
Calcul Hybride Quantique-Classique

Algis David



Table des matières

1	Les imperfections des ordinateurs quantiques dans l'ère NISQ	7
1.1	Correction d'erreur	7
1.1.1	Classique	7
1.1.2	Classification des codages	9
1.1.3	Quantique	10
1.2	Fault-tolerance	14
1.2.1	Principe fault-tolerance	14
1.2.2	Concaténation	15
1.2.3	Le théorème de seuil	16
1.3	Mesure de performance d'un circuit quantique	17
1.3.1	Fidélité	17
1.3.2	D'autres données importantes	18
1.3.3	Une métrique : le volume quantique	19
1.4	L'ère NISQ	19
2	Architecture hybride quantique-classique	23
2.1	Simulation des circuits d-croix	23
2.2	Simulation des algorithmes QC	24
2.2.1	Réseaux de tenseurs	25
2.2.2	Marche à suivre pour la simulation	27
3	Application résolution de système linéaire hybride	31
3.1	L'algorithme HHL□	31
3.1.1	Un rapide aperçu de l'algorithme	31
3.1.2	Exemple de résolution linéaire	35
3.1.3	Contraintes de l'algorithme	37
3.1.4	Analyse de la complexité et amélioration	38
3.2	Algorithme de résolution de système linéaire hybride	39

3.2.1	L'algorithme de Neumann-Ulam	39
3.2.2	l'algorithme quantique	42
Appendice		43
3.3	Théorème de non-clonage	43
3.4	État pure et état mixte	43
3.5	Porte Quantique SWAP	44
3.6	Théorème de Schmidt	44
Formulaire		47
Bibliographie		52

Introduction

Dans un premier temps nous étudierons les limitations des ordinateurs quantiques actuelle, dans un second temps nous essaierons de contourner modestement cette limite par l'hybridation. Et enfin nous donnerons des applications à cette hybridation.

Une implémentation des différents algorithmes sont disponible sur https://colab.research.google.com/drive/1SwCD135C1qCnZX_160cLRP7zlc1jlBUZ, les sections notés d'un carré □, indique que l'algorithme a été implémenté sur le colab.

In spite of dramatic improvements of quantum hardware during these last decades, the number of qubits on a chip, gate fidelities, and qubit coherence times still remain modest in the near future while quantum algorithms require deep quantum circuits involving hundreds of qubits to solve problems that are intractable for classical computers.

Therefore, in an intermediate approach, we propose to explore a hybrid quantum-classical architecture that allows the tradeoff between quantum and classical computational resources to support the evaluation of large quantum circuits by partitioning them into sub-circuits to be run concurrently on quantum and classical processors.

This work can be decomposed into the following main tasks suitable to a 4 PM internship :

- Design of a hybrid quantum-classical architecture ;
- Compilation of quantum algorithms into quantum circuits ;
- Partition of quantum circuits into suitable sub-circuits ;
- Scheduling and execution of sub-circuits onto the hybrid architecture.

Chapitre 1

Les imperfections des ordinateurs quantiques dans l'ère NISQ

Actuellement, les ordinateurs quantiques font face à de multiples imperfections. L'une de ces imperfections est le bruit, qui génère des erreurs sur les circuits quantiques. Nous allons au cours des deux premières sections, donner des outils pour résoudre ces problèmes. Cependant, en étudiant ces outils, nous verrons dans les deux sections suivantes, qu'à l'heure actuelle il est très compliqué, voire impossible de les mettre en pratique.

Pour écrire ce chapitre, nous nous sommes en grande partie inspirés du chapitre 10 de [28], ainsi que l'article [35], et dans une moindre mesure du chapitre 8 de [23].

1.1 Correction d'erreur

En général, lorsque l'on considère des problèmes de calcul quantique, on suppose que l'interaction avec l'environnement est négligeable, que les portes quantiques sont appliquées parfaitement, etc... Cependant dans la pratique, ces hypothèses sont loin d'être négligeables.

Dans cette section, on n'étudiera pas les sources de ces erreurs, qui sont plutôt d'ordre physique, mais les modèles mathématiques pour approximer ces erreurs, et surtout des codes pour pouvoir les corriger.

Ainsi, dans la suite on considère qu'il existe des zones *bruitées* qui peuvent potentiellement provoquer des erreurs sur le circuit. Ces erreurs satisfont un modèle mathématique que l'on détaillera à chaque fois.

Dans un premier temps, nous allons introduire le modèle mathématique derrière la correction d'erreur classique, à travers un exemple sur un code de correction à 3 bits. Nous verrons dans un second temps que les concepts utilisés pour ce modèle se généralisent pour la correction d'erreur quantique.

1.1.1 Classique

Avant de débiter cette section, donnons une rapide définition d'un code correcteur qu'il soit classique ou quantique.

Un code correcteur est un algorithme découpé en plusieurs étapes, qui permet de détecter et de corriger les erreurs dans un message. Ce message peut être une suite de bits, une suite de qubits, une suite de lettres, etc...

Dans ce rapport, nous n'aborderons que très brièvement la théorie des codes. Le lecteur curieux pourra consulter le livre [17].

Avant tout chose pour pouvoir appliquer un code correcteur, il faut une source d'erreur. Cette source

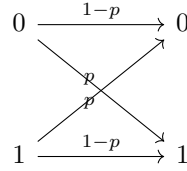


FIGURE 1.1 – modèle bit-flip

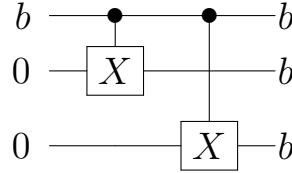


FIGURE 1.2 – On met les bits auxiliaire dans l'état b

d'erreur sera décrite par la modèle mathématique suivant :

Définition 1 (modèle bit-flip). *Dans un registre de n bits, un bit dans un état quelconque b indépendamment des autres bits a une probabilité :*

- $1 - p$ de rester inchangé.
- p de changer dans l'état $X(b)$.¹

(cf. figure 1.1 pour le modèle bit-flip sur un bit)

Remarque 1. *Ce modèle est l'un des plus simpliste. En effet, on aurait pu imaginer un modèle dans lequel la probabilité de changer d'état dépend de l'état d'origine ou de l'état des autres bits, etc... Cependant il donne déjà une bonne approximation de la réalité.*

Supposons que l'on ait un bit dans un état b que l'on souhaite protéger des erreurs. Pour cela nous allons procéder par étape :

1. Nous allons ajouter un certain nombres de bits auxiliaires (dans notre exemple 2 bits auxiliaires sont ajoutés), au bit à protéger, que nous mettrons dans le même état que le bit à protéger. Ceci ajoute de la redondance au bit à protéger. Ainsi, en cas de corruption des données, on possèdera suffisamment de données pour pouvoir détecter si le bit à protéger a été corrompu. De plus, on va appliquer des portes quantiques à ces qubits pour les mettre dans l'état b . On dit qu'on *encode* l'information.
2. Ensuite les bits passent dans la zone bruitée, c'est à dire qu'on applique le modèle bit-flip à ces bits.
3. On applique un circuit pour déterminer quel bit à été corrompu et en fonction, on restaure le bit à protéger dans son état initial. On dit qu'on *décode* l'information.

Voici, donc un exemple de code correcteur classique :

Exemple. 1. *Ici on ajoute 2 bits auxiliaire dans l'état 0 au bit à protéger. On souhaite les mettre, dans l'état b . Pour cela on applique une porte conditionnelle CNOT classique au 2 bit (cf. figure 1.2)*

2. *Supposons sans perte de généralité que $b = 0$. Maintenant on applique le modèle bit-flip à nos 3 bits dans l'état 000. On obtient l'arbre de probabilité suivant figure 1.3.*

Pour restaurer l'information corrompue, on suppose que la probabilité qu'un bit soit corrompu est nulle. Cette supposition n'est pas si gênante dans la pratique, en effet la probabilité que chaque bit soit corrompu et donc qu'on obtiennent 111 est p^3 , or p est généralement petit, et si on utilise n bit auxiliaire la probabilité que chaque bit soit corrompu est $p^{n+1} \sim 0$, donc négligeable.

