



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
16 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

1	Introduction	1
2	Informatique quantique : éléments de base	2
2.1	Postulats de base	2
2.1.1	Etat d'un système quantique	2
2.1.2	Mesure projective	3
2.1.3	Dynamique du système	3
3	Calcul par intervalle : éléments de base	5
3.1	Les intervalles	5
3.1.1	Intervalle et boîte	5
3.1.2	Fonctions d'inclusion	5
3.1.3	Arithmétique élémentaire	7
3.2	Optimisation avec les intervalles	7
3.3	Implémentation	9
4	Construction d'un détecteur optimal	11
4.1	Formulation du problème	11
4.2	Convexité de l'information mutuelle	14
4.3	Formulation des contraintes	15
4.4	Résolution avec ibex	16
4.5	Exemple	17
5	Conclusion	19
A	Création de circuits quantiques pour l'encodage de fonctions booléennes	20
	Bibliographie	24

Table des figures

3.1	Fonction d'inclusion	6
3.2	Fonction d'inclusion composée de moindre qualité	6
3.3	Optimisation naïve	8
3.4	Algorithme de maximisation par le calcul par intervalle	9
4.1	Information mutuelle par rapport à la matrice de probabilités	14
A.1	Porte NOT contrôlée	21
A.2	Circuit quantique pour $f(x_1, x_2, x_3)$	21
A.3	Équivalent sans contrôles par 0	22
A.4	Circuit quantique développé pour $f(x_1, x_2, x_3)$	22
A.5	Simplifications successives pour $f(x_1, x_2, x_3)$	23

Chapitre 1

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Leo vel orci porta non. Tincidunt ornare massa eget egestas purus viverra accumsan in nisl. Tristique senectus et netus et malesuada fames ac turpis egestas. Varius quam quisque id diam vel quam. Rhoncus urna neque viverra justo nec ultrices. Est velit egestas dui id ornare. Arcu non odio euismod lacinia. Ante metus dictum at tempor. Vulputate sapien nec sagittis aliquam malesuada bibendum arcu vitae elementum. Non nisi est sit amet facilisis magna etiam. Quis commodo odio aenean sed adipiscing diam. Etiam erat velit scelerisque in. Tortor id aliquet lectus proin nibh nisl condimentum id. At auctor urna nunc id cursus metus aliquam.

Nec sagittis aliquam malesuada bibendum arcu vitae. Amet nisl purus in mollis. Ac tincidunt vitae semper quis lectus. Ac felis donec et odio pellentesque diam volutpat commodo sed. Ipsum dolor sit amet consectetur adipiscing elit pellentesque. Tortor vitae purus faucibus ornare suspendisse sed nisi. Tristique risus nec feugiat in fermentum posuere urna nec tincidunt. Egestas purus viverra accumsan in nisl nisi scelerisque eu. Ac odio tempor orci dapibus ultrices. Molestie ac feugiat sed lectus vestibulum mattis ullamcorper velit. Interdum velit laoreet id donec ultrices. Donec adipiscing tristique risus nec feugiat in fermentum posuere urna. Porttitor leo a diam sollicitudin. Cras sed felis eget velit aliquet sagittis. Ut etiam sit amet nisl purus in. Viverra suspendisse potenti nullam ac tortor vitae purus faucibus ornare. Lorem ipsum dolor sit amet consectetur adipiscing elit ut aliquam.

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 :

$$|\psi\rangle = \sum_i c_i |k_i\rangle, \quad (2.1)$$

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 :

$$U^\dagger U = U U^\dagger = I, \quad (2.2)$$

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.

Chapitre 3

Calcul par intervalle : éléments de base

Le calcul par intervalle est décrit au départ dans les travaux de Ramon Moore [3]. 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 $\text{signe} \times \text{base}^{\text{exposant}} \times (1 + \text{mantisse})$. 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 boîte

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

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

Exemple 1. Si on veut représenter le nombre $\sqrt{2} = 1.4142\dots$, on peut dire : $1.4 \leq \text{sqrt}(2) \leq 1.5$, 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ une fonction, la fonction $[f] : \mathbb{R}^n \rightarrow \mathbb{R}^m$ est une **fonction d'inclusion** pour f si

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

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

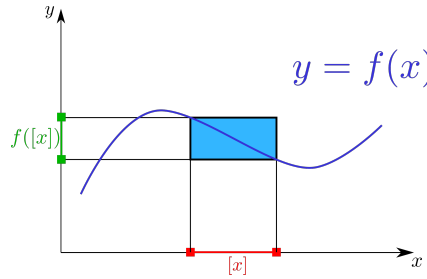


FIGURE 3.1 – Fonction d'inclusion

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.

