensoriel de 2 qubits individuels. Dans ce cas, on dit que les deux qubits sont **intriqués** et donc non séparables.

2.2.2 Portes quantiques

Dans la représentation d'état classique, et spécifiquement en informatique, on peut faire évoluer l'état au travers de portes. En informatique classique, on dispose ainsi de portes élémentaires telles que `AND`, `NOT`, `OR`, etc.

De la même manière, en respectant le troisième postulat posé précédement, on peut construire des portes logiques quantiques, les combiner, afin de créer des circuits quantiques. Ces portes quantiques sont nécessairement unitaires, donc inversibles. En informatique quantique, on distingue donc plusieurs portes élémentaires, utilisées dans beaucoup de circuit [3] :

Les portes quantiques sont complètement caractérisées par la façon dont elles transforment les états quantiques dans la base canonique. On peut alors utiliser des tables de vérité pour les définir, de la même façon qu'en informatique classique :

1. La porte de Hadamard H. Elle permet de passer un qubit d'un état de base $|0\rangle$ à l'état superposé $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, ou de l'état de base $|1\rangle$ à l'état superposé $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Elle est très utilisée en début de circuit pour préparer les qubits entrants dans un état permettant l'évaluation parallèle de toutes les entrées ;
2. Les portes de Pauli X, Y et Z permettant d'effectuer des rotations aux états des qubits ;
3. La porte de Toffoli, similaire d'un `NON` booléen à 3 qubit (il effectue un `NON` sur le dernier qubit quand les deux premiers sont à $|1\rangle$), est une porte universelle quantique [4]. Elle permet donc de construire l'ensemble des autres portes faisables.

Les tables de vérité des différentes portes quantiques évoquées sont disponibles en annexes.

Un exemple de circuit est le suivant : On dispose d'un 3-qubit dans l'état $|000\rangle$. Au départ, on applique à ces trois qubits une porte de Hadamard, qui les fait se retrouver dans une superposition équilibrée des états de base (c'est-à-dire de sorte qu'une mesure nous donne l'un des états de base avec une probabilité de $\frac{1}{8}$). On applique ensuite deux portes de Pauli Z, un au premier qubit, et un au troisième. On applique de la même façon 3 portes de Hadamard à la sortie, puis on mesure.

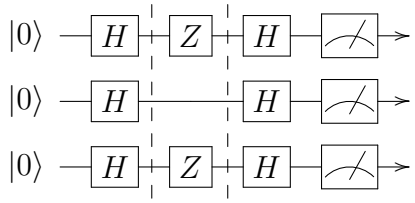


FIGURE 2.1 – Exemple de circuit quantique, avec U_f pour $f(a, b, c) = (\neg a) \oplus c$

Dans cet exemple, les deux portes de Pauli Z permettent d'appliquer une fonction f booléenne. En informatique classique, pour évaluer cette fonction sur les 2^n entrées possibles, il faudrait effectuer 2^n évaluations de f. Ici, avec ce circuit quantique, on n'effectue qu'une seule évaluation quantique de f sur l'état équilibré donné par les portes de Hadamard. On vient évaluer la fonction f sur tout les états de base composant l'état quantique. Puisque cet état est équilibré, les 2^n états de base vont être évalués, c'est-à-dire toutes les entrées qu'on voulait évaluer en classique.

L'avantage du quantique bien montré ici : les deux portes du milieu vont faire changer l'état des qubits, mais en parallèle : on fait évoluer le système simultanément pour tous les états de base qui nous intéressent, puisqu'ils sont superposés.

Chapitre 3

Calcul par intervalle : éléments de base

Le calcul par intervalle est décrit au départ dans les travaux de Ramon Moore [5]. L'utilité de ce mode de calcul vient des problèmes que représente le stockage des nombres réels dans nos ordinateurs via la norme `IEEE 754`. En effet, on sait avec cette norme facilement représenter une certaine quantité, finie de nombres réels tels que 0.5, sous la forme signe $\times$ base$^{exposant} \times (1 + mantisse)$. Il est en revanche impossible de représenter le reste des nombres réels, tels que 0.1. De ce fait, lorsqu'on se place dans des contextes de calculs, il devient évident qu'on peut se retrouver à accumuler des erreurs de précision qui vont venir fausser les résultats. Quand on veut garantir des résultats, par exemple sur des problèmes d'optimisation, cela peut devenir pénalisant.

3.1 Les intervalles

3.1.1 Intervalle et boite

Définition 1. *On définit un intervalle* $[\underline{x}, \overline{x}]$ *comme l'ensemble des nombres réels* x *tels que* $\underline{x} \leq x \leq \overline{x}$.

On note par la suite plus généralement $[x] = [\underline{x}, \overline{x}]$.

Exemple 1. *Si on veut représenter le nombre* $\sqrt{2} = 1.4142\ldots$, *on peut dire :* $1.4 \leq sqrt(2) \leq 1.5$, *donc encadrer ce nombre par l'intervalle* $[1.4, 1.5]$.