1. De façon équivalente $b \mapsto X(b)$.

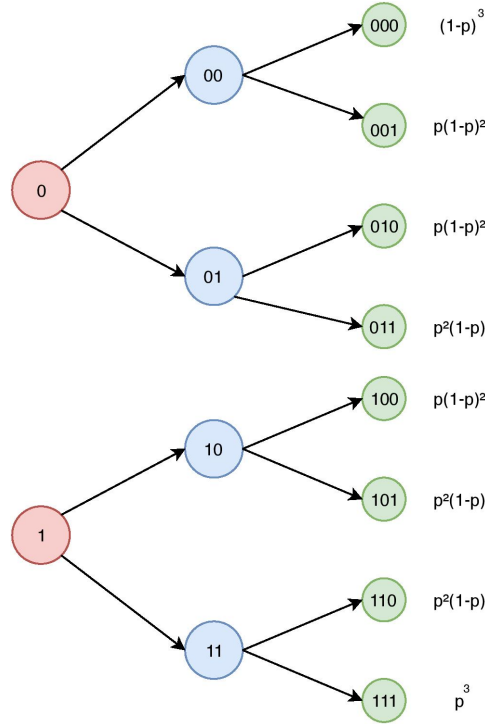


FIGURE 1.3 – arbre de probabilité du modèle bit flip flip sur l'état 000

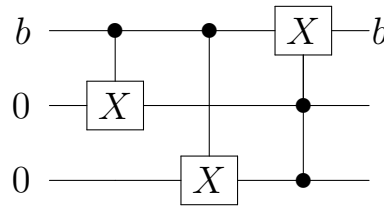


FIGURE 1.4 – Le circuit pour la correction d'erreur classique - modèle bit flip

3. Notons b_{err} le résultat obtenu après l'étape 2.

Maintenant, nous allons appliquer les mêmes opérations qu'à l'étape 1, afin de regarder la parité des bits. En effet, si les valeurs des bits auxiliaires sont les mêmes que celles du bit à protéger, alors après la porte CNOT ils seront dans l'état 0 sinon dans l'état 1. Puis on applique une porte de Toffoli² avec les bits auxiliaires en contrôle et le bit à protéger en cible. Donc si les deux bits auxiliaires sont dans l'état 1, c'est qu'au moins le bit à protéger a été corrompu, et donc avec la porte de Toffoli, on le restaure. (cf. figure 1.4)

Remarque 2. Après l'application du code de correction, un bit n'a plus une probabilité p de changer d'état, mais $3p^2 - 2p^3$.

1.1.2 Classification des codages

Ce code (cf. section 1.1.1), n'est qu'un code parmi d'autres. C'est pourquoi on aimerait donner des quantités pour différencier les codes correcteurs. Les deux premières quantités viennent naturellement sont :

- le nombre de bits nécessaires pour l'encodage noté n .
- le nombre de bits que l'on souhaite protéger noté k .

2. La porte de Toffoli prend 2 bits en contrôle et un bit en cible. L'état du bit en cible est changé uniquement, si les 2 bits de contrôle sont dans l'état 1, sinon l'état du bit cible reste inchangé.

Par exemple, le bit-flip utilisait ci-dessus nécessite 3 bits pour l'encodage, donc $n = 3$, et protéger un seul bit donc $k = 1$ ³

Cependant, on souhaiterait une autre quantité d'ordre qualitatif, capable de caractériser le nombre d'erreurs que le code sera en mesure de corriger. C'est pourquoi on définit la *distance de Hamming*.

Proposition (Distance de Hamming). *Soit x et y deux ensembles de n bits. Alors on définit la distance de Hamming entre ces deux bits comme :*

$$d_H(x, y) = \text{card}(\{i \in \{1, \dots, n\} \mid x_i \neq y_i\}) \quad (1.1)$$

Exemple. *Afin de mieux appréhender la définition qui peut sembler quelque peu abstraite, donnons un premier exemple.*

Soit $x = 00001$ et $y = 10010$, alors $d_H(x, y) = 3$. En effet trois 3 bits sont différents.

Démonstration. Montrons que d_H définit bien une distance :

- La distance est bien évidemment symétrique, car \neq est symétrique.
- Trivialement, l'égalité $d_H(x, y) = 0$, implique que $x = y$ et réciproquement.
- Soit x, y et z trois ensembles de n bits. Procédons par récurrence sur n le nombres de bits. L'initialisation est triviale. Supposons $d(x, z) \leq d(x, y) + d(y, z)$ vraie à l'ordre n , on rajoute un bit à x, y et z qu'on note respectivement $\tilde{x} = (x, x_{n+1})$, $\tilde{y} = (y, y_{n+1})$ et $\tilde{z} = (z, z_{n+1})$. Il y a 4 cas à considérer, on raisonne uniquement sur un seul d'eux, car le raisonnement est identique pour les 3 autres. Soit $x_{n+1} \neq y_{n+1} = z_{n+1}$ on a $d(\tilde{x}, \tilde{z}) = d(x, z) + 1$, car $x_{n+1} \neq z_{n+1}$, $d(\tilde{x}, \tilde{y}) = d(x, y) + 1$, car $x_{n+1} \neq y_{n+1}$, et $d(\tilde{y}, \tilde{z}) = d(y, z)$, car $y_{n+1} = z_{n+1}$. Ainsi on a toujours l'inégalité $d(\tilde{x}, \tilde{z}) \leq d(\tilde{x}, \tilde{y}) + d(\tilde{y}, \tilde{z})$.

□

Avec la distance de Hamming on peut définir la 3ème quantités qui nous intéresse, qui correspond à

$$\min_{x, y \in \text{bit après codage}} d_H(x, y) \quad (1.2)$$

On note d cette quantité.

Remarque 3. *Nous avons donné toutes ces définitions pour des bits, cependant elle se généralise aisément pour des qubits.*

La distance, nous donne le nombre d'erreurs qui peuvent être corrigées par le code qu'on note $t = \lfloor \frac{d-1}{2} \rfloor$

Pour conclure, dans la suite du rapport on caractérisera un code correcteur pour le triplet $\llbracket n, k, d \rrbracket$.

1.1.3 Quantique

En calcul quantique la correction d'erreur est bien moins aisée qu'en classique. Par exemple, la définition d'un modèle d'erreur même simpliste peut sembler beaucoup moins trivial. En effet en classique un bit ne peut qu'être dans deux états, donc le modèle bit-flip (définition 1) est plutôt naturel, car il existe un nombre discret de transformations possibles. Cependant, en quantique il existe une infinité de transformations possibles. (cf. figure 1.5)

Considérons un système d'un qubit dans l'état $|0\rangle$ et de son environnement dans l'état $|E\rangle$. Comme l'affirment les postulats de la mécanique quantique, l'interaction entre se qubit et son environnement sera caractérisée par une transformation unitaire. C'est à dire qu'on a :

$$|0\rangle |E\rangle \mapsto \beta_1 |0\rangle |E_1\rangle + \beta_2 |1\rangle |E_2\rangle \quad (1.3)$$

Donc après mesure, le qubit passe dans l'état $|0\rangle$ avec une probabilité β_1 et l'environnement devient $|E_1\rangle$, sinon il passe dans l'état $|1\rangle$ avec une probabilité β_2 et l'environnement devient $|E_2\rangle$.

3. Dans ce rapport on utilisera uniquement des codes avec $k = 1$.

Classique	Quantique
$\begin{cases} b \mapsto I(b) \\ b \mapsto X(b) \end{cases}$	$\begin{cases} \psi\rangle \mapsto I(\psi\rangle) \\ \psi\rangle \mapsto X \psi\rangle \\ \psi\rangle \mapsto R_k \psi\rangle \\ \vdots \end{cases}$

FIGURE 1.5 – Difficulté supplémentaire en calcul quantique

De façon similaire pour un qubit dans l'état $|1\rangle$ on a :

$$|1\rangle |E\rangle \mapsto \beta_3 |1\rangle |E_3\rangle + \beta_4 |0\rangle |E_4\rangle \quad (1.4)$$

Et on peut aisément généraliser pour un qubit dans un état quelconque $\alpha_0 |0\rangle + \alpha_1 |1\rangle = |\psi\rangle$ par linéarité :

$$(|\psi\rangle) |E\rangle \mapsto \alpha_0(\beta_1 |0\rangle |E_1\rangle + \beta_2 |1\rangle |E_2\rangle) + \alpha_1(\beta_3 |1\rangle |E_3\rangle + \beta_4 |0\rangle |E_4\rangle) \quad (1.5)$$

Néanmoins, cette écriture ne nous convient pas parfaitement. En effet, on souhaite retrouver une expression comme le bit-flip de la section précédente c'est à dire une combinaison linéaire d'opérateurs appliqués à $|\psi\rangle$. On fait donc un changement de base, dans la base des opérateurs (I, X, Z, XZ) . On en déduit très naturellement l'égalité suivante :

$$\begin{aligned} \alpha_0 \beta_1 |0\rangle |E_1\rangle + \alpha_0 \beta_2 |1\rangle |E_2\rangle + \alpha_1 \beta_3 |1\rangle |E_3\rangle \\ + \alpha_1 \beta_4 |0\rangle |E_4\rangle &= (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \frac{1}{2} (\beta_1 |E_1\rangle + \beta_3 |E_3\rangle) \\ &= (\alpha_0 |0\rangle - \alpha_1 |1\rangle) \frac{1}{2} (\beta_1 |E_1\rangle - \beta_3 |E_3\rangle) \\ &= (\alpha_0 |1\rangle + \alpha_1 |0\rangle) \frac{1}{2} (\beta_2 |E_2\rangle + \beta_4 |E_4\rangle) \\ &= (\alpha_0 |1\rangle - \alpha_1 |0\rangle) \frac{1}{2} (\beta_2 |E_2\rangle - \beta_4 |E_4\rangle) \end{aligned}$$