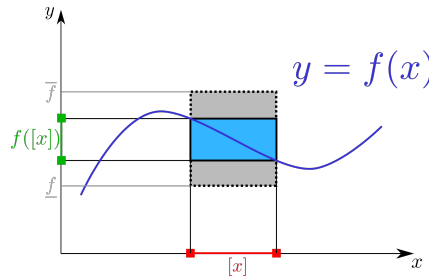


FIGURE 3.2 – Fonction d'inclusion composée de moindre qualité

3.1.3 Arithmétique élémentaire

Comme dit précédemment, on peut construire les fonctions d'inclusions des fonctions nécessaires 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 :*

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

On peut étendre ces définitions à l'ensemble des fonctions strictement 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 boîtes, dépendant de la précision que l'on veut, comme à la figure 3.3a. On choisit ensuite une 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 :*

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

Cela nous permet d'éliminer directement de l'ensemble des solutions les boîtes dont la borne supérieure de l'image est inférieure à l'image de ce critère a , puisque garanties comme ne contenant pas l'optimum du problème. La figure 3.3b illustre cette élimination. À l'issue de cette étape, on voit qu'on obtient un ensemble plus restreint de boîtes garanties comme contenant la solution, et on peut itérer en choisissant au fur et à mesure un critère a meilleur, et on arrive à un encadrement satisfaisant de la solution comme à la figure 3.3c.

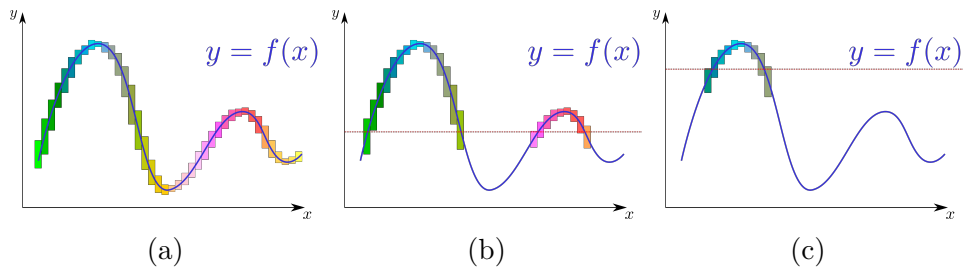


FIGURE 3.3 – Optimisation naïve

Cette méthode d'optimisation, "naïve", permet d'obtenir un résultat satisfaisant mais va être rapidement limitée si on veut des précisions plus élevées. En effet, on va avoir très rapidement un très grand nombre de boîtes à traiter. On peut remarquer entre autres qu'un certain nombre de boîtes vont être dans des "régions" globales pouvant être éliminées (par exemple, sur la figure 3.3b on a tout le bas de la fonction qui pourrait être éliminé d'un coup). Cela nous amène à un algorithme plus avancé, présenté en 3.4.

Au lieu de découper directement l'espace des entrées en un très grand nombre de boîtes, on va bisecter l'espace en deux boîtes, suivant l'axe le plus grand. On obtient deux boîtes, dont l'axe le plus grand aura été coupé en faisant $[x] \rightarrow [\underline{x}; (\bar{x} + \underline{x}).0.5], [(\bar{x} + \underline{x}).0.5; \bar{x}]$. On considère la bisection par le milieu, mais on pourrait aussi utiliser des bisections plus avancées permettant d'accélérer l'algorithme.

