



PROJET SCIENTIFIQUE COLLECTIF

**Étude algorithmique et implémentation sur un ordinateur
quantique de la résolution d'un système d'équations linéaires**

Tuteur : Georges UZBELGER Coordinateur : Teddy PICHARD

Leder Yhivert AGUIRRE RAMIREZ,
Nicolas ESPINA QUILODRAN, Elouan LIMOUSIN,
Thomas POCHART, Simon RICHOUX



CONTENTS

0.1	Introduction	4
1	Notions d'informatique quantique	5
1.1	Définitions et notations	5
1.1.1	Stockage élémentaire de l'information : le qubit	5
1.1.2	Stockage avancé de l'information : registres de qubits	5
1.1.3	Manipulation de l'information : portes quantiques	6
1.1.4	Acquisition de l'information : mesure de qubits	8
1.1.5	Traitement de l'information : algorithmes quantiques	9
1.2	Quelques particularités de l'informatique quantique	9
1.2.1	Réversibilité des algorithmes avant mesure	9
1.2.2	Parallélisme quantique	11
1.2.3	Des algorithmes à lancer un grand nombre de fois	11
1.3	Complexité des algorithmes quantiques	12
1.4	Un peu de physique : de quoi est fait un ordinateur quantique ?	13
1.5	Résumé : ordinateur classique vs ordinateur quantique	13
2	L'algorithme HHL : fonctionnement et implémentation	14
2.1	Position du problème	14
2.1.1	Préliminaire : quelques définitions mathématiques	14
2.1.2	Le problème LSP	14
2.1.3	Le problème QLSP	14
2.2	HHL	15
2.2.1	Première étape : Chargement des données	16
2.2.2	Deuxième étape : <i>QPE</i>	16
2.2.3	Troisième étape : <i>EIGROT</i>	17
2.2.4	Quatrième étape : Uncompute	17
2.2.5	Cinquième étape : Mesures	17
2.2.6	Vue d'ensemble de l'algorithme	18
2.2.7	Un exemple détaillé	18
2.3	Fonctionnement et implémentation de qRAM loading	18
2.4	Fonctionnement et implémentation de QPE	22
2.4.1	Simulation hamiltonienne	22
2.4.2	QFT	22
2.4.3	Vue d'ensemble de l'algorithme	22
2.4.4	Un exemple détaillé	26
2.5	Fonctionnement et implémentation de EIGROT	26
2.5.1	Arithmétique quantique	26
2.5.2	ASNINV	26
2.5.3	Rotation contrôlée	26
2.5.4	Vue d'ensemble de l'algorithme	26
2.5.5	Un exemple détaillé	26
2.6	Mesures	26
3	Analyse de HHL	27



4	Démarche de projet	27
4.1	Travail à plusieurs	27
4.2	Répartition des tâches au sein de l'équipe	27
4.3	Définition d'un framework adapté pour travailler à plusieurs sur l'implémentation de HHL	28
A	Portes quantiques	29
A.1	Définitions et notations	29
A.1.1	Action de $H^{\otimes n}$ sur $ x\rangle_n$	29
A.1.2	L'ensemble standard	29
A.1.3	Décomposition de portes	30

0.1 INTRODUCTION

L'informatique permet à l'humanité de mener à bien des calculs autrement trop gros ou trop nombreux : en ce sens, elle a permis d'extraordinaires révolutions technologiques s'étant opérées en l'espace de moins d'un siècle. Parmi les calculs critiques pour nombre de problèmes scientifiques, on peut citer la résolution de systèmes d'équations linéaires, dont on compte des applications dans tous les domaines. Résoudre en des temps acceptables certains de ces systèmes permettrait des avancées d'intérêt majeur pour notre futur à tous, allant de la médecine à la répartition des ressources terrestres et la préservation de notre environnement. Mais malheureusement, alors que la loi de Moore touche à sa fin et que la puissance de nos ordinateurs ne peut plus indéfiniment évoluer, ces calculs restent hors d'atteinte en pouvant prendre parfois plus d'un siècle sur les meilleurs supercalculateurs du monde.

Mais un espoir renaît à l'échelle atomique. Là, ce sont les lois de la mécanique quantique qui gouvernent le monde, et leurs spécificités permettent d'envisager de nouvelles machines à calculer quantiques : des ordinateurs quantiques. Ce concept fut pensé dès 1981 par l'avant-gardiste Richard Feynman, physicien de génie, pour répondre au problème de la simulation numérique de la mécanique quantique. Face aux importants calculs que demandait celle-ci, Feynman avait suggéré l'idée d'utiliser un ordinateur obéissant lui-même aux lois de la mécanique quantique pour traiter ces calculs en des temps raisonnables. Aujourd'hui, près de 40 ans après, l'avènement de l'ordinateur quantique est plus proche que jamais : au sein de grandes compagnies comme de startups, physiciens, ingénieurs, mathématiciens et nombre d'autres corps de métiers y travaillent d'arrache-pied. L'ordinateur quantique promet pour certaines classes de problèmes, une vitesse de résolution exponentiellement plus rapide que celle de nos ordinateurs actuels, laissant entrevoir la possibilité de mener des calculs jusqu'ici inaccessibles via des algorithmes spécifiques : pour la résolution de systèmes linéaires, il existe un tel algorithme, nommé HHL.

En particulier, IBM, déjà possesseur de prototypes fonctionnels d'ordinateurs quantiques, a choisi de donner au grand public l'opportunité de lancer des calculs dessus grâce au SDK open-source Qiskit. Dans la lignée de cette volonté de préparer la prochaine génération de scientifiques du monde quantique, pour laquelle la demande augmentera fortement du fait des avancées technologiques susmentionnées, IBM a proposé deux sujets de PSC d'informatique quantique aux étudiants de l'École Polytechnique. C'est un de ces PSC, portant sur HHL et encadré par M. Georges Uziel, mathématicien et ingénieur IBM, expert de l'informatique quantique, que nous présenterons ici.

Théorisé en 2009 par Aram Harrow, Avinandan Hassidim et Seth Lloyd, l'algorithme HHL est un exemple d'algorithme quantique promettant un gain de vitesse exponentiel par rapport à ses homologues classiques. Le but est de ce PSC va être la mise en œuvre effective de cet algorithme dans l'environnement de développement quantique Qiskit. L'enjeu est ici double, il s'agit non seulement de bien comprendre le fonctionnement de l'algorithme dans sa globalité mais également de déconstruire toutes les étapes de son déroulement pour pouvoir les assembler par la suite par brique élémentaire. Le fruit de ce travail a été une implémentation complète de l'algorithme dont le code, fourni dans un fichier jupyter notebook Qiskit, est joint avec ce rapport.

Nous avons organisé l'écriture du rapport de ce PSC en quatre parties. Après avoir étudié les aspects majeurs de l'informatique quantique et de sa particularité vis-à-vis de l'informatique classique, on présentera en deuxième partie en détail l'algorithme HHL en entier. Nous avons fait le choix dans cette partie de présenter exactement l'algorithme tel que nous l'avons implémenté dans notre jupyter notebook Qiskit. Pour plus de clarté nous présentons toutes les étapes de l'algorithme en une succession de sous-procédures nécessitant chacune une construction propre. L'analyse et les détails d'implémentation sont donnés dans la troisième partie du rapport, nous procédons à une analyse complète des complexités et des analyse d'erreurs de chacune des sous-procédures, mais également à une analyse complète de l'algorithme HHL. On présentera dans cette partie, une analyse empirique des performances de notre implémentation. Enfin, dans une dernière partie, nous discuterons de la démarche de projet de ce travail collectif, ainsi que des pistes d'améliorations de notre implémentation mais ainsi que des améliorations de l'algorithme HHL présents dans la littérature.

1

NOTIONS D'INFORMATIQUE QUANTIQUE

1.1 DÉFINITIONS ET NOTATIONS

1.1.1 • STOCKAGE ÉLÉMENTAIRE DE L'INFORMATION : LE QUBIT

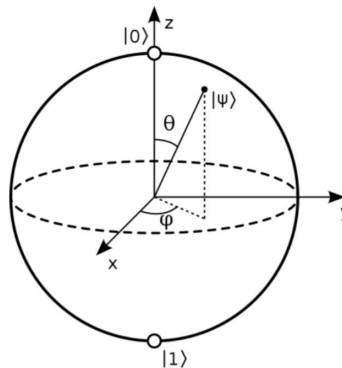
L'informatique dite classique, que nous connaissons et utilisons tous au quotidien, utilise comme brique de base le “**bit**”, unité d'information correspondant à 0 ou 1 exclusivement. Cette grandeur correspond, le plus souvent, à une valeur nulle ou non d'une tension ou d'un courant dans le circuit électrique constituant l'ordinateur.

L'informatique quantique, dont les développements dans le cadre de la seconde révolution quantique sont récents, utilise elle comme brique de base le “**qubit**”. Un qubit est un élément de norme 1 d'un espace de Hilbert \mathcal{E}_H de dimension 2, représentant l'état physique d'une particule utilisée par l'ordinateur pour effectuer ses calculs (voir 1.5 pour de brèves précisions au sujet de la réalisation physique d'ordinateurs quantiques). Cet état étant régi par les lois de la mécanique quantique, on en adoptera les notations : l'état d'un qubit sera noté comme un ket $|\psi\rangle$.

Notons $\{|0\rangle, |1\rangle\}$ une base de \mathcal{E}_H . Les lois de la mécanique quantique imposant, pour la norme euclidienne ($\|\cdot\|_2$), $\| |\psi\rangle \| = 1$, on peut écrire :

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\varphi} \sin\left(\frac{\theta}{2}\right) |1\rangle \quad \theta \in [0, \pi], \varphi \in [0, 2\pi[$$

Interpréter θ et Φ comme des coordonnées sphériques permet de représenter $|\psi\rangle$ sur une sphère, la sphère de Bloch :



Cette représentation est particulièrement pratique pour visualiser l'effet d'opérations sur les qubits.

1.1.2 • STOCKAGE AVANCÉ DE L'INFORMATION : REGISTRES DE QUBITS

Là où un bit contient un nombre parmi 0 ou 1 exclusivement, un qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ contient deux nombres complexes α et β reliés par la relation $|\alpha|^2 + |\beta|^2 = 1$.

On appelle **registre quantique** un ensemble de qubits accessibles par un ordinateur quantique et on appelle **capacité** ou **taille** d'un registre le nombre de qubits le composant. Un registre de n qubits $|\psi\rangle_n = \bigotimes_{i=0}^{n-1} (\alpha_i|0\rangle + \beta_i|1\rangle)$

contient donc 2^n nombres complexes. Dans la littérature, il est souvent noté - par abus - $|\psi\rangle_n = |\psi\rangle$. Dans les calculs que nous mènerons, nous utiliserons si besoin la notation $|\psi\rangle_n$ par souci de clarté.

Introduisons dès à présent un abus de notation commode et commun. Soit $k \in \mathbb{N}$, de décomposition binaire $\sum_{i=0}^{n-1} 2^i k_i$

: on note $|k\rangle = \bigotimes_{i=n-1}^0 |k_i\rangle = |k_{n-1}\rangle \otimes \dots \otimes |k_0\rangle$, et parle de représentation de k dans la base computationnelle. En particulier, cet abus mène à une collision de notations pour $|0\rangle$ et $|1\rangle$, collision pouvant être levée en utilisant la notation $|\psi\rangle_n$ susmentionnée.

1.1.3 • MANIPULATION DE L'INFORMATION : PORTES QUANTIQUES

En logique classique, les bits peuvent être manipulés par des portes logiques les combinant entre eux (NOT, AND, OR, NAND...). Assemblées, ces portes permettent de créer des circuits logiques implémentant des fonctions booléennes.

De même, il existe des portes quantiques permettant de manipuler un ou plusieurs qubits. Les lois physiques de la mécanique quantique imposent que toute porte quantique doit être représentable par un opérateur unitaire : on définit finalement une **porte quantique** comme une transformation unitaire agissant sur 1, 2 ou 3 qubits.