C'est à dire finalement :

$$|\psi\rangle |E\rangle \mapsto I|\psi\rangle \frac{1}{2} (\beta_1 |E_1\rangle + \beta_3 |E_3\rangle) + Z|\psi\rangle \frac{1}{2} (\beta_1 |E_1\rangle - \beta_3 |E_3\rangle) \quad (1.6)$$

$$+ X|\psi\rangle \frac{1}{2} (\beta_2 |E_2\rangle + \beta_4 |E_4\rangle) + XZ|\psi\rangle \frac{1}{2} (\beta_2 |E_2\rangle - \beta_4 |E_4\rangle) \quad (1.7)$$

Cette expression est bien plus simple à comprendre que celle de l'équation (1.5). Donnons son interprétation :

Définition 2 (modèle erreur quantique). *Dans un registre de n qubits, un qubit quel que soit son état $|\psi\rangle$ indépendamment des autres qubits a une probabilité :*

- $\beta_1^2 + \beta_3^2$ de rester inchangé.
- $\beta_1^2 - \beta_3^2$ de changer dans l'état $Z|\psi\rangle$.
- $\beta_2^2 + \beta_4^2$ de changer dans l'état $X|\psi\rangle$.
- $\beta_2^2 - \beta_4^2$ de changer dans l'état $XZ|\psi\rangle$.

Remarque 4. *L'intérêt de cette définition, est que d'un nombre infini de transformation possible, on a réussi à les réduire à un nombre fini.*

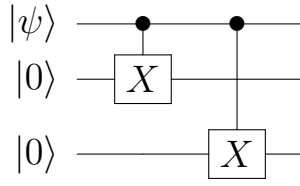


FIGURE 1.6 – On encode le qubits a protéger avec les qubits auxiliaires

Remarque 5. Si on compare, par rapport à la définition 1 du modèle bit flip classique, on remarque que deux termes se sont ajoutés. $Z|\psi\rangle$ qu'on appelle un phase-flip, et le second $XZ|\psi\rangle$ qui est une composition du bit-flip et phase-flip.

Remarque 6. La définition 2, illustre un exemple d'état mixte (cf appendice 3.4 pour une définition d'un état mixte). En l'occurrence on a l'opérateur suivant $\rho \xrightarrow{\mathcal{E}} \mathcal{E}(\rho)$, qui envoie un état mixte sur un autre état mixte $\mathcal{E}(\rho)$ donné par l'opérateur de densité suivant :

$$\mathcal{E}(\rho) = (\beta_1^2 + \beta_3^2)I|\psi\rangle\langle\psi|I + (\beta_1^2 - \beta_3^2)Z|\psi\rangle\langle\psi|Z + (\beta_2^2 + \beta_4^2)X|\psi\rangle\langle\psi|X + (\beta_2^2 - \beta_4^2)XZ|\psi\rangle\langle\psi|XZ \quad (1.8)$$

Nous reviendrons plus tard dessus dans la section 1.3.1.

On a donc donné pour un qubit un modèle d'erreur "général"⁴. Pour mieux comprendre ce modèle, nous allons dans un premier temps considérer un cas particulier de ce modèle avec uniquement le *bit-flip*, puis dans un second temps uniquement le *phase-flip*. Enfin nous allons discuter de plusieurs cas plus généraux.

Le bit-flip code à 3 qubits □

On s'intéresse donc à un cas particulier de la définition 2. Le cas où le qubit a une certaine probabilité de changer dans l'état $X|\psi\rangle$ et sinon de rester inchangé. On cherche à annuler les terme avec du $Z|\psi\rangle$ dans l'équation 1.6. C'est pourquoi on pose $\beta_1|E_1\rangle = \beta_3|E_3\rangle$ et $\beta_2|E_2\rangle = \beta_4|E_4\rangle$.

Maintenant que nous avons clairement défini le modèle d'erreur pour le bit-flip quantique. Nous allons nous attaquer à l'algorithme en lui-même. Celui-ci n'est pas très différent de l'algorithme décrit dans la section 1.1.1.

De même supposons que l'on ait un qubit dans un état quelconque $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ que l'on souhaite protéger d'une zone bruitée. Procédons par étape (figure 1.8) :

1. Tout d'abord, on ajoute deux qubits auxiliaires pour encoder le qubit à protéger. On serait tenté de reproduire ce qui a été fait dans la section 1.1.1 et de faire deux copie de l'état du qubit à protéger. Cependant on sait d'après le théorème de non-clonage (cf. 3.3) que c'est impossible. On applique donc deux portes CNOT sur les deux qubits auxiliaire par rapport au qubit à protéger en contrôle pour encoder le système (cf. figure 1.6). On obtient donc un système dans l'état $\alpha_0|000\rangle + \alpha_1|111\rangle$.
2. Ensuite les qubits passe dans la zone bruitée, c'est à dire qu'on applique le modèle bit-flip quantique décrit ci-dessus. On représente cette zone par la porte quantique figure 1.7. On suppose aussi que la possibilité que plus d'un qubit est subi une erreur soit négligeable.
3. Maintenant on va décoder l'information en appliquant les opérations inverses de l'encodage pour décoder. De plus pour pouvoir récupérer l'état du qubit à protéger on applique une porte de Toffoli⁵, qui restaure l'information dans le cas où l'état le qubit a été corrompu.

Remarque 7. Il existe d'autres façons de faire le bit-flip code à 3 qubits. On a privilégié ici, dans un intérêt pédagogique, la façon la plus simple de faire. Cependant, cette version du bit-flip code à 3 qubits

4. Il n'est pas complètement général, dans le sens où on suppose que pour un registre de plusieurs qubits, l'erreur n'est pas corrélée entre les différents qubits. On considère chaque qubits indépendamment des autres.

5. On raisonne de la même façon que dans la section 1.1.1.

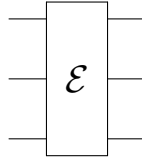


FIGURE 1.7 – Porte représentant la zone bruitée

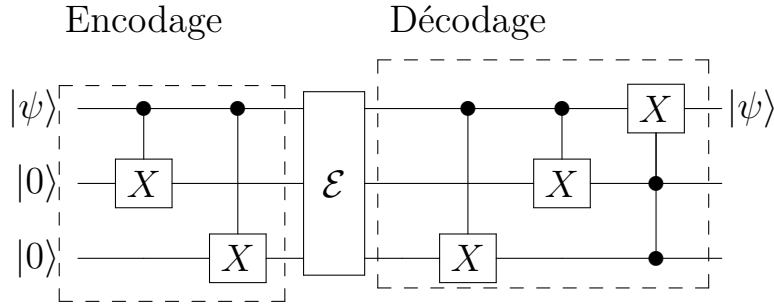


FIGURE 1.8 – bit-flip code à 3 qubits

"n'est pas la meilleure"⁶ par exemple la porte de Toffoli qu'on utilise dans notre circuit, n'est pas simple à implémenter, on lui préfère des portes agissant sur un ou deux qubits.

Une autre façon de concevoir le bit-flip code à 3 qubits (figure 1.9), est d'ajouter deux autres qubits auxiliaires et une étape qu'on appelle récupération d'information syndrome. Celle-ci correspond à un teste de la parité des 3 premiers qubits en utilisant des portes CNOT, puis de la mesure des 2 qubits ajouter. Enfin on enlève l'étape de décodage, à la place on applique des portes X en fonction du résultat de la mesure des deux qubits. Cette méthode a aussi l'avantage de restaurer l'information sur les deux qubits auxiliaire, qui peuvent être utilisés par la suite.