Une fois obtenues ces deux boîtes, on peut évaluer la fonction d'inclusion et les contraintes, et décider de jeter les boîtes ne rentrant pas dans les contraintes. Sur l'ensemble des boîtes obtenues pour une itération, on effectue l'opération de recherche de critère décrit précédemment, et on réitère sur les boîtes restantes. Cela permet d'éliminer rapidement les boîtes de grande dimension qui sont garanties ne comprenant pas l'optimum, et donc réduire considérablement le nombre de boîtes à traiter par la suite.

On considère l'algorithme fini quand on a obtenu une précision suffisante sur l'encadrement de la fonction ou des variables d'entrée, ou alors quand un

certain nombre d'itérations ont été effectuées.

```

Data:  $[I_{init}]$  initial search box;  $\epsilon$  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
             $[left], [right] \leftarrow \text{bisect}(sol);$ 
            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  $\leftarrow$  currentSolutions
    end
    return  $solutions, [f_c]$ 
end

```

FIGURE 3.4 – Algorithme de maximisation par le calcul par intervalle

3.3 Implémentation

L'algorithme 3.4 a été implémenté en C# (dotnet 5) sur la base de la librairie IntSharp modifiée pour répondre à nos besoins (rajout de l'intervalle vide, de la fonction $x \log(x)$, des intervalles booléens, ...). Une interface graphique basique utilisant Blazor a été mise en place pour faci-

liter la visualisation de l'optimisation et des différents problèmes rencontrés lors du développement. L'ensemble du projet est disponible sur <https://github.com/PierreEngelstein/IntervalEval>. La solution est organisée en plusieurs modules :

- `IntervalEval` qui fournit la librairie de base pour le calcul par intervalle et l'optimisation ;
- `IntervalEval.Front` est l'interface web basique développée avec le framework Blazor Server ;
- `IntervalEval.Optimizer` est une interface en ligne de commande pour le problème spécifique détaillé dans ce rapport ;
- `IntervalEval.FrontConsole` est une interface en ligne de commande codée pour tester le problème à trois états quantiques d'entrée, pour tester les performances sur un problème en plus haute dimension ;
- `IntervalEval.Tests` fournit un ensemble de tests unitaires pour le bon fonctionnement de la librairie d'intervalle.

Chapitre 4

Construction d'un détecteur optimal

On présente ici le détail du problème de la détection optimale quantique avec le critère de l'information mutuelle.

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 à priori $\{p_i \geq 0, 1 \leq i \leq m\}$. L'objectif est d'obtenir un ensemble de m opérateurs de mesure $\{\Pi_j, 1 \leq j \leq m\}$ 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 quadratique de mesure [4] ou la maximisation de la probabilité de détection correcte [5]. 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 [6]. 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 a été formulée par Shannon en 1948 [7]. Elle est donnée par :

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

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

$$I(X; Y) = H(X) - H(X|Y) \quad (4.2)$$

$$= H(Y) - H(Y|X) \quad (4.3)$$

$$= H(X) + H(Y) - H(X, Y). \quad (4.4)$$

Avec $H(X)$ entropie marginale de X , $H(Y)$ entropie marginale de Y , $H(X|Y)$ entropie conditionnelle de X sachant Y et enfin $H(X, Y)$ entropie jointe de X et Y . On peut utiliser indifféremment \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 :

$$H(X) = - \sum_{x \in X} p(x) \log(p(x)), \quad (4.5)$$

$$H(Y) = - \sum_{y \in Y} p(y) \log(p(y)), \quad (4.6)$$

$$H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log(p(x, y)), \quad (4.7)$$

$$H(Y|X) = - \sum_{x \in X, y \in Y} p(x, y) \log\left(\frac{p(x, y)}{p(x)}\right) \quad (4.8)$$

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 :

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

On en déduit les probabilités marginales :

$$p(X = \rho_i) = \sum_j p_i \operatorname{tr}(\rho_i \Pi_j) \quad (4.10)$$

$$p(Y = \Pi_j) = \sum_i p_i \operatorname{tr}(\rho_i \Pi_j), \quad (4.11)$$

Et les probabilités conditionnelles :

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

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

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

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 :

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

tel que :

$$\Pi_j \succeq 0 \quad 1 \leq j \leq m \quad (4.15)$$

$$\sum_{j=1}^m \Pi_j = I \quad (4.16)$$

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 $p(j) = \text{tr}(\rho \Pi_j)$ 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 $P_{ij} = p_i \text{tr}(\rho_i \Pi_j)$. Ces coefficients P_{ij} forment une matrice des probabilités, telle que :

$$\sum_{ij} P_{ij} = 1, \quad (4.17)$$

$$\sum_i P_{ij} = p_i. \quad (4.18)$$

L'information mutuelle s'écrit donc :

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

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 $p_2 = 0.7$.