Portes à un qubit

La porte X

Donnons un premier exemple de porte quantique. Pour cela, construisons un équivalent quantique, nommé X, de la porte logique NOT. La table de vérité de la porte logique NOT est :

Entrée	Sortie
0	1
1	0

On définit donc la porte X comme étant linéaire et vérifiant :

$$X|0\rangle = |1\rangle$$

$$X|1\rangle = |0\rangle$$

En règle générale, pour $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$,

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

On pourra donc représenter X par la matrice $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, dont on vérifie bien qu'elle est unitaire.

Remarquons que :

$$X \left(\cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\Phi} \sin\left(\frac{\theta}{2}\right) |1\rangle \right) = e^{i\Phi} \left[\cos\left(\frac{\pi - \theta}{2}\right) |0\rangle + e^{-i\Phi} \sin\left(\frac{\pi - \theta}{2}\right) |1\rangle \right]$$

Une phase globale sur un ket n'ayant aucune influence sur l'état physique de la particule décrite par ce ket (cf. 1.1.4), on en déduit que la porte X s'interprète comme une rotation d'angle π autour de l'axe x de la sphère de Bloch.

La porte de Hadamard

On définit la porte de Hadamard comme étant linéaire et vérifiant :

$$H|0\rangle := |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$H|1\rangle := |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$












La représentation matricielle correspondante étant $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, qui est bien unitaire, nous avons bien défini une porte quantique.

Plusieurs remarques soulignant l'importance de la porte de Hadamard s'imposent :

- Un bit classique ne pouvant pas être dans une superposition d'états, la porte d'Hadamard n'a, au contraire de la porte X, pas d'équivalent classique. Il s'agit donc d'une première opération n'étant possible qu'en mécanique quantique.
- La porte de Hadamard permet d'avoir accès à des états superposés à partir d'états non superposés, et constitue en ce sens un bloc de base de nombre d'algorithmes quantiques, dont certains que nous présenterons dans le présent rapport.
- La porte de Hadamard correspond sur la sphère de Bloch à une rotation d'angle π autour de l'axe donné par les coordonnées $[1, 0, 1]$ (on peut s'en convaincre sans calculs en vérifiant géométriquement que c'est vrai pour les vecteurs de base $|0\rangle$ et $|1\rangle$, et en concluant par linéarité). Ceci illustre la puissance de la sphère de Bloch, dont on peut montrer qu'elle permet de visualiser n'importe quelle porte quantique à un qubit comme une suite de rotations.

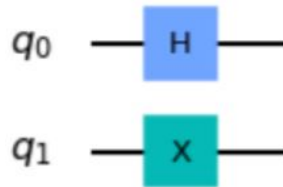
Portes à un qubit les plus communes

Le tableau ci-dessous résume les portes quantiques à un qubit les plus communes.

Nom	Matrice	Circuit	Interprétation sur sphère de Bloch			
S^\dagger or \sqrt{Z}^\dagger	$\begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$			S^\dagger or \sqrt{Z}^\dagger	$\begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$	$-\pi/2$ -rotation d'axe X
X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$		π -rotation d'axe X	T or $\sqrt[4]{Z}$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	$\pi/4$ -rotation d'axe Z
Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$		π -rotation d'axe Y	T^\dagger or $\sqrt[4]{Z}^\dagger$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix}$	$-\pi/4$ -rotation d'axe Z
Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$		π -rotation d'axe Z	$U_3(\theta, \Phi, \lambda)$	$\begin{bmatrix} \cos(\frac{\theta}{2}) & -e^{i\lambda}\sin(\frac{\theta}{2}) \\ e^{i\Phi}\sin(\frac{\theta}{2}) & e^{i(\lambda+\Phi)}\cos(\frac{\theta}{2}) \end{bmatrix}$	
H	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$		π -rotation d'axe $[1, 0, 1]$	$U_2(\Phi, \lambda)$	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -e^{i\lambda} \\ e^{i\Phi} & e^{i(\lambda+\Phi)} \end{bmatrix}$	
R_Φ	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\Phi} \end{bmatrix}$		Φ -rotation d'axe Z	$U_1(\lambda)$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}$	
I	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$		Pas de mouvement			
S or \sqrt{Z}	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$		$\pi/2$ -rotation d'axe X			

Portes à plusieurs qubits

Comme expliqué en 1.1.2, un registre de plusieurs qubits peut être représenté par un vecteur de l'espace produit tensoriel des espaces des états de chacun des qubits. Il est à noter que Qiskit ordonne ses qubits du bas vers le haut ; ainsi, le circuit :



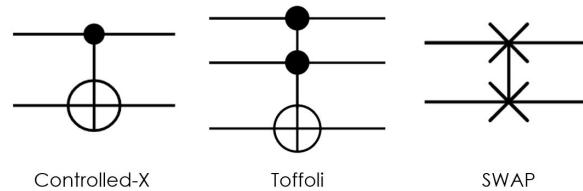
a pour matrice avec la convention d'ordre de Qiskit :

$$X \otimes H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Il est également possible de définir des portes contrôlées : soit U une porte à l qubits. On définit la porte U -contrôlée par k qubits comme la porte telle que sur la base tensorielle, si les k qubits de contrôle sont dans l'état $|1\rangle$, la porte applique U sur les l qubits cibles ; sinon, la porte contrôlée applique I sur les l qubits cibles. On admet l'existence de telles portes pour tout U et tout k .

Deux portes contrôlées remarquables sont la porte X contrôlée par 1 qubit, Controlled- X (aussi appelée CX ou $CNOT$), et la porte X contrôlée par 2 qubits, Controlled-Controlled- X (aussi appelée CCX ou Toffoli).

Enfin, notons l'existence d'une porte $SWAP$ permettant de permuter les états de deux qubits.



1.1.4 • ACQUISITION DE L'INFORMATION : MESURE DE QUBITS

La mesure d'un qubit dans l'état $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ ne peut donner que les résultats $|0\rangle$ ou $|1\rangle$, exclusivement, et ce respectivement avec une probabilité $|\alpha|^2$ et $|\beta|^2$. Suite à une telle mesure, l'état quantique du qubit est projeté sur celui qui a été mesuré : on parle de réduction de l'état quantique. En pratique, le résultat de la mesure d'un qubit est stocké dans un bit classique, et un algorithme quantique est lancé un grand nombre de fois pour déduire de la loi des grands nombres des informations sur les états finaux des qubits. Dans Qiskit, cette opération est représentée par :



Ce principe se généralise à un registre de n qubits. Par exemple, pour $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \delta|10\rangle + \gamma|11\rangle$ avec $|\alpha|^2 + |\beta|^2 + |\delta|^2 + |\gamma|^2 = 1$,

- La mesure des deux qubits donne $|00\rangle$ avec une probabilité $|\alpha|^2$ et dans ce cas, $|\psi\rangle$ est après mesure dans l'état $|00\rangle$.
- La mesure du premier qubit donne $|0\rangle$ avec une probabilité $|\alpha|^2 + |\beta|^2$ et dans ce cas, $|\psi\rangle$ vaut après mesure $\frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|00\rangle + \frac{\beta}{\sqrt{|\alpha|^2 + |\beta|^2}}|01\rangle$.

Notons que la mesure d'un qubit peut affecter les résultats possibles pour l'autre qubit : on parle d'intrication quantique. Par exemple, si un registre à deux qubits est dans l'état $|\psi\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$, et qu'une mesure pour le premier qubit donne le résultat $|0\rangle$, alors une mesure pour le deuxième qubit devra nécessairement donner $|1\rangle$: et ce, bien que le résultat $|0\rangle$ était aussi envisageable pour le deuxième qubit avant la mesure du premier.

1.1.5 • TRAITEMENT DE L'INFORMATION : ALGORITHMES QUANTIQUES

De même que toute fonction booléenne d'entrée de taille finie peut être calculée via une succession finie de portes logiques, appelée circuit logique, toute opération unitaire sur un espace de dimension 2^n peut être calculé par l'application d'une suite finie de portes quantiques à n qubits, appelée circuit quantique. On définit donc un algorithme quantique comme la donnée d'un ensemble de qubits et de bits et d'une suite finie et ordonnée de portes quantiques à appliquer sur ces qubits afin de répondre à un problème fixé, ce éventuellement avec une erreur.

On pourra représenter l'application successive de portes aux qubits via un circuit quantique. Un circuit quantique se lit de gauche à droite : les symboles des portes et les lignes sur lesquelles elles sont disposées indiquent leur nature et les qubits auxquels elles sont appliquées.

On remarque dès à présent qu'étant donné un circuit quantique, on peut déduire la représentation matricielle du circuit tout entier de celle des portes le composant et de la position relative desdites portes en appliquant la correspondance portes en parallèle/produit tensoriel (discutée en 1.1.3) et la correspondance portes en série/produit matriciel.

1.2 QUELQUES PARTICULARITÉS DE L'INFORMATIQUE QUANTIQUE

1.2.1 • RÉVERSIBILITÉ DES ALGORITHMES AVANT MESURE

Qu'est-ce que la réversibilité ?

Prenons un algorithme qui prend des entrées et renvoie des sorties. Cet algorithme peut être vu comme une fonction, dont on peut se demander si elle admet une réciproque : si c'est le cas, on dira que l'algorithme est réversible, sinon, qu'il est irréversible.

Cette caractérisation des algorithmes est non triviale, comme en témoignent les exemples suivants :

- L'algorithme qui prend en entrée deux nombres a et b et renvoie leur somme $a + b$ est irréversible : étant donné la valeur de $a + b$ et aucune autre information, on ne peut déduire les valeurs de a et de b .

- L'algorithme qui prend en entrée deux nombres a et b et renvoie leur somme et leur différence $a - b$ est réversible : étant donné les valeurs de $a + b$ et de $a - b$ et aucune autre information, on peut déduire (par une résolution de système linéaire) les valeurs de a et de b .

L'intérêt d'une telle caractérisation réside dans des considérations physiques. En 1961, alors que la communauté scientifique cherche encore à comprendre la consommation énergétique des ordinateurs, le physicien et ingénieur IBM Rolf Landauer démontre, par des considérations thermodynamiques, que les opérations irréversibles ont un coût énergétique minimal. Ce résultat, appelé principe de Landauer, ne s'applique pas aux opérations réversibles : il trace donc une délimitation physique à l'implémentation entre deux types d'algorithmes.

Application aux algorithmes quantiques avant mesure

Considérons désormais des algorithmes quantiques, en excluant la partie de mesure. Comme nous l'avons montré précédemment, un algorithme quantique se réduit à une matrice unitaire (obtenue par produits matriciel et tensoriel des matrices représentant les portes définissant cet algorithme), et est donc réversible. À ce stade, on peut déjà remarquer une cohérence logique avec ce qui a précédemment été expliqué : la notion de réversibilité est étroitement liée à la réalité physique de l'implémentation d'un algorithme, et c'est la physique qui impose l'unitarité des portes quantiques.

L'inverse d'un algorithme quantique "Algo" a pour représentation matricielle la matrice adjointe à la représentation matricielle de l'algorithme "Algo". Dans la pratique, l'inverse d'un algorithme quantique s'obtient en appliquant en ordre inverse les opérations adjointes de celles définissant l'algorithme d'origine. L'inversion d'ordre découle simplement de la propriété mathématique suivante, où \dagger désigne l'adjointe d'une matrice :

$$\forall(A, B) \in \mathcal{M}_n(\mathbb{C})^2, (AB)^\dagger = B^\dagger A^\dagger$$

(* Cela signifie en pratique qu'il est possible de reconstituer la valeur d'entrée d'un algorithme quantique en connaissant sa sortie *)

Une spécificité parfois contraignante : le cas de la porte AND

Lorsque nous avons défini les portes X et de Hadamard, nous avons insisté sur l'unitarité des matrices obtenues pour justifier que l'objet construit était bien une porte quantique. En pratique, ce critère n'est pas toujours vérifié, et il n'est pas toujours aisé d'obtenir des analogues quantiques des opérations classiques.