6. Le "n'est pas la meilleure" se quantifie mathématiquement par ce qu'on appelle l'entropie du code correcteur, cette quantité permet d'évaluer si un codage est plus performant qu'un autre. Pour plus de détail sur l'entropie on redirigera le lecteur vers [8]

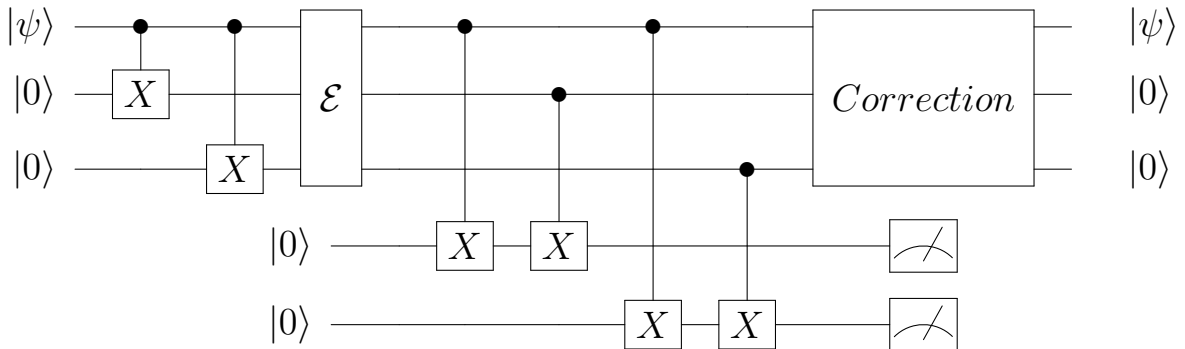


FIGURE 1.9 – bit-flip code à 3 qubits - 2nd méthode

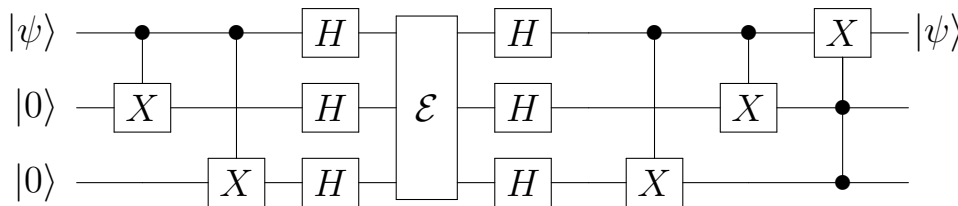


FIGURE 1.10 – phase-flip code à 3 qubits

Le phase-flip code à 3 qubits□

Un autre cas particulier de la définition 2. Le cas où le qubit a une certaine probabilité de changer dans l'état $Z|\psi\rangle$, et sinon de rester inchangé. On cherche à annuler les termes avec du $X|\psi\rangle$ dans l'équation 1.6. C'est pourquoi on pose $\beta_1|E_1\rangle = -\beta_3|E_3\rangle$ et $\beta_2|E_2\rangle = \beta_4|E_4\rangle$. Dans ce cas, ce modèle d'erreur n'a pas d'équivalent classique, contrairement au *bit-flip* code, en effet la phase est une notion purement quantique. Cependant, nous allons voir que les deux modèles sont les mêmes à un changement de base près.

Plaçons nous dans la base $(|+\rangle, |-\rangle)$, dans cette base la porte Z envoie $|+\rangle$ sur $|-\rangle$ et inversement. Comme la porte X envoyait $|0\rangle$ sur $|1\rangle$ dans la base canonique. Ainsi, l'unique différence avec l'algorithme du *bit-flip code à 3 qubits* sera d'appliquer la porte d'Hadamard⁷ et son inverse, c'est à dire elle même, avant et après la zone bruitée. On a donc le circuit représenté figure 1.10.

Codes plus généraux

Les deux codes précédents ne représentaient que des cas particuliers d'erreurs. Pour traiter le cas d'une erreur quelconque de la définition 2, il existe plusieurs codes plus généraux, notamment celui de Shor développé en 1995 dans [41].

Le code de Shor, est un mélange du bit-flip et du phase-flip étudié ci-dessus.

Comme le code est plus général, il nécessite un plus grand nombres de qubits auxiliaires, en l'occurrence 9 qubits, il est décrit par le triplet $[[9, 1, 3]]$. Cependant il existe des codes plus récents nécessitant uniquement 7 qubits comme le code de Steane caractérisé $[[7, 1, 3]]$ et même 5 qubits⁸ décrit cette fois par $[[5, 1, 3]]$, notamment celui introduit dans l'article [31], qui a été implémenté l'année dernière en 2018 [32]. Nous ne développerons aucun de ses codes ici, car les difficultés supplémentaires sont d'ordres technique.

1.2 Fault-tolerance

Dans cette section, nous allons définir le concept de fault-tolerance qui donne une façon de d'éviter la propagation des erreurs au sein du circuit. Puis en utilisant la technique de concaténation, nous allons démontrer le théorème du seuil qui est un résultat remarquable et rassurant sur les possibilités futures des ordinateurs quantiques.

1.2.1 Principe fault-tolerance

Exceptionnellement, dans cette section nous utiliserons le terme anglais *fault-tolerance* unanimement utilisé par la communauté.

Le principe du *fault-tolerance* est d'utiliser des circuits qui ne génèrent pas des erreurs en cascade. Par

7. On rappellera au lecteur que la porte d'Hadamard est celle qui envoie la base $(|0\rangle, |1\rangle)$ sur $(|+\rangle, |-\rangle)$ et inversement.

8. Il a été démontré que 5 qubits est le minimum que l'on puisse faire.

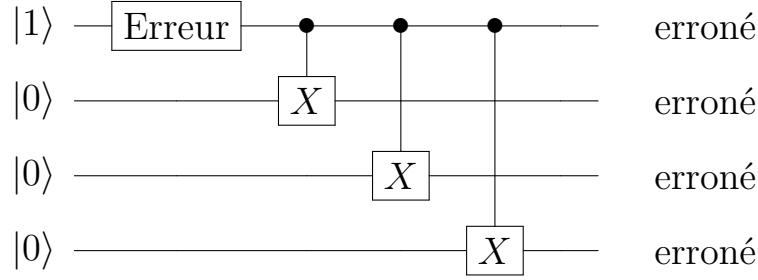


FIGURE 1.11 – Exemple : d'erreur en cascade

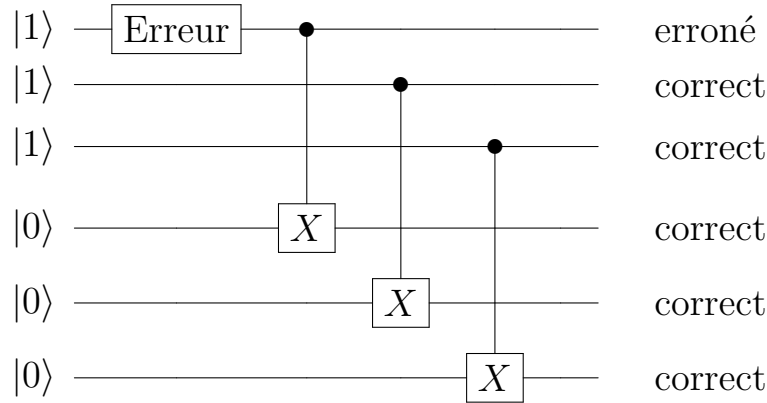


FIGURE 1.12 – Contre-exemple : d'erreur en cascade

exemple, considérons un circuit simple qui applique une porte CNOT à 3 qubits, avec un qubit de contrôle (cf. figure 1.11). Dans ce cas, si qubit en contrôle a subi une erreur, tous les autres qubits seront impactés. On a donc un phénomène d'erreurs en cascade.

Contrairement, à un circuit où la porte CNOT est appliqué au même 3 qubits, mais en contrôle on rajoute deux qubits dans le même état que le premier (cf. figure 1.12). Ce circuit, n'engendre pas d'erreurs en cascade.

Avec ces exemples, on peut donner une définition plus précise d'un circuit *fault-tolerant*.

Définition 3 (Circuit *fault-tolerant*). *On dit qu'un circuit est fault-tolerant, si une unique erreur peut causer au plus une erreur dans le résultat du circuit.*

1.2.2 Concaténation

Un outil important pour la réduction du taux d'erreurs est concaténation, que nous utiliserons dans la section suivante.

La concaténation se base sur le questionnement très simple de se demander « que se passe-t-il si on procède à des corrections d'erreurs de corrections d'erreurs? ».

Nous donnons un première élément de réponse dans cette section à travers un exemple. Et le théorème du seuil abordé dans la section suivante répondra complètement à la question.

Exemple. *Dans cet exemple, nous allons utiliser le code de Steane $[[7, 1, 3]]$.*

Partons d'un qubit que l'on souhaite protéger. On l'encode une première fois avec 6 autres qubits. Puis on ré-applique sur chacun de ses qubits un nouvelle encodage avec pour chacun, 6 autre qubits, on dit qu'on

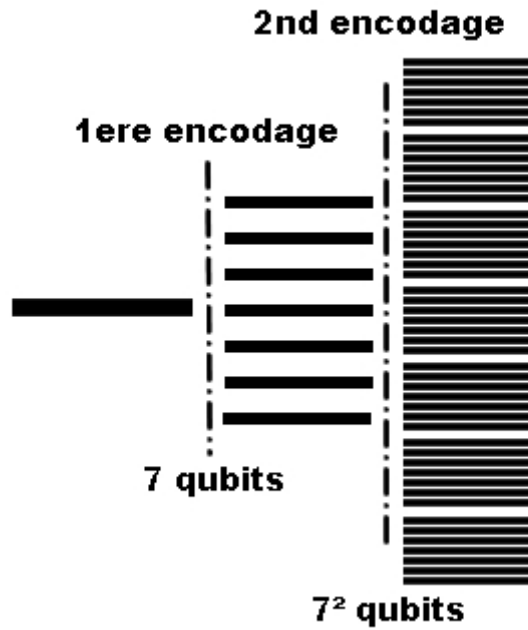


FIGURE 1.13 – Exemple concaténation avec le code de Steane

a un second niveau de correction d'erreur. (cf. figure 1.13) Ainsi on utilise non plus 7 mais 7^2 qubits. En contrepartie la distance de notre code augmente aussi de 3 à 3^2 , et de fait le nombre d'erreurs que notre code peut corriger passe de 1 à $\lfloor \frac{d^2-1}{2} \rfloor = \lfloor \frac{9-1}{2} \rfloor = 4$.