3.1.2 Fonctions d'inclusion

Avec cette notion d'intervalle, on peut définir le comportement quand on applique une fonction. L'idée est de se dire que, pour un intervalle d'entrée $[x]$, l'intervalle image par une fonction f doit contenir l'ensemble des images prises par la fonction f pour tout les $x \in [x]$:

Définition 2. *Soit $f : \mathbb{R}^n \to \mathbb{R}^m$ une fonction, la fonction $[f] : \mathbb{R}^n \to \mathbb{R}^m$ est une* ***fonction d'inclusion*** *pour f si*

$$\forall [x] \in \mathbb{R}^n, f([x]) \subset [f]([x]) \tag{3.1}$$

Exemple 2. *La figure 3.1 montre l'encadrement d'une fonction $y = f(x)$ quelconque. La boite bleue est le produit cartésien de $[x]$ et de $[y]$.*

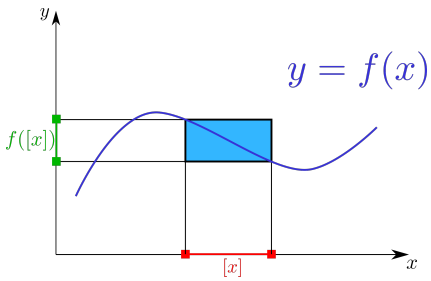


FIGURE 3.1 – Fonction d'inclusion

L'encadrement nécessite la connaissance précise de la forme de la fonction pour pouvoir l'encadrer correctement. Ceci peut se révéler compliquer pour des fonctions non évidentes, typiquement quand on monte en dimension. Pour cela, on peut combiner les fonctions d'inclusion sans perdre la garantie d'inclusion, comme indiqué dans le théorème 1.

Théorème 1. *si $[f]$ et $[g]$ sont des fonctions d'inclusion respectives pour f et g, alors $[f] \circ [g]$ est une fonction d'inclusion pour $f \circ g$.*

Cela permet en pratique de construire des fonctions d'inclusions élémentaires puis de les combiner. En effectuant cette opération, on peut en revanche perdre de la précision sur l'encadrement comme le montre la figure 3.2.

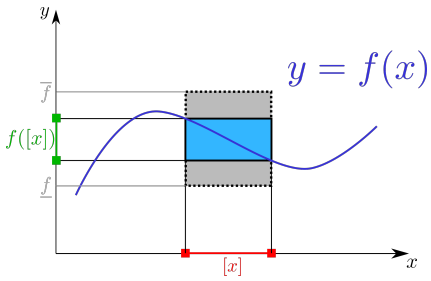


FIGURE 3.2 – Fonction d'inclusion composée de moindre qualité

3.1.3 Arithmétique élémentaire

Comme dit précédement, on peut construire les fonctions d'inclusions des fonctions necessaires pour n'importe quel problème en calcul par intervalle. Spécifiquement, il est utile de définir un certain nombre de fonctions de base permettant de former les blocs de construction pour la formation de fonctions plus complexes. On peut ainsi définir les opérateurs binaires (l'addition, la soustraction, la multiplication, ...) ainsi que les opérateurs unaires (l'exponentielle, la puissance, le sinus, ...).

Exemple 3. *Un certain nombre de fonctions arithmétiques élémentaires peuvent être formulées :*

- $[x_1] + [x_2] = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}]$
- $[x_1] - [x_2] = [\underline{x_1} - \overline{x_2}, \overline{x_1} - \underline{x_2}]$
- $[x_1] \times [x_2] = [min(\underline{x_1 x_2}, \underline{x_1}\overline{x_2}, \overline{x_1}\underline{x_2}, \overline{x_1 x_2}), max(\underline{x_1 x_2}, \underline{x_1}\overline{x_2}, \overline{x_1}\underline{x_2}, \overline{x_1 x_2})]$
- $[x]^2 = [0, max(\underline{x}^2, \overline{x}^2)]$
- $e^{[x]} = [e^{\underline{x}}, e^{\overline{x}}]$
- ...

On peut étendre ces définitions à l'ensemble des fonctions strictements monotones : il est évident de se dire que, si une fonction $f(x)$ est strictement croissante, alors $[f]([x]) = [f(\underline{x}), f(\overline{x})]$, et de même pour une fonction décroissante. On peut alors construire des fonctions plus complexes, comme $f : x \mapsto x^3$ en découpant la définition de la fonction par morceaux.

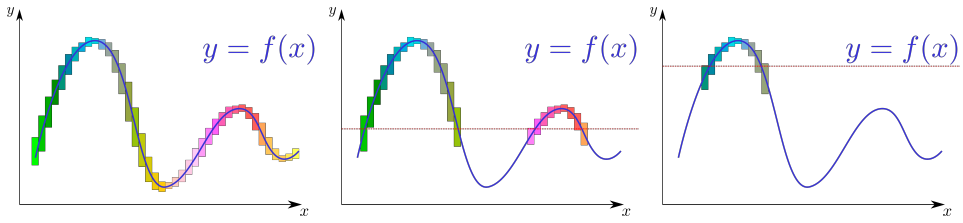
3.2 Optimisation avec les intervalles

On met en place un algorithme d'optimisation utilisant le calcul par intervalle pour obtenir un encadrement garanti de la solution à notre problème.