Montrons par exemple qu'il n'existe pas de porte quantique *AND* sur deux qubits telle que :

$$AND|00\rangle = |0\rangle \otimes \dots$$

$$AND|01\rangle = |0\rangle \otimes \dots$$

$$AND|10\rangle = |0\rangle \otimes \dots$$

$$AND|11\rangle = |1\rangle \otimes \dots$$

où ... peut désigner n'importe quel état quantique.

Supposons par l'absurde disposer d'une telle porte. Sa représentation matricielle, pour la base $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, est de la forme:

$$\begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

Or, une telle matrice est de rang au plus 3, donc non inversible et a fortiori non unitaire. Absurde.

Quitte à considérer l'utilisation de portes SWAP, on en déduit qu'il n'existe pas de porte quantique à deux qubits réalisant un analogue de l'opération logique classique AND sur un de ces qubits.

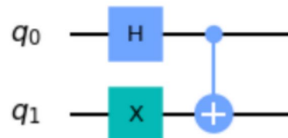
Pour faire le lien avec ce qui a été dit précédemment sur la réversibilité, remarquons que l'opération logique classique AND est irréversible pour des algorithmes prenant deux bits en entrée et deux bits en sortie : c'est un exercice de mathématiques que de montrer qu'il n'existe pas $f : \{0, 1\}^2 \rightarrow \{0, 1\}^2$ telle que pour tout $(a, b) \in \{0, 1\}^2$, la connaissance de $a \times b$ et de $f(a, b)$ permet de déterminer a et b .

1.2.2 • PARALLÉLISME QUANTIQUE

La boucle “Si *condition* Alors *action1* Sinon *action2*” de l'informatique classique a un comportement particulier en informatique quantique. En effet, si *condition* est un bit, alors *condition* est dans l'état 0 (Faux) ou exclusif 1 (Vrai), donc *action1* ou exclusif *action2* est réalisée. Mais si *condition* est un qubit, alors *condition* est dans une superposition linéaire de $|0\rangle$ et de $|1\rangle$, et c'est une superposition linéaire de *action1* et *action2* qui est réalisée : on parle de parallélisme quantique.

En pratique, on pourra souvent raisonner par linéarité.

Étudions par exemple le circuit :



- On sait que la porte CX vérifie :

$$\begin{aligned} CX|00\rangle &= |00\rangle \\ CX|10\rangle &= |11\rangle \end{aligned}$$

- Or, après application de la porte H dans le circuit étudié, l'état du registre est :

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$$

- Donc, par linéarité, l'état du registre après application du circuit complet est :

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

1.2.3 • DES ALGORITHMES À LANCER UN GRAND NOMBRE DE FOIS

Classiquement, un algorithme est implémenté par un programme à exécuter une fois pour obtenir la sortie de l'algorithme.

Quantiquement, il est à noter que la structure quantique de l'opération de mesure fait qu'à l'issue de l'exécution d'un programme implémentant un algorithme quantique, on ne récupère pas l'état quantique des qubits en fin d'algorithme, mais seulement une information partielle.

Pour discuter l'impact que cela a en informatique quantique plus en profondeur, considérons l'exemple d'un algorithme sur un seul qubit, que l'on suppose avoir implémenté sur un ordinateur quantique idéal (ne faisant jamais d'erreur). A entrée donnée, à l'issue de cet algorithme, ce qubit est placé avec certitude dans un état $|\psi\rangle$, que l'on cherche à déterminer (puisqu'il s'agit, fondamentalement, de la réponse à la question qu'adresse notre algorithme). Cela revient à déterminer les coefficients α et β tels que $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

- Commençons par lancer l'algorithme une fois, comme on le ferait en informatique classique, et mesurons le qubit dans la base $\{|0\rangle, |1\rangle\}$ à l'issue de l'exécution. Supposons par exemple trouver le résultat $|0\rangle$: cette information nous indique seulement que "la probabilité de mesurer le qubit dans l'état $|0\rangle$ est non nulle", i.e. que $\alpha \neq 0$.
- Afin d'obtenir plus d'informations, on lance l'algorithme un grand nombre de fois, en mesurant à chaque fois le qubit dans la base $\{|0\rangle, |1\rangle\}$ et gardant note des résultats obtenus. On obtient une proportion p de résultats $|0\rangle$ et une proportion $1 - p$ de résultats $|1\rangle$: par loi des grands nombres, en supposant avoir lancé l'algorithme suffisamment de fois pour avoir eu une convergence satisfaisante, on déduit de cela que $|\alpha|^2 = p$ et $|\beta|^2 = 1 - p$.
- Mais cela ne suffit toujours pas à déterminer entièrement $|\psi\rangle$. Supposons par exemple que $p = 1/2$: l'information que nous avons acquise jusqu'ici laisserait notamment les possibilités $|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ et $|\psi\rangle = |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, qui correspondent pourtant à deux situations totalement différentes (l'ensemble $\{|+\rangle, |-\rangle\}$ étant obtenu par rotation de l'ensemble $\{|0\rangle, |1\rangle\}$, ne pas pouvoir distinguer $|+\rangle$ et $|-\rangle$ correspond à un manque d'information équivalent à ne pas pouvoir distinguer $|0\rangle$ et $|1\rangle$).

En pratique, la mesure des données résultant de l'exécution d'un algorithme quantique constitue donc un problème complexe. Il convient de se demander quelles sont les informations véritablement utiles à récupérer, et en particulier s'il faut déterminer les phases des composantes des qubits dans certaines bases données ou si les modules de ces composantes suffisent.

1.3 COMPLEXITÉ DES ALGORITHMES QUANTIQUES

Première approche

Fondamentalement, le développement d'une théorie de la complexité en informatique vise à répondre à la question "En fonction de la longueur de son entrée et éventuellement d'autres paramètres pertinents, combien de temps met cet algorithme à terminer ?" via des estimations asymptotiques. Dans le cas de l'informatique classique, on compte ce temps en nombre d'opérations élémentaires, le caractère élémentaire ou non d'une opération ayant été défini à l'avance (ce choix est même d'importance relative, puisque par thèse de Church, on peut considérer qu'un temps est polynomial sur une machine usuelle si et seulement si il l'est sur une machine de Turing). Mais dans le cadre de l'informatique quantique, définir ces opérations de base n'est pas évident : en effet, il existe de nombreuses stratégies de fabrication d'ordinateurs quantiques (cf. 1.4), entre lesquelles le temps d'exécution de certaines portes peut varier.

Un résultat classique sur les circuits logiques est que tout circuit logique peut être fabriqué en utilisant seulement des portes NAND : on parle d'universalité de l'ensemble $\{\text{NAND}\}$.

On pourrait penser définir de même un ensemble de portes quantiques universel comme un ensemble de portes quantiques engendrant (avec les multiplications tensorielle et matricielle) toute porte quantique. Néanmoins, l'ensemble des portes quantiques étant indénombrable, on ne pourrait trouver d'ensemble de portes quantiques universel fini ou dénombrable. Considérant de plus qu'il n'est pas évident qu'une construction physique d'un ordinateur quantique

puisse effectuer en un temps efficace une infinité de portes quantiques différentes (avant même de se poser la question de l'universalité), une telle définition d'un ensemble de portes quantiques universel ne servirait pas l'objectif de définir une complexité pour l'algorithmie quantique.

On définira donc un ensemble de portes quantiques universel comme :

Ensemble de portes quantiques universel : *Ensemble de portes quantiques engendrant un sous-ensemble dense de l'ensemble des portes quantiques.*

Théorème de Solovay-Kitaev et conséquences

Théorème de Solovay-Kitaev : *Soit \mathcal{G} un ensemble de portes quantiques universel. Soit U la matrice correspondant à un certain circuit quantique. Pour tout $\epsilon > 0$, il existe un $c > 0$ tel que U peut être approximée au sens de la norme d'opérateur à ϵ près par un circuit fait de $O(\log^c(\frac{1}{\epsilon}))$ portes de \mathcal{G} .*

Une conséquence de ce théorème est que, du moins pour l'étude des temps polynomiaux et supérieurs, on peut définir de façon cohérente la complexité d'un algorithme quantique comme le nombre de portes d'un ensemble universel donné, et ce malgré l'approximation faite sur le résultat. Il est à noter que, par ailleurs, ces approximations ne remettent pas en cause l'exactitude des algorithmes, dans la mesure où les résultats d'un algorithme quantique sont issus d'un moyennage statistique et comportent donc intrinsèquement des imprécisions.

Shi a montré que $\{\text{Toffoli}, H\}$ est universel. **C'est en termes de ces portes que seront exprimées toutes les complexités qui suivent.**

1.4 UN PEU DE PHYSIQUE : DE QUOI EST FAIT UN ORDINATEUR QUANTIQUE ?

En pratique, l'état $|\psi\rangle$ peut représenter différentes caractéristiques physiques de différents systèmes : spin d'une particule quelconque de spin $1/2$, polarisation d'un photon... De ce fait, il existe différentes approches basées sur des technologies très différentes pour construire des ordinateurs quantiques. Ces choix de technologies ne sont pas anodins, car certaines portes peuvent être plus faciles et plus efficaces à implémenter que d'autres selon le choix technologique retenu.

Pour l'instant, de nombreux problèmes liés au hardware entravent l'avènement de l'informatique quantique. Citons la décohérence : au bout d'un certain temps, les particules correspondant aux qubits quittent leur état de superposition pour tomber dans un état parmi $|0\rangle$ et $|1\rangle$, réduisant le qubit à un simple bit. A titre indicatif, cette durée est actuellement de l'ordre de $100\mu\text{s}$, ce qui ne permet pas encore de surpasser les meilleurs ordinateurs classiques pour certains problèmes spécifiques.

1.5 RÉSUMÉ : ORDINATEUR CLASSIQUE VS ORDINATEUR QUANTIQUE

	Informatique classique	Informatique quantique
Information	Bit : binaire, discret	Qubit : quantique, continue
Mesure	Non destructrice, déterministe	Destructrice, indéterministe
Porte	Porte logique : fonction booléenne discrète	Porte quantique : opérateur unitaire continue
Registre de taille n	n bits, 2^n valeurs possibles	n qubits, espace de Hilbert de dimension 2^n

Table 1: Différence entre informatique classique et quantique

2

L'ALGORITHME HHL : FONCTIONNEMENT ET IMPLÉMENTATION

2.1 POSITION DU PROBLÈME

2.1.1 • PRÉLIMINAIRE : QUELQUES DÉFINITIONS MATHÉMATIQUES

On fixe $N \in \mathbb{N}^*$ pour cette partie.

Définition (Matrice s -creuse) : Une matrice est dite s -creuse ssi elle admet au plus s éléments non nuls par colonne.

Une matrice hermitienne admettant des valeurs propres réelles, on peut définir :

Définition (Conditionnement d'une matrice hermitienne) : Soit $A \in GL_N(\mathbb{C})$ hermitienne, dont on note λ_{\max} (resp. λ_{\min}) la valeur propre maximale (resp. minimale) en valeur absolue.

Le conditionnement de A est le réel $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$

2.1.2 • LE PROBLÈME LSP

Dans la littérature, le problème mathématique correspondant à la résolution d'un système linéaire se nomme Linear System Problem, abrégé en LSP. Il se définit mathématiquement comme suit avec le formalisme matriciel :

Définition (Problème LSP) : Soit $N \in \mathbb{N}^*$. Soient $A \in GL_N(\mathbb{C})$ et $b \in \mathbb{C}^N$.

Le problème LSP consiste à déterminer $x \in \mathbb{C}^N$ tel que $Ax = b$.