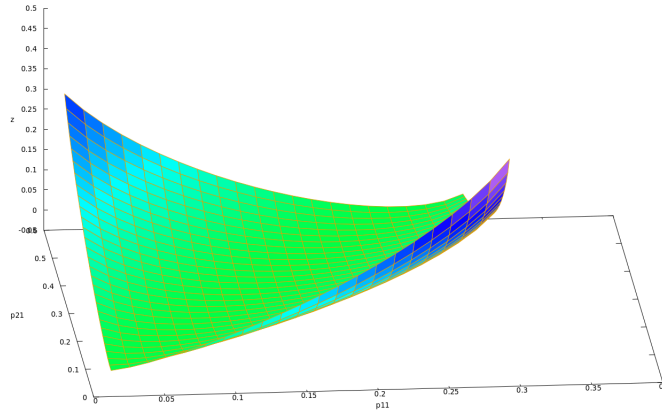


FIGURE 4.1 – Information mutuelle par rapport à la matrice de probabilités

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 , grâce 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'aperçoit que la fonction n'est pas correctement définie sur les bords. Ceci est dû au fait que $x \log(x)$ n'est pas défini 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 notamment 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 symétriques, 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 $\Pi_{1,1} \leq \frac{1}{m}$ pour m opérateurs de mesure, puis $\Pi_{2,1} \leq \frac{1}{m-1}$, etc.

Enfin, pour rappel, les opérateurs de mesure sont des opérateurs densité, qui ne sont pas nécessairement purs. Pour qu'ils soient purs, il faudrait entre autre que $\text{tr}(\Pi_i) = 1$. 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 intervalle, 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'infini quand 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 $|\psi_2\rangle = |+\rangle$ 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 notamment 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 précédemment. On eut alors se limiter aux extrémités de la fonction coût pour réduire le nombre de calculs à effectuer. Il faudrait alors implémenter la condition $0 \in [\mathbf{grad}](f)$, ce qui n'est pas prévu de base dans `ibexopt`.

4.5 Exemple

Prenons le cas de deux états quantiques :

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

avec les probabilités préalables

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

Ces deux états peuvent être réécrits 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 $\{a_1, b_1, d_1\}$:

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

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

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 diagonales 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 :

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

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

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Leo vel orci porta non. Tincidunt ornare massa eget egestas purus viverra accumsan in nisl. Tristique senectus et netus et malesuada fames ac turpis egestas. Varius quam quisque id diam vel quam. Rhoncus urna neque viverra justo nec ultrices. Est velit egestas dui id ornare. Arcu non odio euismod lacinia. Ante metus dictum at tempor. Vulputate sapien nec sagittis aliquam malesuada bibendum arcu vitae elementum. Non nisi est sit amet facilisis magna etiam. Quis commodo odio aenean sed adipiscing diam. Etiam erat velit scelerisque in. Tortor id aliquet lectus proin nibh nisl condimentum id. At auctor urna nunc id cursus metus aliquam.

Nec sagittis aliquam malesuada bibendum arcu vitae. Amet nisl purus in mollis. Ac tincidunt vitae semper quis lectus. Ac felis donec et odio pellentesque diam volutpat commodo sed. Ipsum dolor sit amet consectetur adipiscing elit pellentesque. Tortor vitae purus faucibus ornare suspendisse sed nisi. Tristique risus nec feugiat in fermentum posuere urna nec tincidunt. Egestas purus viverra accumsan in nisl nisi scelerisque eu. Ac odio tempor orci dapibus ultrices. Molestie ac feugiat sed lectus vestibulum mattis ullamcorper velit. Interdum velit laoreet id donec ultrices. Donec adipiscing tristique risus nec feugiat in fermentum posuere urna. Porttitor leo a diam sollicitudin. Cras sed felis eget velit aliquet sagittis. Ut etiam sit amet nisl purus in. Viverra suspendisse potenti nullam ac tortor vitae purus faucibus ornare. Lorem ipsum dolor sit amet consectetur adipiscing elit ut aliquam.