On veut résoudre le problème $\max(f(x))$ tel que $g(x) \leq 0$, avec f fonction coût et g ensemble des contraintes. Avec le calcul par intervalle, on cherche à avoir un encadrement *garanti* de la solution au problème. Le principe de base est de découper l'ensemble des entrées en un certain nombre de boites, dépendant de la précision que l'on veut, comme à la figure 3.3a. On choisit ensuite un a solution admissible du problème suivant le théorème 2.

Théorème 2. *Soit un a une solution admissible du problème $\max_x f(x)$ tel que $g(x) \leq 0$ et x^* la solution optimale, on a :*

$$sup([f]([x])) \leq f(a) \Rightarrow x^* \notin [x] \tag{3.2}$$



(a) (b) (c)

FIGURE 3.3 – Optimisation naïve

La figure 3.4 présente l'algorithme utilisé pour ce problème de maximisation (la logique étant la même pour une minimisation). Quatre éléments sont important à comprendre dans cet algorithme.

Data: $[I_{init}]$ initial search box ; ϵ stop criterion ; f cost function ; g constraints function ;
Output: $[f]$ bounds of best solution ; $[I]$ solution box
begin
  `solutions` list of solution boxes;
  Add $[I_{init}]$ to `solutions`;
  $[f_c]$ current bounds of solutions;
  while $\overline{f_c} - \underline{f_c} \geq \epsilon$ **do**
    `currentSolutions` empty list of boxes;
    `/* Bisect, evaluate cost, manage constraints */`
    forall *`sol` in `solutions`* **do**
      [`left`], [`right`] ⟵ `bisect(sol)`;
      if *g*([*`left`*]) *is valid* **then**
        Add [`left`] to `currentSolutions`;
      end
      if *g*([*`right`*]) *is valid* **then**
        Add [`right`] to `currentSolutions`;
      end
    end
    `/* Remove boxes certified not to contain maximum */`
    Evaluate $[f]$ for all [`currentSolutions`];
    f_{best} best f(`sol.mid`) in all $[f]$;
    Remove all [`sol`] in [`currentSolutions`] where
     $sup([f]([sol])) \leq f_{best}$;
    `solutions` ⟵ `currentSolutions`
  end
  return *`solutions`*, $[f_c]$
end

FIGURE 3.4 – Algorithme de maximisation par le calcul par intervalle

Chapitre 4

Construction d'un détecteur optimal

Presentation problème + résolution

4.1 Formulation du problème

Le problème de la détection d'état quantique porte sur un ensemble de m états quantiques représentés par les opérateurs densité $\{\rho_i \ , \ 1 \leq i \leq m\}$ munis des probabilités préalables $\{p_i \geq 0 \ , \ 1 \leq i \leq m\}$. L'objectif est d'obtenir un ensemble de m opérateurs de mesure $\{\Pi_j \ , \ 1 \leq i \leq m\}$ permettant de mesurer le mieux possible par rapport aux probabilités les états d'entrée qui nous arrivent.

Les opérateurs ρ_i et Π_i sont des matrices Hermitiennes semi-définies positives, de la forme $\begin{bmatrix} a & b+ic \\ b-ic & d \end{bmatrix}$.

Plusieurs critères ont été proposés à optimiser afin de construire ces détecteurs optimaux. D'une part, on a la possibilité de travailler sur la minimisation de l'erreur de mesure (l'erreur moyenne ou l'erreur quadratique). D'autre part, et c'est ce sur quoi nous avons travaillé, on peut considérer le critère de l'information mutuelle comme critère à maximiser. Ce critère indique la dépendance de deux variables aléatoires entre elles, il permet dans notre cas de bien caractériser la quantité d'information qu'on peut retirer des états d'entrée en ayant les opérateurs de mesure

L'information mutuelle de deux variables aléatoires X et Y est donnée par :

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p_{(X,Y)}(x,y) \log \Big(\frac{p_{(X,Y)}(x,y)}{p_X(x) p_Y(y)} \Big), \tag{4.1}$$

mais peut aussi être écrite en fonction des entropies des variables aléatoires :

$$I(X;Y) = H(X) - H(X|Y) \tag{4.2}$$
$$= H(Y) - H(Y|X) \tag{4.3}$$
$$= H(X) + H(Y) - H(X,Y). \tag{4.4}$$

Avec $H(X)$ entropie marginale de X, $H(Y)$ entropie marginale de Y, $H(X|Y)$ entropie conditionelle de X sachant Y et enfin $H(X,Y)$ entropie jointe de X et Y. On peut utiliser indifférement $\log_2$, $\log_{10}$ ou ln pour le logarithme, le changement étant à une constante près. On utilise par la suite le logarithme base exponentielle pour l'ensemble des calculs.