Dans de nombreuses applications pratiques pouvant notamment avoir trait à la physique ou aux mathématiques appliquées, connaître une approximation du vecteur x susmentionné est suffisant. Ceci mène à définir le problème LSP étendu :

Définition (Problème LSP étendu) : Soit $N \in \mathbb{N}^*$. Soient $A \in GL_N(\mathbb{C})$ et $b \in \mathbb{C}^N$. Soit $\varepsilon > 0$. Le problème LSP consiste à déterminer $\tilde{x} \in \mathbb{C}^N$ tel qu'en notant $x \in \mathbb{C}^N$ tel que $Ax = b$,

$$|\tilde{x} - x|_2 < \varepsilon$$

Le paramètre ε est appelé paramètre d'erreur ou précision du résultat.

2.1.3 • LE PROBLÈME QLSP

Le problème LSP admet un équivalent quantique, QLSP (Quantum Linear System Problem). Ce dernier est défini par :

Définition (Problème QLSP) : Soit $N \in \mathbb{N}^*$. Soient $A \in GL_N(\mathbb{C})$ telle que $\det A = 1$, $b \in \mathbb{C}^N \setminus \{0\}$, et $\varepsilon > 0$. On note $x = A^{-1}b$, et pose :

$$|x\rangle = \frac{\sum_{i=0}^{N-1} x_i |i\rangle}{\left\| \sum_{i=0}^{N-1} x_i |i\rangle \right\|_2}$$

Le problème *QLSP* consiste à trouver, avec une probabilité strictement supérieure à $\frac{1}{2}$, un état quantique $|\tilde{x}\rangle$ tel que :

$$\| |\tilde{x}\rangle - |x\rangle \|_2 < \varepsilon$$

Plusieurs remarques s'imposent :

- Le fait de ne demander à un éventuel algorithme solution de ne donner un résultat correct qu'avec une certaine probabilité découle, intrinsèquement, de la nature quantique de ses opérations. Si ce qui a été vu en partie 1 laisse penser qu'à l'issue d'un algorithme quantique, l'état de sortie du système est parfaitement déterminé, il ne faut pas oublier que la mesure est source d'indétermination ; aussi, l'utilisation d'une mesure de qubits auxiliaires à ceux stockant le résultat final peut permettre d'opérer une sélection de résultats au prix d'une probabilité que l'algorithme renvoie une réponse erronée. Nous discuterons tout ceci plus concrètement lorsque nous présenterons HHL.
- le $1/2$
- L'hypothèse sur \mathbf{A} peut être réduite à \mathbf{A} inversible, quitte à considérer :

$$\tilde{\mathbf{A}} = \frac{1}{|\det(\mathbf{A})|^2} \begin{bmatrix} 0 & \mathbf{A} \\ \mathbf{A}^\dagger & 0 \end{bmatrix}$$

2.2 HHL

L'algorithme HHL a été découvert en 2009 par Arram Harrow, Avinatan Hassidim et Seth Lloyd, qui lui ont légué leurs initiales. Cet algorithme quantique permet de résoudre QLSP avec une complexité compétitive par rapport aux meilleurs algorithmes classiques. Nous allons ici en expliquer le fonctionnement en passant détails et technicités liées à l'implémentation, qui seront discutées partie suivante.

On prend en entrée les données de QLSP, avec $N = 2^n$. Par théorème spectral, on représente \mathbf{A} sous la forme $\mathbf{A} = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle \langle u_j|$, où les $|u_j\rangle$ sont les vecteurs propres de \mathbf{A} et les λ_j leurs valeurs propres associées.

HHL s'exécutera sur un circuit de $n_b + n_l + 1$ qubits, où :

- n_b qubits forment un registre servant à stocker d'abord l'état $|b\rangle$, que nous ferons évoluer pour obtenir $|\tilde{x}\rangle$. Avec les notations de QLSP, on a donc $n_b = n$.
- n_l qubits forment un registre servant à manipuler une représentation des valeurs propres de la matrice \mathbf{A} , comme expliqué plus loin. Cette taille est arbitraire.
- 1 qubit forme un registre auxiliaire S servant à opérer une sélection des résultats, action également détaillée plus loin.

On notera donc l'état quantique manipulé par HHL sous la forme $|\chi\rangle_S |\psi\rangle_{n_l} |\phi\rangle_{n_b}$.

2.2.1 • PREMIÈRE ÉTAPE : CHARGEMENT DES DONNÉES

Initialement, l'état quantique du système complet est $|0\rangle_S |0\rangle_{n_1} |0\rangle_{n_b}$.

A l'aide de l'algorithme dit de qRAM loading, l'état du registre à n_b qubits est transformé en :

$$|\mathbf{b}\rangle_{n_b} = \frac{\sum_{i=0}^{N-1} b_i |u_i\rangle}{\left\| \sum_{i=0}^{N-1} b_i |u_i\rangle \right\|}$$

Quitte à renormaliser le système à résoudre, on peut supposer que $|b|_2 = 1$. Ceci permet d'alléger l'expression précédente en :

$$|\mathbf{b}\rangle_{n_b} = \sum_{i=0}^{N-1} b_i |u_i\rangle$$

Donc, à l'issue de cette première étape, l'état du système complet est :

$$|0\rangle_S |0\rangle_{n_1} |\mathbf{b}\rangle_{n_b}$$

2.2.2 • DEUXIÈME ÉTAPE : QPE

Commençons par une définition de l'algorithme QPE pour à une matrice unitaire U .

Définition (Algorithme QPE pour une matrice unitaire U) :

Soient $n_1 \in \mathbb{N}^*$ et $n_2 \in \mathbb{N}^*$. Soit un opérateur unitaire $U = \sum_{j=0}^{N-1} e^{i2\pi\theta_j} |u_j\rangle\langle u_j|$, où les θ_j appartiennent à \mathbb{R} . On considère un ensemble de $n_1 + n_2$ qubits constitué d'un registre de n_1 qubits et d'un registre de n_2 qubits. Il existe un algorithme quantique sur cet ensemble de qubits, noté QPE, tel qu'en notant pour tout $j \in \llbracket 0, N-1 \rrbracket$ $\tilde{\theta}_j$ la meilleure approximation binaire à n_1 chiffres de $2^{n_1}\theta_j$,

$$\forall j \in \llbracket 0, N-1 \rrbracket, QPE(|0\rangle_{n_1} |u_j\rangle_{n_2}) = |\tilde{\theta}\rangle_{n_1} |u_j\rangle_{n_2}$$

Deux brèves remarques s'imposent :

- Le nom QPE est acronyme de Quantum Phase Estimation : ceci fait sens dans la mesure où cet algorithme permet d'estimer ce qui correspond physiquement à des phases.
- Nous avons noté QPE chaque élément d'une famille d'algorithmes indexée par $U \in SU(n)$; en toute rigueur, il aurait fallu noter l'algorithme défini ci-dessus QPE_U .

Ceci étant fait, faisons le lien avec HHL. La seconde étape de HHL consiste à appliquer QPE pour e^{iAt} , où $t \in \mathbb{R}$ est un paramètre fixé préalablement à l'exécution de l'algorithme - ceci sera discuté plus en détail lorsque nous décrirons l'implémentation de QPE. On peut néanmoins observer dès maintenant que l'hermitianité de A garantit l'unitarité de U .

En notant $A = \sum_{j=0}^{n-1} \lambda_j |u_j\rangle\langle u_j|$, il vient que $U = \sum_{j=0}^{n-1} e^{i\lambda_j t} |u_j\rangle\langle u_j|$. Par abus, notons $\widetilde{\lambda}_j = \widetilde{\lambda}_j t$ et a fortiori $|\widetilde{\lambda}_j\rangle = |\widetilde{\lambda}_j t\rangle$

(la validité et la commodité de cet abus de notations seront justifiés plus loin).

Avant application de QPE, l'état du système est :

$$|0\rangle_S \otimes \sum_{j=0}^{n-1} b_j |0\rangle_{n_l} |u_j\rangle_{n_b}$$

Donc après application de QPE, l'état du système est :

$$|0\rangle_S \otimes \sum_{j=0}^{n-1} b_j |\tilde{\lambda}_j\rangle_{n_l} |u_j\rangle_{n_b}$$

2.2.3 • TROISIÈME ÉTAPE : *EIGROT*

EIGROT est un nom venant de EIGenvalu ROTation : ceci se comprendra aisément à la fin de la description sommaire que nous allons ici en faire.

Notons $C \in \mathbb{R}$ un paramètre défini préalablement à l'exécution de l'algorithme (discuté plus en détail en partie implémentation) tel que pour tout $j \in \llbracket 0, N-1 \rrbracket$ $\tilde{\theta}_j, \vartheta_j = \arcsin\left(\frac{C}{\lambda_j}\right)$ est bien défini.

Via des algorithmes permettant des manipulations arithmétiques réunis en un algorithme nommé *ASNINV* que l'on applique au registre de taille n_l , on commence par placer le système dans l'état :

$$|0\rangle_S \otimes \sum_{j=0}^{n-1} b_j |\vartheta_j\rangle_{n_l} |u_j\rangle_{n_b}$$

Puis, à l'aide de rotations successives appliquées au qubit solitaire S et contrôlées par les qubits du registre de taille n_l , on peut placer le système dans l'état :

$$\begin{aligned} & \sum_{j=0}^{n-1} b_j (\pm \arccos(\vartheta_j) |0\rangle_S + \arcsin(\vartheta_j) |1\rangle_S) |\tilde{\lambda}_j\rangle_{n_l} |u_j\rangle_{n_b} \\ &= \sum_{j=0}^{n-1} b_j \left(\pm \sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle_S + \frac{C}{\lambda_j} |1\rangle_S \right) |\tilde{\lambda}_j\rangle_{n_l} |u_j\rangle_{n_b} \end{aligned}$$

2.2.4 • QUATRIÈME ÉTAPE : UNCOMPUTE

Dans cette étape, on ramène dans leur état initial les qubits du registre de taille n_l . Pour ce faire, on applique successivement l'inverse de *ASNINV*, qui est *ASNINV*[†], puis l'inverse de *QPE*, qui est *QPE*[†].

Ainsi, ces deux algorithmes ayant été appliqués, on obtient l'état :

$$\sum_{j=0}^{n-1} b_j \left(\pm \sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle_S + \frac{C}{\lambda_j} |1\rangle_S \right) |0\rangle_{n_l} |u_j\rangle_{n_b}$$

2.2.5 • CINQUIÈME ÉTAPE : MESURES

On mesure désormais le qubit seul (formant le registre S), et ne continue à effectuer des calculs que si l'état obtenu est $|1\rangle$. Ceci permet d'effectuer une sélection des résultats : remarquons d'une part que cela n'empêche pas l'algorithme de donner une réponse, puisque celle-ci est obtenue statistiquement (cf. 1.2.3), et d'autre part que cela explique l'introduction d'une condition de réussite avec une certaine probabilité dans la définition de QLSP.

L'état du système est désormais :

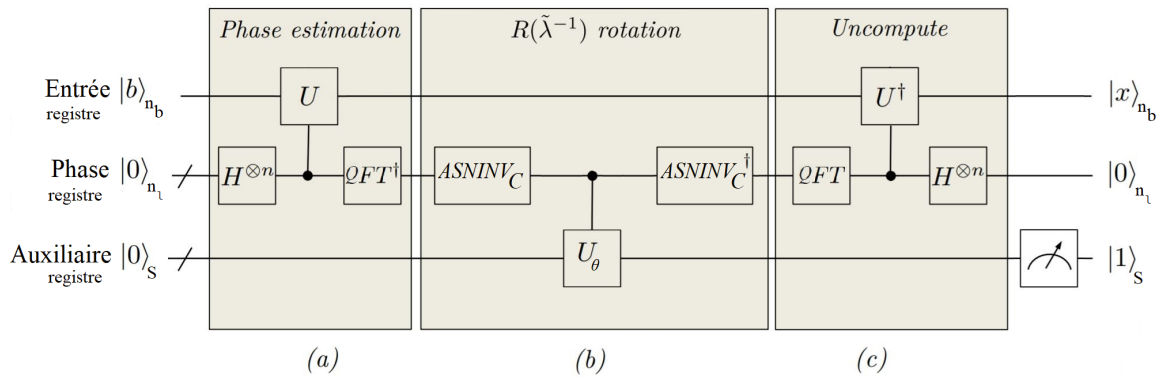
$$\sqrt{\frac{1}{\sum_{j=0}^{N-1} \frac{b_j^2}{\lambda_j^2}}} |0\rangle_S |0\rangle_{n_l} \sum_{j=0}^{n-1} \frac{b_j}{\lambda_j} |u_j\rangle_{n_b}$$

$$\propto |0\rangle_S |0\rangle_{n_l} \sum_{j=0}^{n-1} \frac{b_j}{\lambda_j} |u_j\rangle_{n_b}$$

<https://www.overleaf.com/project/60748229eb4c4f648db302a8> Mesurer différentes observables permet alors de déterminer $|\tilde{x}\rangle$, approximation à ε près de $|x\rangle \propto \sum_{j=0}^{n-1} \frac{b_j}{\lambda_j} |u_j\rangle_{n_b}$.