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 [8].

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 fournit 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** contrôlée. 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** contrôlées 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 \bullet (contrôle par 1) et par \circ (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).

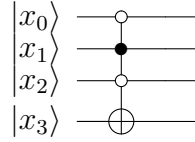


FIGURE A.1 – Porte **NOT** contrôlée

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.

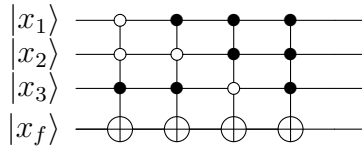


FIGURE A.2 – Circuit quantique pour $f(x_1, x_2, x_3)$

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 :

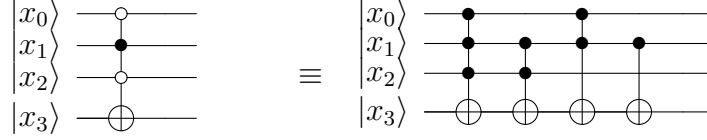


FIGURE A.3 – Équivalent sans contrôles par 0

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

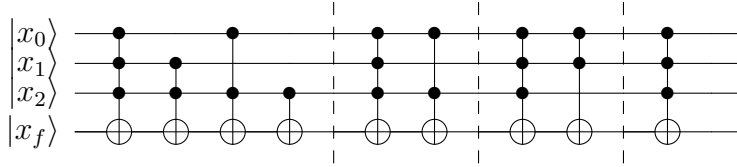


FIGURE A.4 – Circuit quantique développé pour $f(x_1, x_2, x_3)$

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 :

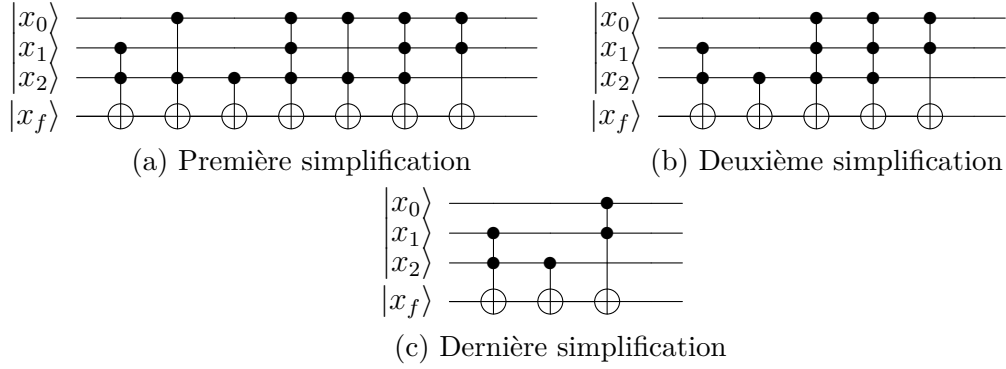


FIGURE A.5 – Simplifications successives pour $f(x_1, x_2, x_3)$

Extension à des circuits plus complexes

Avec cette méthode, il est facile de construire l'ensemble des fonctions booléennes à 2 bits, ainsi qu'à 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 à partir de la chaîne 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 algorithmiquement 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] E. Moore, R, “Interval analysis,” *Science*, vol. 158, no. 3799, pp. 365–365, 1967.
- [4] Y. C. Eldar and G. D. Forney, “On quantum detection and the square-root measurement,” *IEEE Transactions on Information Theory*, vol. 47, pp. 858–872, 2001.
- [5] Y. C. Eldar, A. Megretski, and G. C. Verghese, “Designing optimal quantum detectors via semidefinite programming,” *IEEE Transactions on Information Theory*, vol. 49, pp. 1007–1012, 2003.
- [6] E. B. Davies, “Information and quantum measurement,” *IEEE Transactions on Information Theory*, vol. 24, pp. 596–599, 1978.
- [7] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.
- [8] 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 (Deutsch-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