Dans le cas classique, les entropies marginales, conditionnelles et jointes sont définies par :

$$H(X) = -\sum_{x\in X} p(x)\log(p(x)), \tag{4.5}$$
$$H(Y) = -\sum_{y\in Y} p(y)\log(p(y)), \tag{4.6}$$
$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log(p(x,y)), \tag{4.7}$$
$$H(Y|X) = -\sum_{x\in X, y\in Y} p(x,y)\log\Big(\frac{p(x,y)}{p(x)}\Big) \tag{4.8}$$

Dans le cas quantique, les formules restent les mêmes, mais on exprime les probabilités des variables en fonction des valeurs des états quantiques d'entrée.

En fonction d'un état d'entrée ρ_i de probabilité préalable p_i, et d'un opérateur de sortie Π_i, on peut définir leur probabilité jointe :

$$p(X = \rho_i, Y = \Pi_i) = p_i \operatorname{tr}(\rho_i \Pi_i). \tag{4.9}$$

On en déduit les probabilités marginales :

$$p(X = \rho_i) = \sum_j p_i \operatorname{tr}(\rho_i \Pi_j) \tag{4.10}$$
$$p(Y = \Pi_j) = \sum_i p_i \operatorname{tr}(\rho_i \Pi_j), \tag{4.11}$$

Et les probabilités conditionelles :

$$P(Y = \Pi_j | X = \rho_i) = \frac{\mathrm{tr}(\rho_i \Pi_j)}{\sum_k \mathrm{tr}(\rho_i \Pi_k)} \tag{4.12}$$

L'information mutuelle pour notre problème peut donc être ré-écrite de la façon suivante, en utilisant $\alpha_{ij} = \mathrm{tr}(\rho_i \Pi_j)$:

$$\begin{aligned} I(\rho; \Pi) &= H(\rho) + H(\Pi) - H(\rho, \Pi) \\ &= -\sum_{i=1}^{m} \Big(\sum_{j=1}^{m} \alpha_{ij} \Big) \log \Big(\sum_{j=1}^{m} \alpha_{ij} \Big) - \sum_{i=1}^{m} \Big(\sum_{j=1}^{m} \alpha_{ji} \Big) \log \Big(\sum_{j=1}^{m} \alpha_{ji} \Big) + \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_{ij} \log(\alpha_{ij}) \end{aligned} \tag{4.13}$$

On peut aussi exprimer l'information mutuelle en fonction de l'entropie conditionnelle, mais il est plus efficace d'utiliser celle donnée à l'équation 4.13 pour la résolution numérique.

Le problème se formule comme un problème de maximization de l'information mutuelle : on cherche à maximiser l'information qu'on peut obtenir sur ρ_i quand on a les opérateurs de mesure Π_i :

$$\max_{\Pi} I(\rho, \Pi) \tag{4.14}$$

tel que :

$$\Pi_j \succeq 0 \quad 1 \leq j \leq m \tag{4.15}$$

$$\sum_{j=1}^{m} \Pi_j = I \tag{4.16}$$

La contrainte 4.15 correspond à la semi-définition positive des opérateurs de mesure Π_j. Enfin, la contrainte 4.16 permet d'obtenir des opérateurs de mesure cohérents pour que les probabilités de mesure $p(j) = \mathrm{tr}(\rho \Pi_j)$ soient positives et se somment à 1.

On est en présence d'une fonction convexe, et les contraintes engendrent un ensemble admissible convexe. C'est le cas idéal lors d'une minimisation, mais le problème est une maximisation, de même difficulté qu'une minimisation concave, on ne peut donc pas juste faire une descente de gradient pour le résoudre. On peut utiliser un certain nombre de méthodes approximatives, nous utilisons le calcul par intervalle afin d'obtenir un résultat sûr dans un intervalle.

4.2 Convexité de l'information mutuelle

Davies considère en 1978 que l'information mutuelle pour ce problème peut être considérée comme étant convexe, simplifiant la résolution du problème en ayant à chercher le maximum sur les bords. On s'intéresse ici à l'étude de cette convexité.

Dans son article, Davies regroupe les traces et probabilités sous une seule variable $P_{ij} = p_i \operatorname{tr}(\rho_i \Pi_j)$. Ces coefficients P_{ij} forment une matrice des probabilités, telle que :

$$\sum_{ij} P_{ij} = 1, \tag{4.17}$$

$$\sum_{i} P_{ij} = p_i. \tag{4.18}$$

L'information mutuelle s'écrit donc :

$$I(P) = \sum_i H(\sum_j P_{ij}) + \sum_j H(\sum_i P_{ij}) - \sum_{ij} H(P_{ij}) \tag{4.19}$$

La fonction $H(x) = -x \log(x)$ est convexe, et donc I est convexe par rapport à la matrice des probabilités P. La figure 4.1 illustre cette fonction en fixant $p_1 = 0.3$ et $p_2 = 0.7$.