2.2.6 • VUE D'ENSEMBLE DE L'ALGORITHME

Ce que nous venons de décrire peut être résumé schématiquement par :



2.2.7 • UN EXEMPLE DÉTAILLÉ

2.3 FONCTIONNEMENT ET IMPLÉMENTATION DE QRAM LOADING

Soit $x \in \mathbb{C}^N$ le vecteur d'entrée, $|x\rangle := \|x\|_2^{-1} \sum_{i=1}^N x_i |i-1\rangle$ un état quantique sur $\lceil \log(N) \rceil$ qubits et \tilde{x} un vecteur qui approxime x avec une précision finie. Une RAM quantique (qRAM) est l'opération \mathcal{R} tel que :

$$\mathcal{R} : \tilde{x} \mapsto \widetilde{\|x\|_2} |x\rangle$$

où $\widetilde{\|x\|_2}$ est un état pur encoding la valeur approximative de $\|x\|_2$. L'implémentation de cet opérateur est faite de manière très similaire à l'approche proposé par (Dernovic et al., 2018) : en utilisant la structure de mémoire proposée par (Prakash, 2014) avec la procédure pour préparer un état initial de (Grover & Rudolph, 2002). La structure et la procédure originales traitaient seulement le cas où x était un vecteur à composantes réelles.

Il s'agit d'une structure de données classique à accès quantique.

• Arbre binaire B_x

Le vecteur normalisé x est stocké dans un arbre binaire B_x (cf. Figure 1). Pour chaque élément de x , x_i il y a une feuille dans l'arbre. Chaque feuille contient le module de x_i au carré et l'argument de x_i . Tout autre sommet contient la somme de ses fils.

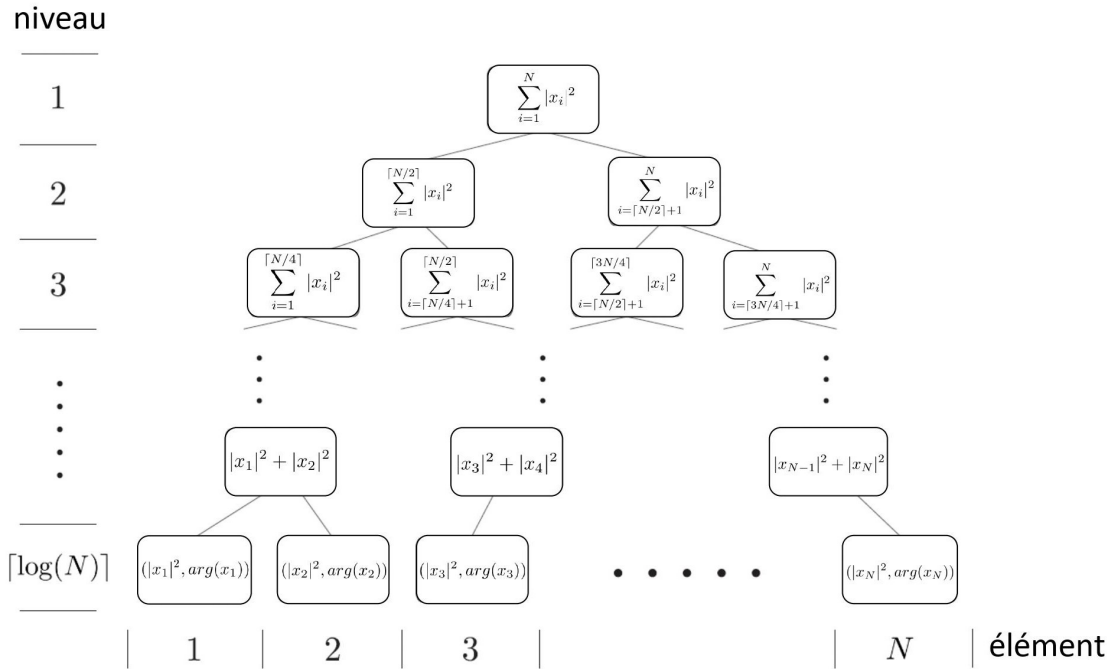


Figure 1: Illustration d'une structure de données classique B_x qui, lorsqu'elle est équipée d'un accès quantique, constitue une qRAM, stockant un vecteur $x \in \mathbb{R}^N$.

• chargement par qRAM

La procédure pour préparer $|x\rangle$ à partir de x est montrée dans l'Algorithme 1. Pour un x donné, le but est d'associer x_i à chaque état pur $|i-1\rangle$. Ceci est réalisé par le parcours en profondeur de l'arbre binaire B_x , dont la représentation binaire des feuilles correspondent aux états purs $|i\rangle$. Au niveau k , on va effectuer une rotation sur le qubit k contrôlée par les qubits $0, 1, \dots, k-1$. Il y aura autant de rotations contrôlées que des sommets au niveau $k-1$. Chaque rotation va attribuer une amplitude (de probabilité) à chaque état pur $|i\rangle$ qui va s'approcher de la bonne valeur au fur et à mesure que le parcours atteint la feuille correspondante.

Théorème: Soit $x \in \mathbb{C}^N$, $\|x\|_2 = 1$ et $|x\rangle = \sum_{i=1}^N x_i |i-1\rangle$. Supposons que x est stocké dans la structure de données B_x telle qu'elle est décrite ci-dessus. Alors, l'algorithme 1 prépare l'état $|x\rangle$ en utilisant $\mathcal{O}(N \log(N))$ portes CNOT et $\mathcal{O}(N \log(N)^2)$ portes à un qubit..

Preuve. Tout d'abord, vérifions que l'Algorithme 1 prépare bien l'état $|x\rangle$. Soit $|\tilde{x}\rangle$ l'état sortante de l'algorithme et \tilde{x}_i son i -ème amplitude. Par construction, \tilde{x}_i a été calculé en parcourant B_x depuis la racine jusqu'à la feuille i

Algorithm 1 Chargement d'un vecteur par une qRAM

```

1: Input : vecteur  $x \in \mathbb{C}^N$ ,  $\|x\|_2 = 1$ , stocké dans un arbre binaire classique  $B_x$  (cf. Figure 1).
2: Output : état quantique  $|x\rangle$  dans un circuit quantique.
3: Initialiser  $\lceil \log(N) \rceil$  qubits à  $|0\rangle^{\otimes \lceil \log(N) \rceil}$ 
4: Soit  $v$  la racine de  $B_x$ . Alors, exécuter  $\text{processNode}(v)$ .
5: function  $\text{PROCESSNODE}(\text{sommet } v)$ 
6:   Soit  $u_l$  et  $u_r$  le fils à gauche et à droite de  $v$  respectivement et  $k$  le niveau où se trouvent  $u_l$  et  $u_r$  dans  $B_x$ .
7:    $\theta = 2 \arccos\left(\sqrt{\text{valeur}(u_l)/\text{valeur}(u)}\right)$ .
8:   Soit  $C^{k-1}\text{-}R_y(\theta)$  la porte de rotation contrôlée:  $|q_1 \dots q_{k-1}\rangle \langle q_1, \dots, q_{k-1}| \otimes \exp(-iq_1 \dots q_{k-1} \sigma_y) \otimes I_{N-k}$ 
9:   où  $q_1 \dots q_{k-1}$  est la représentation binaire de  $v$  par rapport aux autres sommets au niveau  $k-1$ .
10:  if  $u_l, u_r$  sont des feuilles then
11:    Soit  $x_l$  et  $x_r$  les deux composantes stockés dans  $u_l$  et  $u_r$  respectivement.
12:    if  $x_l, x_r \in \mathbb{R}$  then
13:      Ajuster  $\theta$  de façon à obtenir le bon signe des deux composantes.
14:      Effectuer la porte quantique  $C^{k-1}\text{-}R_y(\theta)$ .
15:    else
16:      Effectuer la porte quantique  $C^{k-1}\text{-}R_y(\theta)$ .
17:      Effectuer la porte contrôlée :  $|q_1 \dots q_{k-1}\rangle \langle q_1, \dots, q_{k-1}| \otimes U_1(\arg(x_r) - \arg(x_l)) \otimes I_{N-k}$ 
18:    end if
19:  else
20:     $\text{PROCESSNODE}(u_l)$ 
21:     $\text{PROCESSNODE}(u_r)$ 
22:  end if
23: end function

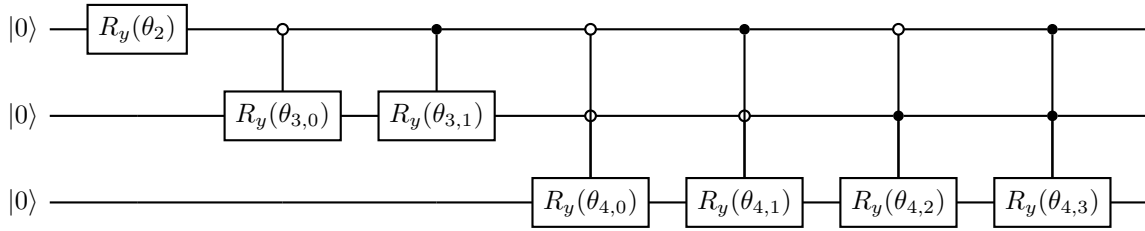
```

le long d'un chemin P_i , multipliant par un facteur pertinent à chaque sommet intermédiaire suivi de l'argument de x_i à la fin. Maintenant, prenons $P_i = (u_1, u_2, \dots, u_{\lceil \log(N) \rceil})$. Le facteur par lequel nous multiplions à chaque sommet intermédiaire u_k est $\sqrt{\text{valeur}(u_k)/\text{valeur}(u_{k-1})}$. D'où on obtient

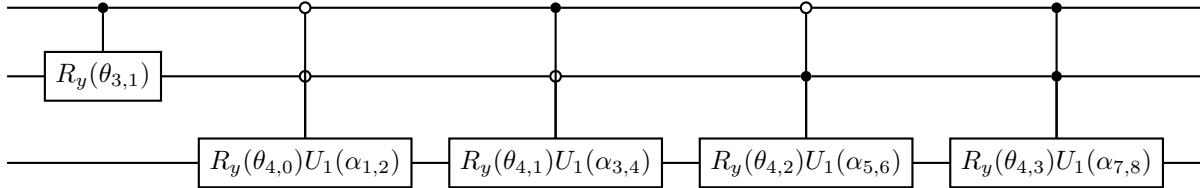
$$\tilde{x}_i = \prod_{k=2}^{\lceil \log(N) \rceil} \sqrt{\frac{\text{valeur}(u_k)}{\text{valeur}(u_{k-1})}} \text{sgn}(x_i) = \sqrt{\frac{\text{valeur}(u_{\lceil \log(N) \rceil})}{\text{valeur}(u_1)}} \text{sgn}(x_i) = \sqrt{\frac{|x_i|^2}{1}} \arg(x_i) = x_i$$

Comme cet argument est valable pour tout $i \in \{1, \dots, n\}$ nous avons que $|\tilde{x}\rangle = |x\rangle$.