Dans notre exemple on a fait 2 encodages, cependant on peut imaginer l'enchaînement de m encodage, dans ce cas pour un code décrit par $\llbracket n, k, d \rrbracket$, on utiliserait n^m qubits, mais cela corrigerait $\lfloor \frac{d^m-1}{2} \rfloor$ erreurs.

1.2.3 Le théorème de seuil

La conclusion de cette section porte sur un théorème remarquable du fault-tolerance qui va utiliser la concaténation, le *théorème du seuil*. Grossièrement celui-ci nous dit qu'avec un grand nombre de qubits, on peut approcher un taux d'erreurs arbitrairement petit pour les portes quantiques.

Soit un circuit, auquel à chaque porte est appliquées le modèle d'erreur de la définition 2. Où on pose que la probabilité que X et/ou Z soit appliqué est p . On suppose que les portes sont appliquées de façon à ce que le circuit soit *fault-tolerant*. De plus après chaque porte élémentaire au circuit, on applique un code correcteur $\llbracket n, k, 3 \rrbracket$.

Théorème 1 (du seuil). *Soit un circuit dans les conditions données ci-dessus, soit c la borne supérieure du nombre de 2 erreurs possible. Supposons que $cp < 1$, alors le taux d'erreur de l'encodage peut être rendu arbitrairement petit en rajoutant plusieurs niveaux de concaténation.*

Remarque 8. *De façon plus vulgaire, on peut dire qu'on corrige les erreurs plus rapidement qu'elles n'apparaissent.*

Démonstration. Sous les conditions du théorème du seuil, on remarque qu'après un premier niveau d'encodage, la probabilité qu'une porte protégée soit altérée est donnée par $p_P^1 = cp^2$. Où le 1 de p_P^1 correspond au niveau d'encodage, et P que la porte est protégé.

Maintenant, on concatène avec un second niveau de code correction avec le même code $\llbracket n, k, 3 \rrbracket$ sur chacune des portes à protéger. Dans ce cas, la la probabilité qu'une porte protégé soit altérée est donnée par $p_P^2 = c(p_P^1)^2$.

On continue de manière itérative en concaténant, avec g niveau de code correction, toujours avec le code correction $[[n, k, 3]]$. On a :

$$p_P^g = \frac{(cp)^{2^g}}{c} \quad (1.9)$$

Ainsi si $cp < 1$, on a bien que l'erreur peut être arbitrairement petite. \square

1.3 Mesure de performance d'un circuit quantique

Lorsque l'on souhaite comparer deux ordinateurs quantiques entre eux, la première idée spontanée est de comparer le nombre de qubits disponibles dans chacun d'entre eux. Cependant, tout comme il serait absurde de comparer uniquement deux ordinateurs classiques sur leur quantité de mémoire vive par exemple. Avec des ordinateurs quantiques, il est absurde d'utiliser uniquement le nombre de qubits pour les comparer.

Par exemple, prenons deux ordinateurs quantiques :

- Un premier avec un milliers de qubits mais très sensible au bruit, et dans lequel l'application de chaque portes à de très grande chance de provoquer une erreur.
- Un second avec une centaine de qubits très peu sensible au bruit.

Dans ce cas le second, sera toujours "meilleur" que le premier.

C'est pourquoi dans cette section, nous allons introduire plusieurs données afin de nous donner d'autres outils de comparaison que le nombre de qubits. Enfin, nous définirons une métrique au travers du *volume quantique*. Il nous donne une "note" pour évaluer les performances de l'ordinateur quantique en fonction de nombreuses caractéristiques, au même titre que *l'indice de performance de windows* par exemple.

1.3.1 Fidélité

Une première quantité qui semble pertinente mathématiquement d'introduire, quant aux sections précédentes, est la notion de *fidélité*. Néanmoins dans ce rapport il ne nous sera pas nécessaire d'utiliser la définition générale de la *fidélité*, mais uniquement un cas particulier, qui mesure "l'efficacité d'une porte". Le lecteur qui souhaite plus d'information sur la *fidélité* pourra lire le chapitre 9 de [23].

Définition 4. On définit la fidélité F entre un état pur $|\psi\rangle$ et un état mixte⁹ représenté par son opérateur de densité ρ par :

$$F(|\psi\rangle, \rho) = \sqrt{|\psi\rangle \rho \langle \psi|} \quad (1.10)$$

Remarque 9. La fidélité "mesure"¹⁰ le rapprochement entre deux états.

Cette définition en tant que telle ne nous intéresse pas dans le cadre de ce rapport, cependant elle est nécessaire pour définir la notion de *la fidélité de porte*.

Dans un circuit, quand on veut appliquer une porte U , on applique en pratique une porte bruitée \mathcal{E} . La porte bruitée associe un état mixte à un état mixte comme par exemple pour la porte :

$$\rho \xrightarrow{\mathcal{E}} pX\rho X + (1-p)Z\rho Z \quad (1.11)$$

où $p \gg (1-p)$. Dans cet exemple, on applique la porte X à l'état avec une probabilité p , mais avec une probabilité $1-p$ on applique Z sur l'état. Nous avons déjà vu un autre exemple d'opérateur bruité dans la remarque 6.

Notre objectif est donc de rendre \mathcal{E} le plus proche possible de U . Pour calculer ce rapprochement on utilise la définition suivante :

9. cf. appendice 3.4

10. la fidélité ne définit pas une distance mathématiquement parlant !

Définition 5. On définit la fidélité de porte entre une porte U idéal et sa version bruité \mathcal{E} qui est appliquée en pratique comme :

$$F(U, \mathcal{E}) = \min_{|\psi\rangle} F(U|\psi\rangle, \mathcal{E}(|\psi\rangle\langle\psi|)) \quad (1.12)$$

La notion de fidélité est possible ici, car $U|\psi\rangle$ est un état pur, et $\mathcal{E}(|\psi\rangle\langle\psi|)$ est bien un état mixte

Illustrons directement cette définition par un exemple :

Exemple. Supposons que l'on souhaite implémenter une porte X , dans la pratique on applique une version bruité de X comme par exemple celle de l'équation 1.11. Où on pose $p = 0.9$. Dans ce cas la fidélité de la porte est donnée par :

$$\begin{aligned} F(X, \mathcal{E}) &= \min_{|\psi\rangle} F(X|\psi\rangle, \mathcal{E}(|\psi\rangle\langle\psi|)) \\ &= \min_{|\psi\rangle} \sqrt{\langle\psi| X (\mathcal{E}(|\psi\rangle\langle\psi|)) X |\psi\rangle} \\ &= \min_{|\psi\rangle} \sqrt{\langle\psi| X (pX|\psi\rangle\langle\psi| + (1-p)Z|\psi\rangle\langle\psi|Z) X |\psi\rangle} \\ &= \min_{|\psi\rangle} \sqrt{p\langle\psi| XX|\psi\rangle\langle\psi|X + (1-p)Z|\psi\rangle\langle\psi|ZX|\psi\rangle} \\ &= \min_{|\psi\rangle} \sqrt{p + (1-p)\langle\psi| ZX|\psi\rangle^2} \\ &= \sqrt{p} \\ &\simeq 0.94868 \end{aligned}$$

Remarque 10. Dans de nombreux cas, la définition ci-dessus n'est pas très convenable pour calculer la fidélité d'une porte. Pour une formule plus pratique on conseillera au lecteur l'article [38].

1.3.2 D'autres données importantes

Dans cette section nous allons traiter de façon bref, d'autre données d'un ordinateur quantique qui peuvent être importantes. Pour plus de détails sur ces quantités le lecteur pourra lire l'article [38].

La cohérence

Un qubit peut retenir une information, uniquement pendant un temps limité. Ce temps est appelé *le temps de cohérence*. Il y a deux temps de cohérence données par :

- Un qubit dans l'état $|1\rangle$ « descend »¹¹ naturellement dans l'état $|0\rangle$ ce temps de cohérence est noté $T1$.
- Un qubit peut être affecté par l'environnement, comme nous l'avons dit dans les sections précédentes. Ainsi on note $T2$ le temps de cohérence à partir duquel l'environnement va changer l'état d'un qubit donné.

Ces durées dépendent de la technologie utilisée, mais ils sont en général de l'ordre de la micro seconde.

La connectivité

Dans un ordinateur quantique pour pouvoir appliquer une porte CNOT ou n'importe quelle porte sur 2 qubits, il faut que ces 2 qubits soit connectés physiquement. Idéalement, on voudrait que tous les qubits soient connectés afin de pouvoir appliquer des portes CNOT entre n'importe quel qubit. Cependant, à l'heure actuelles, il est très compliqué physiquement de connecter chaque qubit.