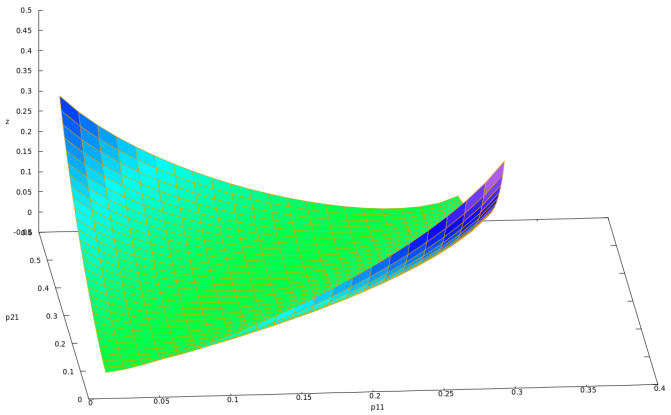


FIGURE 4.1 – Information mutuelle par rapport à la matrice de probabilités

La convexité semble bien vraie par rapport à P, mais on cherche à optimiser les matrices Π_j. La matrice P comporte les traces de la multiplication

$\rho_i \Pi_j$, qui est linéaire par rapport aux coefficients de Π_j. Si la fonction $I(P)$ est convexe par rapport à P, alors elle l'est par rapport aux Π_j, grace à la linéarité.

Quand on trace la même fonction mais par rapport aux variables $\Pi_{k_{ij}}$, en se fixant dans un espace 2 dimensions, on s'apperçoit que la fonction n'est pas correctement définie sur les bords. Ceci est dû au fait que $x \log(x)$ n'est pas définit pour $x < 0$, ce qui fausse ou bloque les calculs, suivant l'implémentation. Le détail de notre implémentation est expliqué en **??**.

4.3 Formulation des contraintes

La définition du problème permet de résoudre notament les cas immédiats des opérateurs de densité orthogonaux, mais la résolution devient très lente lorsqu'on passe à d'autres cas non orthogonaux. On rajoute des conditions au problème pour accélérer la résolution.

Le premier élément à simplifier est l'expression de l'entropie marginale de $X = \rho_i$. En effet, nous l'avons exprimé en fonction de la trace de la multiplication matricielle, mais on peut reprendre la définition donnée lors du cas classique qui dit que $H(X) = -\sum_{x \in X} p(x) \log(p(x))$. Le problème nous indique que nous connaissons les probabilités préalables des états d'entrée, on peut donc directement exprimer cette entropie en fonction de ces données et donc sans les variables de sortie Π_i.

Ensuite, on sait que les opérateurs de mesure se somment à l'identité. Cela signifie d'une part qu'on peut passer d'un problème à m matrices à un problème à $m-1$ matrices pour $m \geq 2$. Les matrices étant carrées de dimension n, on passe de $m \times n^2$ variables à $(m-1) \times n^2$ variables, ce qui est non négligeable.

De plus, le problème et les contraintes sont symmétriques, on peut intervertir les Π_j sans influencer le résultat de la fonction coût. Cela nous permet de couper le problème au moins en deux pour réduire à nouveau le temps de calcul. Du fait de la somme à l'identité, on peut ajouter en contrainte que $\Pi_{1_{1,1}} \leq \frac{1}{m}$ pour m opérateurs de mesure, puis $\Pi_{2_{1,1}} \leq \frac{1}{m-1}$, etc.

Enfin, pour rappel, les opérateurs de mesure sont des opérateurs densité, qui ne sont pas necessairement purs. Pour qu'ils soient purs, il faudrait entre autre que $\mathrm{tr}(\Pi_i) = 1$. On peut considérer qu'on restreint le problème à un cas purs, et dans ce cas rajouter la contrainte que la somme des éléments diagonaux des opérateurs de mesure doit sommer à 1. Cela permet soit de retirer une variable par matrice densité au problème, en l'exprimant par

$x_{n+1} = 1 - \sum_{i}^{n} x_i$ avec les x_i éléments diagonaux de l'opérateur densité, ce qui nécessite une reformulation du problème, soit l'ajout de la contrainte.

4.4 Résolution avec `ibex`

Pour la résolution de ce problème, utilisons la librairie `ibex` permettant de faire du calcul par intervale, et possède entre autres un outil d'optimisation, `ibexopt`. Le problème est formulé avec un langage dédié, `minibex`. Nous avons eu besoin de définir un opérateur additionnel à ceux présents, l'opérateur `xlog` permettant d'effectuer l'opération $x \times \log(x)$ en redéfinissant $0 \times \log(0) = 0$ pour que les intervalles ne tombent pas à l'infiniquand ils contiennent 0. De plus, `minibex` ne considère que des problèmes de minimisation, on ré-écrit le problème en prenant la fonction coût opposée : $\max f(x) \Leftrightarrow \min -f(x)$.