Notons $\theta_{k,s}$ l'argument de la rotation effectuée au niveau k de l'arbre B_x et au sommet s dans l'algorithme 1. Pour $x \in \mathbb{R}^8$, la sortie le résultat qu'on obtient de l'algorithme 1 est le suivant:



En revanche, si $x \notin \mathbb{R}^8$, les 4 dernières portes changent pour introduire les phases relatives entre chaque pair (x_s, x_{s+1}) pour $s \in \{1, 3, 5, 7\}$, lesquelles seront notées $\alpha_{s,s+1}$:



En utilisant les identités des portes quantiques et en supposant que $k = \lceil \log(N) \rceil = \log(N)$, le nombre de portes pour chaque étape (niveau k dans l'arbre B_x) est montré ci-dessus. À partir de $k = 4$ le nombre de portes commence à augmenter rapidement à cause de l'utilisation des portes à plusieurs qubits de contrôle pour effectuer les rotations contrôlées. Cependant, ce n'est qu'à la dernière étape où on trouve le maximum du nombre de portes. En effet, aux portes utilisées pour effectuer les rotations à plusieurs qubits de contrôle (obéissant à la relation qu'il a été trouvé pour $4 \leq k < \log(N)$ en bas) il faut ajouter des portes pour effectuer les portes U_1 à un contrôle, utilisées pour introduire l'argument des composantes du vecteur complexe x .

- Pour $k = 2$, 1·{porte à un qubit}
- Pour $k = 3$, 4·CNOTs + 10·{porte à un qubit}
- Pour $k = 4$, 4 · ((12 · 2 - 10)·CNOTs + (20 · 2 - 16) · {porte à un qubit}) + 8 · {porte à un qubit}
- Pour $4 \leq k < \log(N)$, $2^{k-2}((12(k-2) - 10) \cdot \text{CNOTs} + 2^{k-2}(k-2)(20(k-2) - 16) \cdot \{\text{porte à un qubit}\}) = 2^{k-1}((6k-17) \cdot \text{CNOTs} + 2^{k-1}(k-2)(10k-28) \cdot \{\text{porte à un qubit}\})$

- Pour $k = \log(N)$, $\frac{N}{4}((12 \log(N) - 34) \cdot \text{CNOTs} + \frac{N}{4}(\log(N) - 2)(20 \log(N) - 56) \cdot \{\text{porte à un qubit}\}) + \frac{N}{4}(2 \cdot \text{CNOTs} + 4 \cdot \{\text{porte à un qubit}\}) = \frac{N}{2}((6 \log(N) - 16) \cdot \text{CNOTs} + \frac{N}{2}((\log(N) - 2)(10 \log(N) - 28) + 2) \cdot \{\text{porte à un qubit}\})$.

Pour calculer le nombre total de portes CNOTs on utilisera sans démontrer le résultat suivant: $\sum_{k=1}^n k 2^k = (n-1)2^{n+1} + 2$. De l'analyse précédente, on observe que le nombre portes CNOTs à chaque étape de l'algorithme 1 est borné par $2^k((3k-8))$. Ainsi, le nombre total de portes CNOTs est borné par

$$\sum_{k=1}^{\log(N)} 2^k((3k-8)) < 6 \sum_{k=1}^{\log(N)} k 2^k = 6((\log(N)-1)2N + 2) = \mathcal{O}(N \log(N))$$

De même, pour le nombre total de portes à un qubit on utilisera sans démontrer le résultat suivant: $\sum_{k=1}^n k^2 2^k = n 2^{n+2} + (n^2-1)2^{n+1} - (n-1)2^{n+3} - 6$. De l'analyse pas à pas ci-dessus, on observe qu'à chaque étape le nombre de portes à un qubit est borné par $2^{k-1}((k-2)(10k-28)-2)$. Ainsi, le nombre total de portes à un qubit est borné par

$$\sum_{k=1}^{\log(N)} 2^{k-1}((k-2)(10k-28)-2) < 5 \sum_{k=1}^{\log(N)} k^2 2^k = 4N \log(N) + 2N(\log(N)^2 - 1) - 8N(\log(N) - 1) - 6 = \mathcal{O}(N \log(N)^2)$$

2.4 FONCTIONNEMENT ET IMPLÉMENTATION DE QPE

Problème: Soit U un opérateur unitaire d'un espace de Hilbert de dimension $N_b := 2^{n_b}$. Notons $|u_1\rangle, |u_2\rangle, \dots, |u_{N_b}\rangle$ les vecteurs propres associés aux valeurs propres $\lambda_1, \lambda_2, \dots, \lambda_{N_b}$ où λ_j est de la forme $\exp(2\pi i \theta_j)$, c'est-à-dire $U = \sum_j \exp\{2\pi i \theta_j\} |u_j\rangle \langle u_j|$, le but de l'estimation de phase est de construire une approximation de l'opérateur suivant :

$$QPE_U : |u_j\rangle_{n_b} |0\rangle_{n_l} \mapsto |u_j\rangle_{n_b} |\tilde{\lambda}_j\rangle_{n_l}$$

où les $\tilde{\lambda}_j$ sont des approximations de $2^{n_l} \theta_j$ à ϵ_n près avec une probabilité maximale d'erreur η , soit $\mathbb{P}(|2^{n_l} \theta_j - \tilde{\lambda}_j| > \epsilon_n) < \eta$.

2.4.1 • SIMULATION HAMILTONIENNE

2.4.2 • QFT

2.4.3 • VUE D'ENSEMBLE DE L'ALGORITHME

Soit $|\psi\rangle_{n_b}$ l'état initial dans le registre n_b . Sans perte de généralité, supposons que cet état et l'un des états propres de l'opérateur U , soit $|\psi\rangle_{n_b} = |u_j\rangle$ pour un certain $j \in \{1, \dots, N_b\}$. Notons pour la suite $N_l = 2^{n_l}$, la dimension de l'espace de Hilbert engendré par le registre de comptage. Notons $\tilde{\theta} = \lfloor 2^{n_l} \theta \rfloor / 2^{n_l}$ et écrivons $\epsilon_\theta = \theta - \tilde{\theta}$, on a donc $\theta = \tilde{\theta} + \epsilon_\theta = a/2^{n_l} + \epsilon_\theta = 0.a_{n_l} \dots a_2 a_1 + \epsilon_\theta$, où $a_i \in \{0, 1\}$, $a := a_{n_l} \dots a_2 a_1$ et $0 \leq \epsilon_\theta < \frac{1}{N_l}$. Les différentes étapes du General QPEA sont présentées ci-après:

1. **Setup** : Préparer l'état initial: $|\Psi_0\rangle = |\psi\rangle_{n_b} |0\rangle_{n_l}$

2. **Superposition** : Appliquer une opération Hadamard aux n_l qubits du registre de comptage :

$$|\Psi_1\rangle = \frac{1}{\sqrt{N_l}} |\psi\rangle_{n_b} \bigotimes_{i=1}^{n_l} (|0\rangle + |1\rangle) = \frac{1}{\sqrt{N_l}} \sum_{k=0}^{N_l-1} |\psi\rangle_{n_b} |k\rangle_{n_l}$$

3. **Controlled Unitary Operations** ($C-U^{2^j}$) : Appliquer n_l opérations contrôlées $C-U^{2^j}$ avec $0 \leq j \leq n_l-1$ où le qubit de contrôle est le $(n_l - j)$ -ème qubit du registre de comptage et $|\psi\rangle$ est le qubit contrôlé. Ceci revient à appliquer l'opération U^k à $|\psi\rangle_{n_b} |k\rangle_{n_l}$. L'objectif est de repérer les n_l premiers chiffres de θ dans la base de Fourier en utilisant du «phase kick-back» :

$$|\Psi_2\rangle = \frac{1}{\sqrt{N_l}} \sum_{k=0}^{N_l-1} U^k |\psi\rangle_{n_b} |k\rangle_{n_l} = \frac{1}{\sqrt{N_l}} \sum_{k=0}^{N_l-1} \exp(2\pi i \theta k) |\psi\rangle_{n_b} |k\rangle_{n_l}$$

4. **Inverse Fourier Transform** (QTF^\dagger) : Appliquer les opérateurs de rotation comme décrit dans l'algorithme QTF inverse :

$$\begin{aligned} |\Psi_3\rangle &= \frac{1}{\sqrt{N_l}} \sum_{k=0}^{N_l-1} \left(\frac{1}{\sqrt{N_l}} \sum_{j=0}^{N_l-1} \exp(2\pi i (\theta - j/2^{n_l}) k) |\psi\rangle_{n_b} |j\rangle_{n_l} \right) \\ &= \sum_{j=0}^{N_l-1} \left(\frac{1}{N_l} \sum_{k=0}^{N_l-1} \exp(2\pi i (\theta - j/2^{n_l}) k) \right) |\psi\rangle_{n_b} |j\rangle_{n_l} = \sum_{j=0}^{N_l-1} \alpha_j |\psi\rangle_{n_b} |j\rangle_{n_l} \end{aligned}$$

Où $\alpha_j := \frac{1}{N_l} \sum_{k=0}^{N_l-1} \exp(2\pi i (\theta - j/2^{n_l}) k)$. Si $\theta = a/2^{n_l}$ ($\epsilon_\theta = 0$) alors $|\Psi_3\rangle = |\psi\rangle_{n_b} |2^{n_l} \theta\rangle_{n_l} = |\psi\rangle_{n_b} |a\rangle_{n_l}$.

5. **Measurement** : Lorsque $2^{n_l} \theta$ est un entier, la mesure donne θ avec certitude. Dans le cas contraire, la fonction de masse de probabilité aura un sommet au point $2^{n_l} \theta$, ce qu'on peut vérifier si on lance l'algorithme plusieurs fois. Ainsi, on divise le résultat le plus fréquent (ou l'unique) par 2^{n_l} pour obtenir l'estimation de θ .

• Borne supérieur de la probabilité maximale d'erreur

L'état du système après l'étape 4 (QTF^\dagger) peut s'écrire de la façon suivante:

$$|\Psi_3\rangle = \sum_{j=0}^{N-1} \left(\frac{1}{2^{n_l}} \sum_{k=0}^{N-1} \exp\{2\pi i (\theta - j/2^{n_l}) k\} \right) |\psi\rangle_{n_b} |j\rangle_{n_l} = \sum_{j=0}^{N-1} \alpha_j |\psi\rangle_{n_b} |j\rangle_{n_l}$$

Prenons α_l comme l'amplitude du ket $|(a+l)(\text{mod } N_l)\rangle$, où $a = \lfloor 2^{n_l} \theta \rfloor$. D'où on obtient

$$\alpha_l = \frac{1}{2^{n_l}} \sum_{k=0}^{N-1} (\exp\{2\pi i (\theta - (a+l)/2^{n_l})\})^k = \frac{1}{2^{n_l}} \left(\frac{1 - \exp\{2\pi i (2^{n_l} \theta - (a+l))\}}{1 - \exp\{2\pi i (\theta - (a+l)/2^{n_l})\}} \right) = \frac{1}{N_l} \left(\frac{1 - \exp(2\pi i (2^{n_l} \epsilon_\theta - l))}{1 - \exp(2\pi i (\epsilon_\theta - l/2^{n_l}))} \right)$$

Dans ce qui suit, notons $\mathbb{P}(a+l)$ la probabilité que les qubits dans le registre n_b soient dans l'état $|(a+l)(\bmod N_l)\rangle$ et $\mathbb{P}(a+l|\epsilon_\theta \in [\alpha, \beta])$ la probabilité du même évènement conditionnel à ϵ_θ dans l'intervalle $[\alpha, \beta] \subset [0, 1]$.

$$\begin{aligned}\mathbb{P}(a+l) &= |\alpha_l|^2 = \frac{1}{2^{2n_l}} \frac{\sin^2(\pi(2_l^n \epsilon_\theta - (a+l)))}{\sin^2(\pi(\epsilon_\theta - (a+l)/2^{n_l}))} && \text{cas général} \\ \mathbb{P}(a) &= |\alpha_l|^2 = \frac{1}{2^{2n_l}} \frac{\sin^2(\pi(2_l^n \epsilon_\theta - a))}{\sin^2(\pi(\epsilon_\theta - a/2^{n_l}))} && \text{cas particulier } l=0 \\ \mathbb{P}(a+1) &= |\alpha_l|^2 = \frac{1}{2^{2n_l}} \frac{\sin^2(\pi(2_l^n \epsilon_\theta - (a+1)))}{\sin^2(\pi(\epsilon_\theta - (a+1)/2^{n_l}))} && \text{cas particulier } l=1\end{aligned}$$

D'où $\mathbb{P}_{\epsilon_\theta \leq 1/2^{n_l+1}}(a) > 4/\pi^2$ et $\mathbb{P}_{\epsilon_\theta > 1/2^{n_l+1}}(a+1) > 4/\pi^2$, ce qui démontre que la fonction de masse de probabilité a un sommet au point $2^{n_l}\theta$.