C'est pourquoi on parle de *connectivité*. La connectivité correspond au nombres de qubits connectés dans un ordinateurs quantique.

11. sans rentrer dans les détails, le verbe descendre est utilisé, car physiquement il s'agit de niveau d'énergie.

La plupart des ordinateurs quantiques actuels utilisent un réseau en filet¹² où uniquement des qubits voisins sont connectés. Pour appliquer deux qubits non connectés, on va devoir utiliser la porte SWAP¹³.

1.3.3 Une métrique : le volume quantique

Dans cette section, nous allons brièvement présenter une métrique, prenant plusieurs paramètres d'un ordinateur quantique en compte, pour donner une mesure de sa puissance. Précisons cependant,

Pour cette section, on a utilisé l'article d'IBM [19]. Cependant cet article ne fournit qu'une rapide introduction théorique, pour plus de précision sur la pratique du calcul quantique le lecteur se référera à [5].

Afin, de pouvoir calculer le volume quantique, nous devons d'abord définir une notion :

Définition 6. On définit le taux d'erreur effectif qu'on note ϵ_{eff} , comme le taux d'erreur moyen pour une porte sur 2 qubit choisi tout deux aléatoirement.

Exemple. Supposons que notre ordinateur quantique puisse exécuter n'importe quelle porte sur deux qubits¹⁴, que chacun des qubits est connectés aux autres¹⁵ (ie. la connectivité est au maximum), que chaque portes à le même taux d'erreur qu'on note ϵ . Dans ce cas, $\epsilon = \epsilon_{\text{eff}}$.

Remarque 11. La notion de taux d'erreur effectif, est pertinente dans la création d'une métrique, car dans sa définition va transparaître la connectivité, la fidélité des portes, etc...

Soit n le nombre de qubits minimales nécessaire pour exécuter un circuit. Soit d la profondeur du circuit $d \simeq \frac{1}{n\epsilon_{\text{eff}}}$. On définit le volume quantique comme :

Définition 7.

$$V_Q = \max_{n' \leq n} (\min(n', \frac{1}{n'\epsilon_{\text{eff}}}))^2 \quad (1.13)$$

Dans la section suivante nous verrons des exemples de volumes quantiques.

1.4 L'ère NISQ

Depuis la découverte des algorithmes de Shor (en 1994 cf. [42]) et de Grover (en 1996 cf. [12]), la recherche sur la conception d'un ordinateur quantique n'a fait qu'accélérer. Comme on peut le voir sur la figure 1.14, le nombre de qubits que possède les ordinateurs quantiques n'a pas cessé de croître ces dernières années, et ne semble pas près d'arrêter. Au point de dépasser le seuil de suprématie quantique¹⁶.

Cependant, nous avons vu dans la section précédente que le nombre de qubits, ne suffisait pas pour caractériser un ordinateur quantique. C'est pourquoi, nous avons utilisé le site <https://quantumcomputingreport.com/scorecards/qubit-quality/> qui synthétisent un grand nombre de mesure, notamment celles dont nous avons parler dans la section précédente.

En effet, comme l'illustre la figure 1.15, la fidélité des portes à 1 qubit est plutôt élevé généralement, celle des portes à 2 qubits varient énormément d'un ordinateur quantique à l'autre.

De même sur la figure 1.16, on voit que la connectivité varie aussi énormément. Notamment le IonQ où tout les qubits sont connectés entre eux¹⁷.

On observe le même phénomène avec les temps de cohérence T1 et T2 sur la figure 1.17

12. mesh en anglais.

13. cf. appendice 3.5.

14. Si ce n'est pas le cas, il faut utiliser des algorithmes pour approximer les portes.

15. Si ce n'est pas le cas, il faut utiliser des portes SWAP.

16. Le seuil de suprématie quantique, est un seuil de qubits théorique dans un ordinateur quantique. Au delà de ce seuil un ordinateur quantique, sera capable de faire des tâches qu'aucun ordinateur classique actuelle n'est capable d'exécuter

17. Cette spécificité est due à la technologie utilisée : « Ion Trap ».

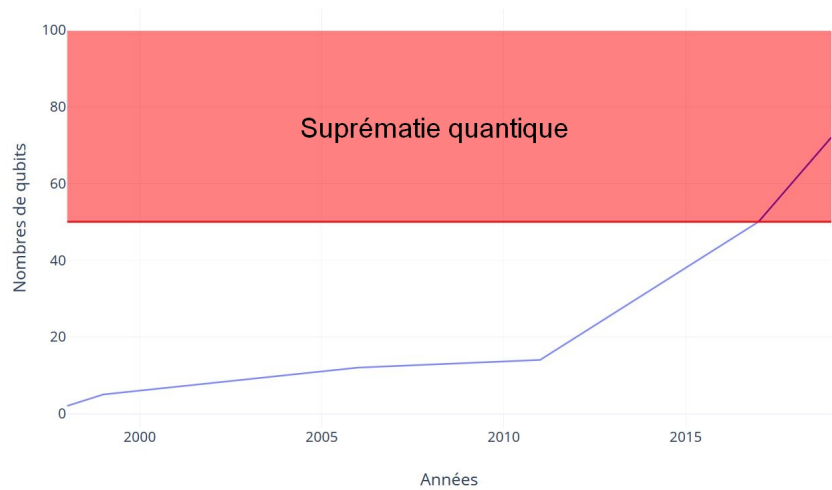


FIGURE 1.14 – L'évolution du nombres de qubits dans les ordinateurs quantiques

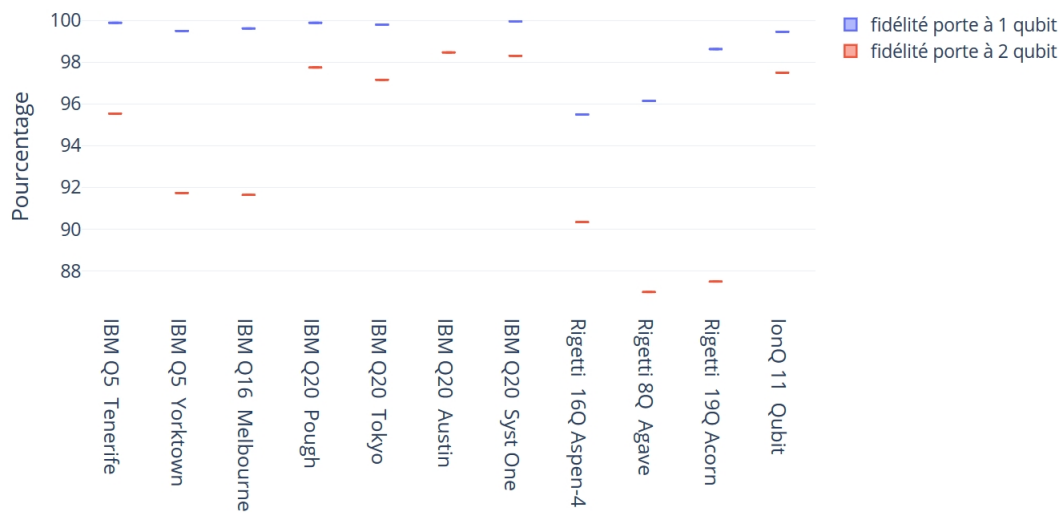


FIGURE 1.15 – Fidélité des portes à 1 et 2 qubits en fonction de l'ordinateur quantique