Le premier test effectué est sur le cas de deux états d'entrée $|\psi_1\rangle = |0\rangle$ et $|\psi_2\rangle = |1\rangle$ ayant pour probabilité respective $p_1 = 0.1$ et $p_2 = 0.9$. Le résultat théorique est connu : les états étant orthogonaux, on doit obtenir les opérateurs de mesure égaux aux opérateurs densité d'entrée. On obtient bien avec `ibex` les opérateurs suivant :

$$\Pi_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \Pi_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$

qui correspondent bien à deux opérateurs de mesure orthogonaux. Dans ce cas, l'information mutuelle est comprise dans l'intervalle $I(\rho, \Pi) \in [0.3250, 0.3254]$, avec un temps de calcul de 8.6 millisecondes.

Le deuxième cas qu'on peut présenter est le suivant : $|\psi_1\rangle = |0\rangle$ et $|\psi_1\rangle = |+\rangle$ avec comme probabilité respectives $p_1 = 0.1$ et $p_2 = 0.9$. Le résultat théorique n'est pas donné, et on obtient avec `ibex` le résultat suivant :

$$\Pi_1 = \begin{bmatrix} 0.445 & 0.497 \\ 0.497 & 0.555 \end{bmatrix}, \quad \Pi_2 = \begin{bmatrix} 0.555 & -0.497 \\ -0.497 & 0.445 \end{bmatrix},$$

avec une information mutuelle comprise dans l'intervalle $I(\rho, \Pi) \in [0.1348, 0.1349]$, et un temps de calcul de 0.79 secondes.

En revanche, ibex ralentit fortement dès qu'on sort des cas simples présentés si-dessus, et notament quand on retire la restriction des opérateurs de mesure à des opérateurs densité purs (de trace unitaire). Deux éléments importants sont à l'origine du problème. Tout d'abord, en analysant l'utilisation des ressources cpu lors de la résolution d'un problème, on voit qu'un seul thread est utilisé par `ibexopt`. Les algorithmes d'optimisation peuvent être parallélisés, et dans le cas de processeurs à plusieurs cœurs on pourrait

avoir un gain de performance conséquent. Le deuxième élément est en lien avec la convexité de la fonction coût évoqué précédement. On eut alors se limiter aux extremités de la fonction coût pour réduire le nombre de calculs à effectuer. Il faudrait alors implémenter la condition $0 \in [\texttt{grad}]([f])$, ce qui n'est pas prévu de base dans `ibexopt`.

4.5 Exemple

Prenons le cas de deux états quantiques :

$$|\psi_1\rangle = \begin{bmatrix} \frac{1}{3} \\ \frac{2\sqrt{2}}{3} \end{bmatrix}, \quad |\psi_2\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \tag{4.20}$$

avec les probabilités préalables

$$p(|\psi_1\rangle) = 0.1, \quad p(|\psi_2\rangle) = 0.9. \tag{4.21}$$

Ces deux états peuvent être réécris sous la forme d'opérateurs densité $\rho_1 = \begin{bmatrix} \frac{1}{9} & \frac{2\sqrt{2}}{9} \\ \frac{2\sqrt{2}}{9} & \frac{8}{9} \end{bmatrix}$ et $\rho_2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$.

Le problème est de trouver les deux opérateurs densité optimaux au sens de l'information mutuelle. On considère le problème comme ne possédant pas de termes complexes sur les antidiagonaux, et l'équation 4.16 nous permet de réduire le problème à trois variables seulement $\{a_1, b_1, d_1\}$:

$$\Pi_1 = \begin{bmatrix} a_1 & b_1 + ic_1 \\ b_1 - ic_1 & d_1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 \\ b_1 & d_1 \end{bmatrix} \tag{4.22}$$

$$\Pi_2 = I - \Pi_1 = \begin{bmatrix} 1 - a_1 & -b_1 \\ -b_1 & 1 - d_1 \end{bmatrix} \tag{4.23}$$

On pose ensuite les contraintes sur ces variables. Tout d'abord, ces variables sont définies sur ces bornes spécifiques : $a_1 \in [0, 0.5]$, $b_1 \in [-1, 1]$, $d_1 \in [0, 1]$. La détermination de la semi-définition positive passe par les diagnonales et le déterminant strictement positifs, d'une part avec les bornes précédentes et d'autre part avec deux nouvelles contraintes sur les 3 variables. Le problème s'écrit donc :

$$\max_{a_1, b_1, d_1} I(a_1, b_1, d_1),$$

tel que :

$$
\begin{aligned}
&a_1 \in [0, 0.5], b_1 \in [-1, 1], d_1 \in [0, 1], \\
&a_1 * d_1 - b_1^2 \geq 0, \\
&(1 - a_1) * (1 - d_1) - b_1^2 \geq 0.
\end{aligned}
$$

La résolution avec `ibex` ou avec notre solveur donne les deux opérateurs de mesure suivants :

$$
\Pi_1 = \begin{bmatrix} 0.456 & -0.498 \\ -0.498 & 0.544 \end{bmatrix}, \quad \Pi_2 = \begin{bmatrix} 0.544 & 0.498 \\ 0.498 & 0.456 \end{bmatrix}, \tag{4.24}
$$

Avec une information mutuelle $I(a_1, b_1, d_1) = 0.04723$. Notre solveur résout le problème en 6.4 secondes pour une précision sur I de 1×10^{-5}, et `ibexopt` résout en 88.3 secondes pour une précision sur I de 4×10^{-5}.