• Relation entre la taille du registre quantique est la précision de l'estimation

Prenons $e = \epsilon_n/2^n$ et calculons la probabilité maximale d'erreur. Pour le faire, supposons qu'obtient a^* (au lieu de a) lorsqu'on mesure.

$$\begin{aligned}\mathbb{P}(2^n|\theta - \bar{\theta}| > e) &= \mathbb{P}(|a_1^* - a| > e) = \mathbb{P}(|a_1^* - a| \geq e+1) = \mathbb{P}\left(\left|\frac{a_1^*}{2^{n_l}} - \frac{a}{2^{n_l}}\right| \geq \frac{e+1}{2^{n_l}}\right) \\ &= \sum_{-2^{n-1} < l \leq -(e+1)} |\alpha_l|^2 + \sum_{(e+1) < l \leq 2^{n-1}} |\alpha_l|^2 \\ &\leq \frac{1}{4} \left[\sum_{l=-2^{n-1}+1}^{-(e+1)} \frac{1}{(l-2^n\epsilon_\theta)^2} + \sum_{l=(e+1)+l}^{2^{n-1}} \frac{1}{(l-2^n\epsilon_\theta)^2} \right] \\ &\leq \frac{1}{4} \left[\sum_{l=-2^{n-1}+1}^{-(e+1)} \frac{1}{l^2} + \sum_{l=(e+1)+l}^{2^{n-1}} \frac{1}{(l-1)^2} \right] \leq \frac{1}{2} \sum_{l=e}^{2^{n-1}-1} \frac{1}{l^2} \leq \frac{1}{2} \int_{l=e-1}^{2^{n-1}-1} \frac{dl}{l^2} \leq \frac{1}{2(e-1)}\end{aligned}$$

Supposons que l'on veut approximer θ avec une précision de l'ordre 2^n pour un $n \in \mathbb{N}^*$. D'où $e = 2^{n_l-n} - 1$ et donc la probabilité d'obtenir une telle approximation est supérieur à $1 - 1/2(2^{n_l-n} - 2)$. Ainsi, pour réussir à obtenir θ précis à n bits avec une probabilité d'au moins $1 - \epsilon$, où $\epsilon \in [0, 1[$, on doit utiliser $n_l = n + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits et garder le n premiers chiffres de l'estimateur $\tilde{\theta}_1 = a_1^*/2^{n_l}$.

• Algorithme et circuit quantique QPE

L'algorithme QFT-based QPE ainsi comme son circuit quantique associé sont présentés ci-dessous. Après, une analyse du nombre de portes est réalisée.

Algorithm 2 Quantum Phase Estimation

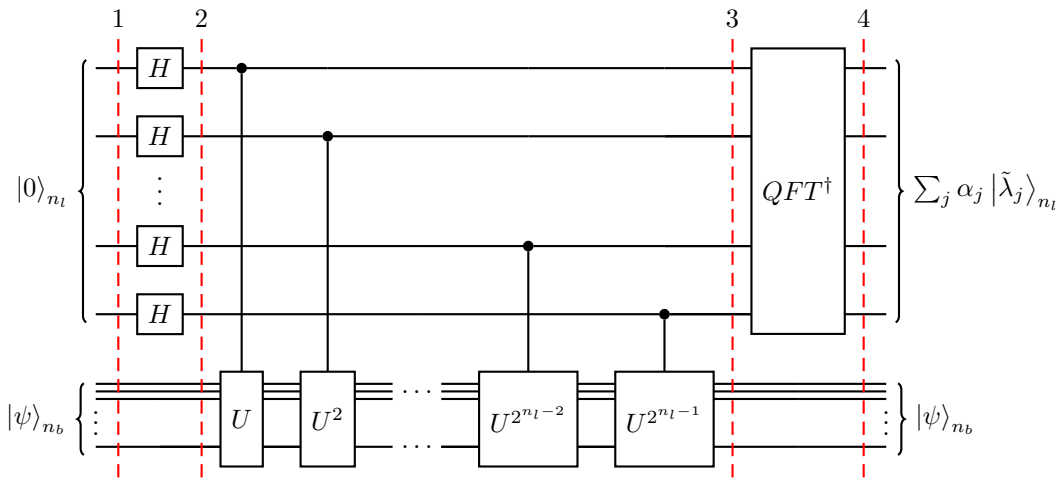
Input:

- (i) une liste de matrices U_j de taille n et un entier r tel que $\tilde{U} = \prod_{i=1}^r \prod_{j=1}^n U_{n-j+1}$ approxime l'opérateur U défini dans cette section.
- (ii) $|b\rangle_{n_b} := \sum_j |u_j\rangle \langle u_j|$, l'état initial préparé, où u_j sont les états propres de U .
- (iii) $n_l = n + \lceil \log \left(2 + \frac{1}{2\eta} \right) \rceil$ qubits initialisés à $|0\rangle$ où η est la probabilité maximale d'erreur choisie et $n = \log(1/\epsilon_n)$ où ϵ_n est la précision souhaitée.

Output : état quantique $|\tilde{\lambda}_j\rangle_{n_l}$ préparé.

procedure

1. $|\psi\rangle_{n_b} |0\rangle_{n_l}$
2. $\rightarrow \frac{1}{\sqrt{N_l}} \sum_{k=0}^{N_l-1} |\psi\rangle_{n_b} |k\rangle_{n_l}$
3. $\rightarrow \frac{1}{\sqrt{2^{n_l}}} \sum_{k=0}^{2^{n_l}-1} \exp(2\pi i \theta k) |\psi\rangle_{n_b} |k\rangle_{n_l}$
4. $\rightarrow \sum_{j=0}^{N_l-1} \left(\frac{1}{N_l} \sum_{k=0}^{N_l-1} \exp(2\pi i (\theta - j/2^{n_l}) k) \right) |\psi\rangle_{n_b} |j\rangle_{n_l} = \sum_{j=0}^{N_l-1} \alpha_l |\psi\rangle_{n_b} |(a+l)(\text{mod } N_l)\rangle_{n_l}$

end procedure


De la même façon que dans la section de qRAM, on supposera que $k = \lceil \log(N) \rceil = \log(N)$ dans ce qui suit. Le nombre de portes quantiques dans chaque étape est

1. $\mathcal{O}(N_l \log(N_l))$ portes CNOTs et $\mathcal{O}(N_l \log(N_l)^2)$ portes à un qubit (chargement de l'état initial par qRAM).
2. $\log(N_l)$ portes Hadamard.
3. $N_l - 1$ portes contrôlées $C - U$. Chacune de ces portes sont équivalentes à 2 portes CNOTs et 4 portes à un qubit. Par conséquent, nous avons $2(N_l - 1)$ portes CNOTs et $4(N_l - 1)$ portes à un qubit.
4. $\log(N_l)$ portes Hadamard et $\log(N_l)(\log(N_l) - 1)/2$ portes de rotations contrôlées. Chaque porte de rotation contrôlée est équivalent à 2 portes CNOTs et 2 portes à un qubit. Ainsi, nous avons $\log(N_l)(\log(N_l) - 1)$ portes CNOTs et $\log(N_l)^2$ portes à un qubit.

L'analyse précédente nous montre que la préparation de l'état initial est la partie la plus coûteuse en termes du nombre de portes. En outre, à l'intérieur de QPE, c'est l'application des portes $C - U$ qui utilise le plus de portes.

Il est intéressant de voir que dans ces deux cas, la performance avec les portes CNOTs et les portes à un qubit est également mauvaise.

Propriété linéaire du schéma avancé

Essentiellement, avant l'étape de mesure le QFT-based QPEA transforme l'état $|0\rangle_{n_l}|\psi\rangle_{n_b}$ de la façon suivante :

$$QPE_U : |0\rangle_{n_l}|\psi\rangle_{n_b} \rightarrow \sum_{j=0}^{2^{n_l}-1} \alpha_j |\tilde{\lambda}_j\rangle_{n_l} |\psi\rangle_{n_b}$$

Où les états $|\tilde{\lambda}_j\rangle_{n_l}|\psi\rangle_{n_b}$ tels que $|\theta - \tilde{\lambda}_j/2^{n_l}| < 1/2^{n_l}$ ont des probabilités dominantes $|\alpha_j|^2$ et donc l'un d'entre eux est observé avec une grande probabilité. En augmentant le nombre de qubits dans le registre de comptage on obtient un bon estimateur $2^{n_l} * \theta$. Il est important de noter que comme U est un opérateur linéaire, la transformation qui vient d'être décrite peut s'appliquer à une superposition de vecteurs propres de U. Ainsi, nous pouvons éviter de préparer un état propre (éventuellement inconnu) au prix d'introduire un caractère aléatoire supplémentaire dans la sortie de l'algorithme.

Soit $|\phi\rangle = \sum_i \gamma_i |u_i\rangle$ un état quelconque développé en termes des vecteurs propres $|u_i\rangle$ de U (à noter que ces états propres ne forment pas nécessairement une base puisque on peut avoir des valeurs propres dégénérées). Ainsi, la transformation avant la mesure est la suivante :

$$QPE_U : |0\rangle_{n_l}|\phi\rangle_{n_b} = \sum_i \gamma_i |0\rangle_{n_l}|u_i\rangle_{n_b} \rightarrow \sum_i \gamma_i \sum_{j=0}^{2^{n_l}-1} \alpha_j |\tilde{\lambda}_j\rangle_{n_l} |u_i\rangle_{n_b}$$

Après mesure dans le registre de comptage on va trouver un estimateur $2^{n_l}\theta_i$ avec une probabilité $|\gamma_i|^2$. Lorsqu'on mesure cet état, d'après les postulats de la mécanique quantique le registre composé par n_b qubits va être dans l'état propre $|u_i\rangle_{n_b}$. C'est cette généralisation qui va nous permettre d'utiliser le QFT-based QPEA dans l'algorithme HHL.

2.4.4 • UN EXEMPLE DÉTAILLÉ

2.5 FONCTIONNEMENT ET IMPLÉMENTATION DE EIGROT

2.5.1 • ARITHMÉTIQUE QUANTIQUE

2.5.2 • ASNINV

2.5.3 • ROTATION CONTRÔLÉE

2.5.4 • VUE D'ENSEMBLE DE L'ALGORITHME

2.5.5 • UN EXEMPLE DÉTAILLÉ

2.6 MESURES

3

ANALYSE DE HHL

4

DÉMARCHE DE PROJET

4.1 TRAVAIL À PLUSIEURS

Dès le début du projet, nous avons cherché à mettre en place un cadre de travail commun. Pour ce faire, nous avons convenu de nous retrouver pour travailler, échanger et faire des points chaque semaine dans une salle du DrahiX réservée pour l'occasion sur le créneau alloué au PSC par notre emploi du temps. Afin de pouvoir dresser un suivi de ces réunions, nous avons rempli chaque semaine un acte de réunion dans lequel figure ordre du jour, déroulement effectif de la réunion, et liste des tâches que chacun doit accomplir pour la semaine suivante. Au cours du PSC, nous avons également eu recours à différents outils formels pour organiser notre travail et en suivre l'évolution, parmi lesquels un diagramme de Gantt et une matrice RACI, deux objets fréquemment utilisés en gestion de projet.