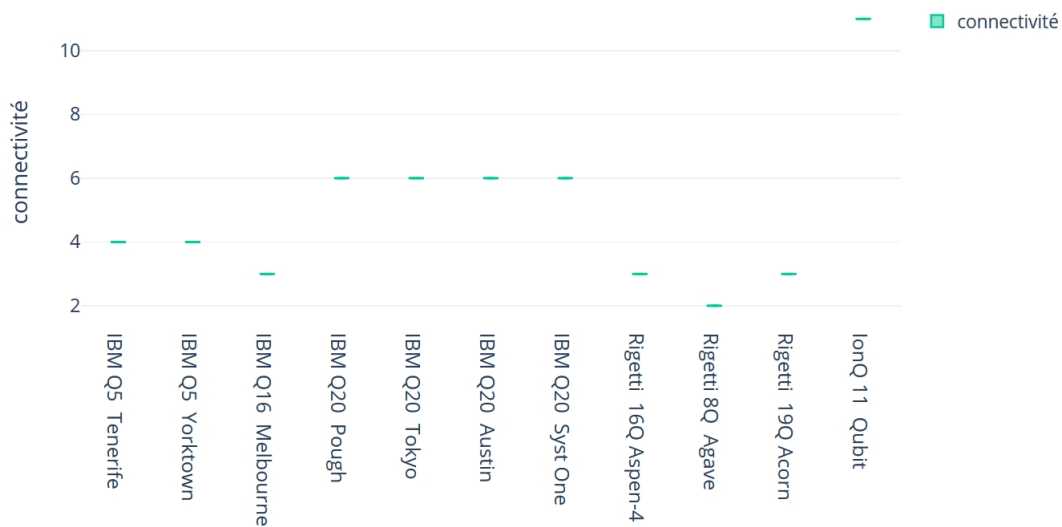


FIGURE 1.16 – Connectivité maximale en fonction de l'ordinateur quantique

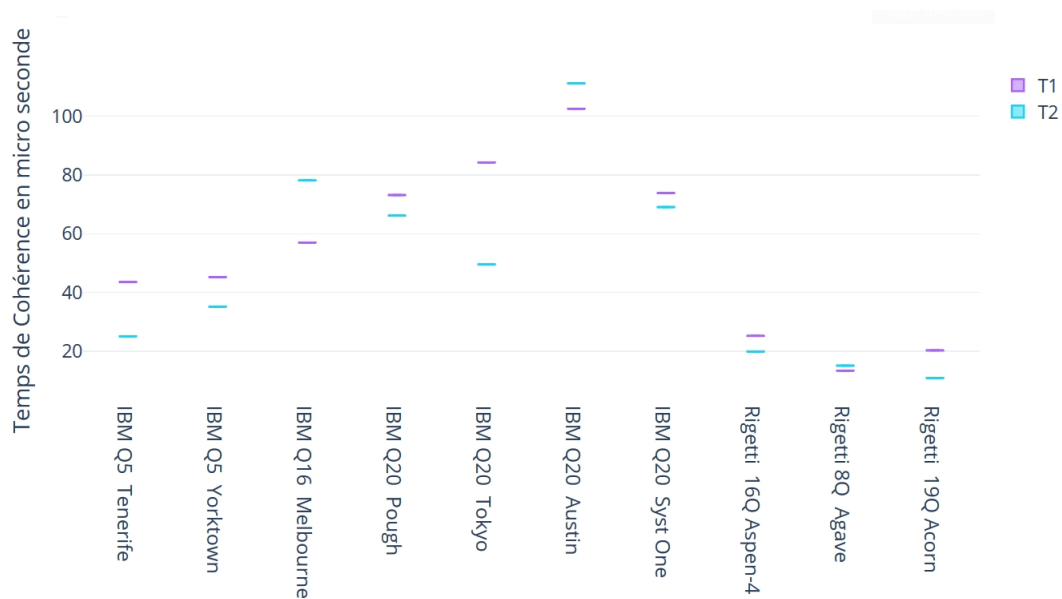


FIGURE 1.17 – Temps de cohérence en fonction de l'ordinateur quantique

De fait on aperçoit un soucis, la fidélité des portes même si elles augmente et est relativement grande, elle reste loin d'être négligeable. Surtout sur des circuits comptant un grand nombre de portes. De fait idéalement, si on ne veut pas négliger ce bruit, il faut utiliser les outils développer dans les deux premières sections 1.2 et 1.1.

Cependant cela engendre un nouveau problème, pour l'expliquer faisons une petite expérience, supposons que nous ayons accès au meilleur ordinateur quantique actuelle celui de google ayant 72 qubits. On souhaite implémenter un algorithme le plus protégé possible du bruit. Quelle est le nombre maximum de qubits que l'on peut utiliser ?

Supposons que l'on utilise un code à 5 qubits $[[5, 1, 3]]$ et qu'il y ait une unique zone bruité, dans ce cas chaque qubits nécessitera 5 autres qubits auxiliaire pour le protéger du bruit. De fait, dans cette zone on peut utiliser un maximum de 14 qubits. Un algorithme sur 14 qubits¹⁸ représente peu d'intérêt si ce n'est aucun, en effet même un ordinateur classique peu performant, serait capable de simuler un algorithme à 14 qubits.

Remarquons, ici que nous avons uniquement imaginé un cas, où l'on veut uniquement réduire le taux d'erreur. Mais imaginons que l'on veuille rendre le circuit fault-tolerant, et réduire le taux d'erreur à un niveau arbitraire, on sait que c'est possible grâce au théorème du seuil 1, mais à quelle coût... Finalement nos qubits seront protégés, mais si c'est pour en utiliser 2 ou 3 !

Récapitulons

Nous avons vu que les ordinateurs quantique actuelles ont déjà un certains nombre de qubits, mais pas suffisamment pour pouvoir appliquer des codes correcteurs et encore moins pour faire des algorithmes fault-tolerant.

Ainsi, à court terme si on veut trouver des applications aux ordinateurs quantiques, il faudra se satisfaire du contrôle imparfait que nous avons sur les qubits.

John Preskill a nommé cette période dans l'article [30] le *NISQ* pour *Noise Intermediate Scale Quantum*. Noise pour le bruit que nous ne pourrons pas restreindre. Intermediate Scale, car nous aurons accès à des ordinateurs ayant peu de qubits entre 50 et quelque centaines.

Le but de se projet est de trouver une application pour les ordinateurs quantique dans le NISQ. On cherche donc à utiliser le moins de qubits possible, qui nous sont limités. Une des pistes pour réduire le

18. Pour donner un ordre de grandeur $2^{14} = 16384$

nombre de qubits d'un circuit est l'hybridation qui est étudié dans le chapitre suivant.

Chapitre 2

Architecture hybride quantique-classique

Comme nous l'avons vu dans la dernière section du chapitre précédent, dans le NISQ, nous serons constamment limité par le nombre de qubits de nos ordinateurs quantique. Un exemple de problème récurrent auquel on pourrait faire face, est le suivant :

Supposons que l'on ait un ordinateur quantique de 100 qubits, comment peut-on faire si l'on souhaite implémenter un algorithme de 101 qubits dessus ?

Une des solution, que nous allons introduire dans ce chapitre, à ce problème est parfaitement illustrer par la maxime :

«Diviser pour mieux régner»

Nous allons entre autre diviser l'algorithme en plusieurs sous algorithme quantique qui seront exécutés par un ordinateur quantique et par un simulateur sur un ordinateur classique.

Dans ce chapitre, nous allons d'abord énoncer des résultats théorique, qui nous garantissent la possibilité de procéder ainsi. Ce chapitre est en grande partie inspiré de l'article [33].

2.1 Simulation des circuits d-creux

Dans cette section, nous allons développer une première approche à l'hybridation classique-quantique. Celle-ci se base sur les circuits d-creux, des circuits avec très peu de portes.

Dans toute cette section on travaillera avec des circuits avec n qubits, sur lesquelles on utilise uniquement des portes qui agissent sur au plus 2 qubits. Pour simplifier, on parlera à partir de maintenant d'une n -porte, pour parler d'une porte agissant sur n qubits. Commençons par donner la définition d'un circuit d-creux.

Définition 8. *Un circuit est dit d-creux si sur chaque qubits on applique au plus d 2-portes.*

Remarque 12. *On veut que d soit petit par rapport à n , soit une constante soit $d \sim \log(n)$*

Pour donner des exemples de circuits qui sont d-creux, nous allons introduire une autre caractéristique des circuits : ¹

Définition 9. *Si on considère qu'un circuit² exécute chacune des portes en une unité de temps, la profondeur d'un circuit est donnée par le maximum d'unité de temps nécessaire pour aller de l'entrée du circuit à sa sortie. De plus un circuit est dit de petite profondeur si sa profondeur est logarithmique par rapport à sa taille.*

1. Pour plus de précision cf. [36].

2. quantique mais aussi classique.

Ainsi, il est évident que pour $d = \log(n)$ tout les circuit de petite profondeur sont des circuits d-creux.

Maintenant que nous avons la définition d'un circuit d-creux, on va définir une procédure de calcul sur lequel on va s'appuyer dans la suite. On appelle cette procédure *le processus quantique d-creux* abrégier d-SQC³, qui est défini par les étapes suivante :

1. L'initialisation des n qubits dans l'état $|0\rangle$.
2. L'action d'un circuit d-creux.
3. La mesure de chaque qubits dans la base canonique.
4. Enfin un traitement classique du résultat qui retourne un unique bit noté b_{out}

Définition 10. On dit qu'un ordinateur classique ou quantique simule un d-SQC, si il donne la probabilité d'obtenir $b_{out} = 1$.

Ce qui nous amène à énoncer le théorème fondamentale de cette section :

Théorème 2. Soit U un d-SQC sur $n + k$ qubits, supposons que $n \geq kd + 1$. Alors, U peut être simulé par un $d + 3$ -SQC sur n qubits répété $2^{O(kd)}$ fois, et par un processus classique d'ordre $2^{O(kd)} \text{poly}(n)$.