Chapitre 5

Conclusion

Ce travail bibliographique met ainsi en place les différents éléments necessaires à la compréhension de l'informatique quantique : la notion de système quantique, amenant la définition de qubit, les mécanismes permettant de faire évoluer les systèmes, ainsi que la mesure. On illustre de plus trois algorithmes majeurs dans l'histoire de ce domaine, en montrant leur champ d'application, les problèmes qu'ils permettent de résoudre. De tout ce travail découle la conclusion que l'informatique quantique permet d'accélérer considérablement la résolution d'un certain nombre de problèmes impossibles à réoudre avec les technologies classiques.

Annexe A

Création de circuits quantiques pour l'encodage de fonctions booléennes

On étudie ici la problématique de pouvoir construire systématiquement une fonction booléenne avec un ordinateur quantique, présenté par Younes et Miller en 2003 [6].

En informatique classique, l'ensemble des fonctions booléennes peuvent être décrites à l'aide des opérateurs **NAND** et **NOR**. Il s'agit donc de pouvoir les transcrire en quantique, et de pouvoir établir un système de combinaison de ces portes, pour permettre l'élaboration des circuits.

Dans ce modèle, on considère un registre de n qubits composant l'entrée du système, un registre de m qubits composant la sortie du système, et un registre de k qubits auxiliaires pour certaines opérations intermédiaires.

Pour cette construction, on se base sur la porte quantique **X** et ses équivalents composés **CNOT**, **CCNOT** (Toffoli), etc. On fourni alors un certain nombre de circuits de base pouvant être recomposés pour former des circuits plus complexes.

La compilation d'une fonction booléenne passe alors par 4 étapes :

1. Écriture de la table de vérité,
2. Pour chaque sortie donnant 1, former une porte **NOT** controlée. Chaque entrée va servir de contrôle, par 1 si l'entrée est à 1, et par 0 si l'entrée est à 0,
3. Développer le circuit résultant pour n'avoir que des portes **NOT** controlées par 0,
4. Simplifier le circuit en éliminant les doublons.

Étape 1 : établissement des premières portes contrôlées

La figure A.1 représente une porte **NOT** contrôlée. On note que les qubits de contrôle sont indiqués par • (contrôle par 1) et par ∘ (contrôle par 0). Le

dernier qubit est la cible (*target*). On effectue l'opération **NOT** sur la cible si et seulement si les bits de contrôles respectent leur condition (si il sont à 1 pour ceux qui sont contrôlés par 1, et si ils sont à 0 pour ceux contrôlés par 0).

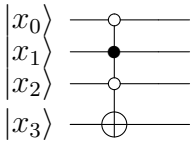


FIGURE A.1 – Porte **NOT** contrôlée

Cette porte en revanche ne peut pas être construite, on ne dispose en effet que des portes **NOT** contrôlées par 1 et pas de celles contrôlées par 0.

Exemple 4. *Soit la fonction booléenne* $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge \neg x_2) \vee (x_1 \wedge x_3)$. *Sa table de vérité est la suivante :*

x_1	x_2	x_3	$F(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

On a quatre sorties à 1. La figure A.2 représente donc le circuit initial qu'on obtient.

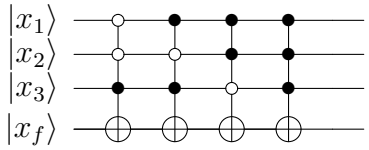


FIGURE A.2 – Circuit quantique pour $f(x_1, x_2, x_3)$

Étape 2 : Développement du circuit

La deuxième étape consiste à prendre le circuit obtenu précédemment et à le développer de façon à n'obtenir que des portes **NOT** contrôlées par 0. Dans le principe, une porte ayant un mélange de contrôle par 0 et par 1 va être équivalent à la combinaison des portes contrôlées par 0, qui vont avoir

des contrôles de moins en combinaison sur les contrôles par 1. Un exemple est plus clair pour comprendre. Reprenons la porte de la figure A.1. Elle est en fait équivalente au circuit A.3 :

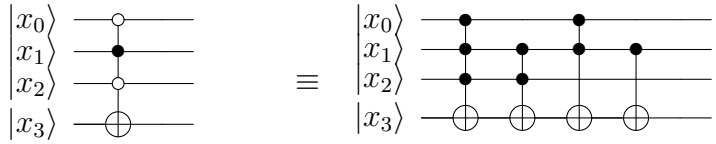


FIGURE A.3 – Équivalent sans contrôles par 0

Exemple 5. *On reprends notre exemple de fonction booléenne* $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge \neg x_2) \vee (x_1 \wedge x_3)$*. Une fois développée, son circuit équivalent est illustré à la figure A.4*