Les réunions hebdomadaires au DrahiX ont permis de poser un socle de travail régulier pour le PSC, qui nous aura été particulièrement utile pour nous approprier les objets manipulés. En effet, le PSC que nous avons mené traite fondamentalement d'une discipline que nous n'avons jamais étudiée auparavant, l'informatique quantique : bien qu'ayant eu le bagage mathématique nécessaire dans notre scolarité antérieure et le bagage physique adéquat avec PHY361 puis PHY430, une pratique régulière des concepts présentés en partie 1 et une réflexion de long terme sur HHL étaient nécessaires pour remplir l'objectif final qui était l'implémentation de HHL.

Le PSC a été ponctué de plusieurs échéances, lors desquelles nous avons tous appris sur le travail de groupe. Par exemple, lors de la soutenance intermédiaire, nous avions prévu de chercher notre partie chacun de notre côté comme nous le faisons jusque-là : mais le jour J, il s'est avéré que nos travaux étaient très hétérogènes et ne se mariaient pas en un oral cohérent et clair. Nous avons après coup pris du recul sur la situation, et identifié quatre problèmes : le premier était de s'être réparti les parties sans avoir défini de cadre de travail commun, le deuxième de ne pas prévoir de mise en commun intermédiaire avant le rendu final, le troisième de ne pas s'être donné de marge sur le rendu (nous avons convenu d'assembler les slides la veille, mais suite à certains retards, elles l'ont été le jour même), et le quatrième de ne pas avoir fait de répétition ensemble pour travailler les enchaînements entre prises de parole et régler les timings.

Enfin, en ces temps de crise, nombre d'élèves ont été en décrochage scolaire. Nous avons donc dû demander l'aide de notre coordinateur pour remotiver l'un d'entre nous avec lequel nous n'arrivions plus à établir de contact : bien qu'étant initialement réticents à cette idée afin de ne pas pénaliser notre camarade, nous avons pu compter sur la bienveillance de M. Pichard pour arranger la situation et ainsi permettre à une implémentation complète de voir le jour.

4.2 RÉPARTITION DES TÂCHES AU SEIN DE L'ÉQUIPE

Nous avons, comme précédemment mentionné, utilisé une matrice RACI au début de notre PSC. Néanmoins, son efficacité n'a été que de courte durée, car nous l'avons écrite pour la répartition des tâches telle que nous

l'avions initialement pensée après seulement deux semaines de travail : or, cette répartition fut sujette à nombre d'évolutions. En effet, nous n'avions initialement pas conscience de bien des aspects qu'il a fallu comprendre et travailler pour parvenir à une compréhension et une implémentation de HHL.

En pratique, c'est donc au fur et à mesure que nous découvrons subtilités et difficultés que la répartition des tâches évoluait. Cette évolution se faisant en écrasante majorité lors des réunions du mercredi, où nous prenions le temps nécessaire pour présenter aux autres nos découvertes de la semaine passée, les actes de réunions se sont avérés être notre principal outil formel de travail pour suivre la répartition des tâches.

4.3 DÉFINITION D'UN FRAMEWORK ADAPTÉ POUR TRAVAILLER À PLUSIEURS SUR L'IMPLÉMENTATION DE HHL

Bien que notre PSC soit rattaché au département MAP, il faisait l'objet d'une forte pluridisciplinarité avec l'informatique, le but final étant de parvenir à écrire un programme à plusieurs. Dès lors, définir en amont de l'écriture de toute ligne de code un framework permettant de travailler à la fois chacun de notre côté et à la fois en tant que membres d'une équipe a été un enjeu majeur de la réussite de notre projet.

Un premier pan de la résolution de ce problème concerne la structure interne du code. Après avoir initialement considéré l'idée d'une programmation orientée-objet pour permettre une modularité requise pour le travail à plusieurs, nous avons rapidement cherché autre chose en constatant que ce cadre de développement ne correspondait pas au type de programme que nous écrivions, à tel point que toute tentative de l'appliquer semblait forcée et rendait les choses inutilement compliquées. C'est finalement une solution conçue spécifiquement pour notre projet que nous avons retenue. L'idée est la suivante : on peut remarquer que Qiskit, SDK d'IBM que nous avons utilisé pour programmer, se construit en première approche autour des instructions de base "créer un circuit quantique", "poser telle porte sur tel(s) qubit(s) de tel circuit", "effectuer la mesure de tel(s) qubit(s) de tel circuit" et "exécuter tant de fois l'algorithme correspondant à tel circuit sur tel ordinateur quantique (virtuel ou réel)". Aussi, nous avons décidé de travailler comme suit :

- Chacun écrit des fonctions prenant en entrée un circuit et des positions de qubits au sein de ce circuit, et plaçant sur ce circuit aux positions données en entrée les portes correspondant à un algorithme donné (QFT, QPE, une opération arithmétique...)
- Il existe une fonction lançant HHL dans sa globalité, qui étant donné un système linéaire à résoudre compatible, crée un circuit de taille adéquate, appelle dans le bon ordre les fonctions précédemment mentionnées pour construire HHL sur ce circuit, demande l'exécution de ce circuit en nombre de fois suffisante, et affiche les résultats. Cette fonction est donc la seule à ne pas prendre de circuit en entrée.
- Pour les imports, tout package pouvant entrer en collision avec un autre (par exemple, numpy avec math) doit être nommé avec sa nomenclature usuelle (par exemple, import numpy as np).
- Toute variable globale ou fonction définie doit avoir un nom explicatif et suffisamment détaillé pour ne pas risquer de rentrer en collision avec le travail d'un autre membre du groupe.

Ceci étant acquis, un deuxième défi à relever a été de trouver une solution pour le partage du code. Notre solution se résume par :

- Tout le code est stocké dans un seul jupyter notebook, où chacun a ses cellules. Ce format a l'avantage de permettre de visualiser facilement la structure globale du programme, tout en définissant clairement l'espace de chacun.

- Pour la gestion du fichier : dans le Google Drive se trouve un dossier Code, contenant à sa racine la dernière version (numérotée dans son titre) du projet et un dossier Old contenant les anciennes versions. Lorsqu'un membre de l'équipe veut faire une modification, il télécharge la dernière version sur son ordinateur, fait la modification hors ligne, incrémente le numéro de version dans le titre, met en ligne la dernière version et déplace l'ancienne version dans Old. Ce faisant, il n'y a pas de risque de pertes de données, et nous avons archive des différentes versions si besoin est de retrouver du code ancien. Un calcul rapide nous a convaincus que notre Drive avait un espace suffisant pour couvrir notre travail sur toute la durée du PSC.

A

PORTES QUANTIQUES

A.1 DÉFINITIONS ET NOTATIONS

Cette section est emprunté de (Carrera Vásquez, 2018),...

A.1.1 • ACTION DE $H^{\otimes n}$ SUR $|x\rangle_n$

Lemma xx. Denotons par \bullet_n le produit scalaire bit à bit. L'action de $H^{\otimes n}$ sur un état à n qubits $|x\rangle_n$ est

$$H^{\otimes n}|x\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (-1)^{k \bullet_n x} |k\rangle_n$$

En particulier, si $|x\rangle_n = |0\rangle_n$ l'application de $H^{\otimes n}$ emmène à une superposition d'états pures :

$$H^{\otimes n}|0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle_n$$

La démonstration de ce résultat est à 2 étapes : la preuve du cas $n = 1$ et ensuite la généralisation au cas général.

Remarque. Lorsqu'il n'y a pas d'ambiguïté, on adoptera l'écriture suivante : $k \cdot x := k \bullet_n x$.

A.1.2 • L'ENSEMBLE STANDARD

Définition xx. L'ensemble standard de portes universelles est constitué des portes Hadamard (H), phase (U_1), controlled-NOT ($CNOT$) et $\pi/8$ (T).

Dans la littérature on retrouve les égalités suivantes :

$$U_1(\lambda) = e^{i\lambda/2} RZ(\lambda)$$

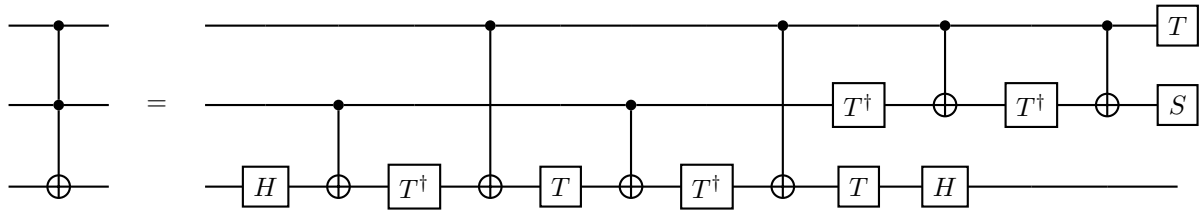
$$T = U_1(\pi/4) = e^{i\pi/8} RZ(\pi/4)$$

Il existent d'autres combinaisons de portes qui donnent un ensemble universel, cependant il y a un intérêt particulier sur l'ensemble standard puisque c'est celui qui est disponible dans le logiciel Qiskit d'IBM utilisé pour implémenter ce projet.

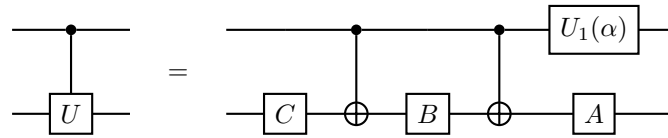
A.1.3 • DÉCOMPOSITION DE PORTES

Dans cette section nous présenterons les décompositions de portes les plus importantes que nous utiliserons dans le projet pour l'analyse du nombre de portes des différents algorithmes. Celles-ci sont tirées du chapitre 4.3. de (Nielsen, 2010).

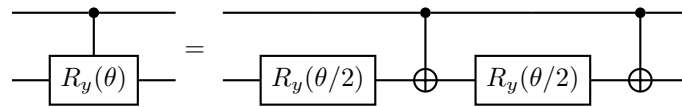
- Toffoli : $6 \cdot \text{CNOTs} + 7 \cdot T + 2 \cdot H + 1 \cdot S = 6 \text{CNOTs} + 10 \cdot \{\text{porte à un qubit}\}$ au total mais avec une profondeur égale à 13.



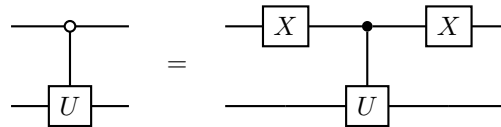
- $C - U$: $2 \cdot \text{CNOTs} + 4 \cdot \{\text{porte à un qubit}\}$ au total avec une profondeur égale à 6. Dans la décomposition de cette porte, le nombre réel α et les opérateurs unitaires A, B et C satisfont $U = e^{i\alpha}AXBXCX$ et $ABC = I$.



- $C - R_y$: $2 \cdot \text{CNOTs} + 2 \cdot \{\text{porte à un qubit}\}$ au total avec une profondeur égale à 4.



- $C_0 - U$: $2 \cdot \text{CNOTs} + 6 \cdot \{\text{porte à un qubit}\}$ au total avec une profondeur égale à 7.



- $C^n - U$ pour $n \geq 2$: $(12n - 10) \cdot \text{CNOTs} + (20n - 16) \cdot \{\text{porte à un qubit}\}$ au total, avec une profondeur égale à $26(n - 1) + 29$ et en utilisant $n - 1$ qubits auxiliaires. Ce résultat peut en découler du fait que cette porte est équivalente à celle montrée dans le circuit ci-dessus pour le cas particulier $n = 5$.