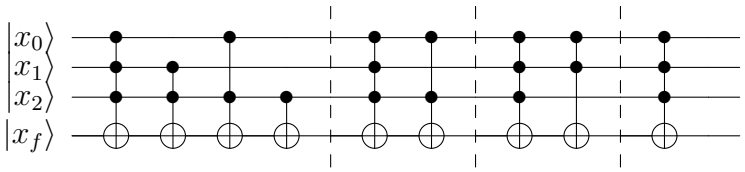
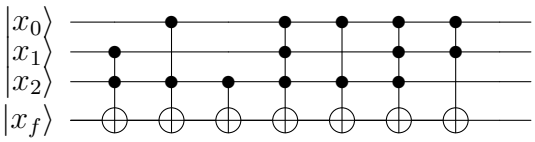


FIGURE A.4 – Circuit quantique développé pour $f(x_1, x_2, x_3)$

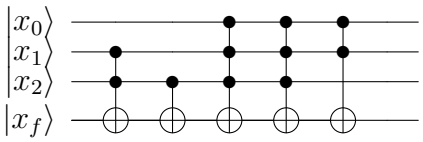
Étape 3 : Simplification du circuit

La dernière étape permet d'obtenir un circuit comportant le moins de portes possibles, en se basant sur le principe suivant : lorsqu'un circuit (ici composé de **CNOT**) est entouré de deux mêmes **CNOT**, alors celles-ci s'annulent et on peut alors enlever la paire doublon sans changer le résultat. En effectuant ce raisonnement récursivement, on arrive à obtenir un circuit minimal.

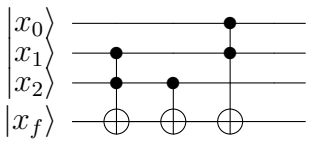
En reprenant l'exemple de la fonction booléenne précédente, on peut faire par étapes la simplification :



(a) Première simplification



(b) Deuxième simplification



(c) Dernière simplification

FIGURE A.5 – Simplifications successives pour $f(x_1, x_2, x_3)$

Extension à des circuits plus complexes

Avec cette méthode, il est facile de construire l'ensemble des fonctions booléennes à 2 bits, ainsi qu'a 3 bits. Pour construire des circuits plus complexes, on dispose alors de deux façons de procéder.

Tout d'abord, on peut refaire ces étapes sur la fonction booléenne plus complexe, et trouver un circuit minimal étant composé de n entrées, et d'un seul qubit auxiliaire pour la sortie.

En revanche, si on ne veut pas établir la table de vérité avant la construction du circuit, on peut utiliser les fonctions élémentaires et construire un circuit a partir de la chaine de caractère représentant la fonction. Il est important de noter que la méthode présentée ici ne modifie pas les entrées de la fonction, on peut donc les réutiliser comme on le souhaite pour venir greffer des fonctions supplémentaires au circuit. Cette méthode s'apparente algrithmiquement à la compilation des fonctions sur les ordinateurs classiques. En revanche, par rapport au circuit minimal qu'on pourrait trouver, on aura ici un qubit auxiliaire par sous-circuit, et un nombre de portes bien plus important. Pour des considérations de limitations matérielles qui sont pour le moment importantes, cette deuxième façon de procéder peut ne pas donner des résultats implémentables pour de trop grosses fonctions.

Bibliographie

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge : Cambridge University Press, 2000.

[2] D. N. Mermin, *Quantum Computer Science : An introduction*. Cambridge : Cambridge University Press, 2007.

[3] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, pp. 3457–3467, 1995.

[4] Y. Shi, "Both Toffoli and Controlled-NOT need little help to do universal quantum computation," 2002.

[5] E. Moore, R, "Interval analysis," *Science*, vol. 158, no. 3799, pp. 365–365, 1967.

[6] A. Younes and J. Miller, "Automated method for building cnot based quantum circuits for boolean functions," 05 2003.

Résumé — Ce rapport bibliographique présente les différents éléments de base nécessaires à la compréhension de l'informatique quantique. On y présente en premier les concepts fondateurs de l'informatique quantique : le qubit, les mécanismes d'évolution des systèmes quantiques et enfin la mesure projective. On y montre ensuite trois algorithmes fondateurs en illustrant les problèmes résolus et leur fonctionnement. Ces trois algorithmes permettent de montrer l'apport de l'informatique quantique, ils permettent en effet d'obtenir des performances de traitement de l'information inaccessibles en classique.

Mots clés : Informatique quantique, qubit, mesure projective, Deutsch-Jozsa, Grover, Shor, parallélisation.

Abstract — This bibliographic report presents the fundamental aspects of quantum computing. We firstly show 3 concepts required to understand quantum computing : the notion of qubit, the mechanisms of evolution of quantum systems and the projective measurement of qubits. We then show three major quantum algorithms (Deustch-Jozsa, Grover and Shor), explaining the problems they solve and how they work. With this report, we understand how quantum computing improves the solving of a certain category of problems, via parallelism.

Keywords : Quantum computing, qubit, projective measurement, Deutsch-Jozsa, Grover, Shor, parallelism.

Polytech Angers
62, avenue Notre Dame du Lac
49000 Angers