La démonstration du théorème est légèrement technique, c'est pourquoi nous démontrerons uniquement le résultat intermédiaire suivant :

Lemme 1. Soit U un d-SQC sur $n + k$ qubits. Soit AB une partition de l'ensemble des qubits, où $\text{card}(A) = k$ et $\text{card}(B) = n$. Alors

$$U = \sum_{\alpha=1}^{\chi} c_{\alpha} V_{\alpha} \oplus W_{\alpha} \quad (2.1)$$

où $\chi = 2^{4kd}$, V_{α} et W_{α} sont deux circuits d-creux. Et $\sum_{\alpha} |c_{\alpha}|^2 = 1$.

Démonstration. Comme U est un d-SQC, on sait qu'il contient au plus kd 2-portes qui couple des qubits de A à des qubits de B . On note G_1, G_2, \dots, G_m pour $m \leq kd$ ces portes.

Or soit $U(i, j)$ une 2-porte agissant sur le i -ième qubit de A et le j -ième qubit de B . Comme $U(i, j) \in SU(4)$, d'après le théorème de décomposition de Schmidt [cf appendice 3.6] on peut l'écrire comme une somme de produit tensoriel d'opérateur de Pauli. C'est à dire, pour $P_{\alpha} \in \{I, X, Z, Y\}$ on a l'égalité suivante

$$U(i, j) = \sum_{\alpha=1}^{16} c_{\alpha} P_{\alpha}(i) \otimes P_{\alpha}(j) \quad (2.2)$$

Et où $\sum_{\alpha} |c_{\alpha}|^2 = 1$. Ainsi en appliquant l'égalité de l'équation 2.2 sur les G_i , on obtient l'équation 2.1. \square

La suite de la démonstration est purement technique pour montrer que le d-SQC donne bien la probabilité d'obtenir $b_{out} = 1$.

2.2 Simulation des algorithmes QC

En avril 2019, une équipe de chercheurs a donné dans l'article [39] une technique plus général est plus souple que celle de la section précédente. Celle-ci se base sur *les réseaux de tenseurs*, une notion que nous expliquerons dans la section 2.2.1.

Rapidement, les réseaux de tenseurs donne une façon formelle et graphique de voir un circuit quantique en terme de tenseurs, et permettent de donner une décomposition d'un circuit comme plusieurs sous circuits sur moins de qubit.

3. d-SQC pour d-Sparse Quantum Computation

2.2.1 Réseaux de tenseurs

Avant d'expliquer la méthode de partitionnement des circuits quantique, nous allons introduire la notion de réseaux de tenseurs, qui est une notion dont les applications ne se limite pas aux calcul quantique. Pour cette section, nous avons utilisé l'article [16] pour les réseaux tenseurs.

Brève introduction à algèbre tensorielle

Tout d'abord faisons une brève introduction à l'algèbre tensorielle (cf. [11]).

Définition 11. Soit E un \mathbb{K} -espace vectoriel de dimension n , et soit E^* son dual. On appelle tenseur p contravariant et q fois covariant toute forme multilinéaire sur $(E^*)^p \times E^q$.

Remarque 13. On appelle ordre tenseur $p + q$ et p et q étant ses variances.

Théorème 3. L'ensemble des tenseurs pour p et q fixé forment un espace vectoriel.

Donnons deux exemples basiques de tenseurs :

Exemple. — $p = 1$ et $q = 0$: Un tel tenseur est une forme linéaire de E^* donc correspond à un simple vecteur de E .
 — $p = 0$ et $q = 1$: Un tel tenseur est une forme linéaire de E donc un élément de E^* .
 — Ordre 2 : Un tel tenseur est une matrice.

On peut faire le produit tensoriel de deux tenseurs, qui est défini comme :

Définition 12. Soit deux tenseurs :

$$\begin{aligned}\mathcal{T} &: (E^*)^p \times E^q \rightarrow \mathbb{K} \\ \mathcal{T}' &: (E^*)^{p'} \times E^{q'} \rightarrow \mathbb{K}\end{aligned}$$

Le produit tensoriel $\mathcal{T}, \mathcal{T}'$, noté $\mathcal{T} \otimes \mathcal{T}'$ est un tenseur tel que :

$$\mathcal{T} \otimes \mathcal{T}'(u^{*1}, \dots, u^{*p}, v_1, \dots, v_q, u'^{*1}, \dots, u'^{*p'}, v'_1, \dots, v'_{q'}) = \mathcal{T}(u^{*1}, \dots, u^{*p}, v_1, \dots, v_q) \cdot \mathcal{T}'(u'^{*1}, \dots, u'^{*p'}, v'_1, \dots, v'_{q'})$$

Introduisons la notation indicielle de l'algèbre tensorielle. Soit E un espace vectoriel de dimension n et soit (e_i) une base de cette espace. On sait que pour tout $v \in E$. Il existe $v^i \in E^*$ tel que :

$$v = \sum_{i=1}^n v^i e_i \quad (2.3)$$

On va utiliser la convention des indices muets, en omettant le symbole somme, ainsi on note l'équation 2.3 comme :

$$v = v^i e_i$$

De façon duale on considère une base (e^i) de E^* . Alors on peut décomposer $u^* \in E^*$, comme :

$$u^* = v_i e^i$$

où les v_i sont des éléments de E .

Remarque 14. On remarque, que pour les éléments du duales E^* (covariant) on mets les indices en haut. Et pour les éléments de E (contravariant) on mets les indices en bas.

Ainsi considérons un tenseur \mathcal{T} d'ordre 2, avec $p = 1$ et $q = 1$. On suppose que E et E^* possède respectivement une base (e_i) et (e^i) .

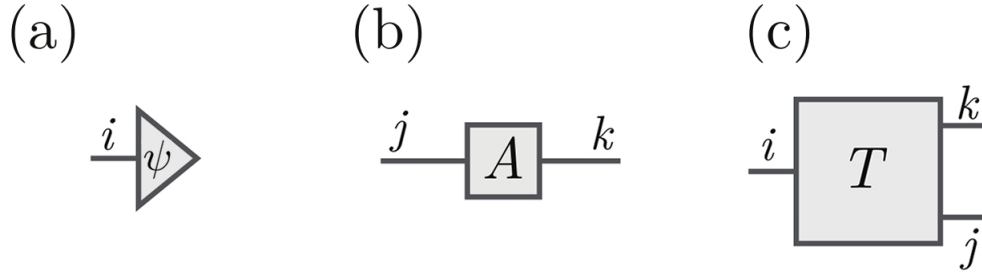


FIGURE 2.1 – 3 exemples de diagramme de tenseurs.

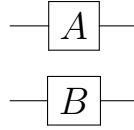


FIGURE 2.2 – Diagramme de deux tenseurs déconnectés

Soit $u \in E$ et $v \in E^*$, tel que $u = u^i e_i$ et $v = v_i e^i$, alors en décomposant $\mathcal{T}(u, v)$ dans les bases (e_i) et (e^i) , on a $\mathcal{T}(u, v) = u^i v_i \mathcal{T}(e_i, e^i)$. De plus pour faciliter les notations, on note $\mathcal{T}_i^j = \mathcal{T}(e_i, e^i)$. Ainsi, on a :

$$\mathcal{T}(u, v) = u^i v_i \mathcal{T}_i^j \quad (2.4)$$

On peut aisément généraliser cette décomposition à tout types de tenseurs.

Enfin, définissons la contraction de deux tenseurs à travers un exemple qui nous sera utile dans la suite.

Définition 13. *Le produit de deux matrices vu comme deux tenseurs $M_j^i N_k^j = P_k^i$ peut être vu comme une contraction. « On enlève les indices redondants ».*

Diagramme de tenseurs

Un tenseur peut être symboliser par un diagramme. Un tenseur p fois contravariant et q fois covariant peut être symboliser par une forme, ayant un p câble en entrée et q câble en sortie. Où l'on considère une figure se "lisant" de gauche à droite.

Ainsi sur la figure 2.1 on a 3 exemples de diagramme de tenseurs :

- (a) Représente le tenseur ψ^i , un tenseur 1 fois contravariant c'est à dire un vecteur.
- (b) Représente le tenseur A_k^j c'est à dire une matrice (ou un opérateur).
- (c) Représente le tenseur T_{jk}^i .

On peut utiliser plusieurs tenseurs déconnectés sur un seul diagramme. Dans ce cas, le diagramme correspondra à un produit tensoriel de tenseur. Comme sur la figure 2.2 qui correspond à $A \otimes B$.

A *contrario*, on peut connecté des tenseurs avec leurs câbles, lorsque c'est le cas, cela veut dire que l'on contracte les couples de tenseurs par l'indice du câble connectés. Par exemples sur la figure 2.3, les tenseurs A_k^j (une matrice) et ψ^i (un vecteur) sont connectés, de ce fait on les contracte selon l'indice i , on obtient :

$$A_k^j \psi^i = \phi^j \quad (2.5)$$

Algébriquement on a simplement appliqué une application linéaire A à un vecteur ψ ce qui nous donne un nouveau vecteur ϕ .

Ainsi on définit *un réseau de tenseurs* comme deux tenseurs ou plus dans un diagramme.

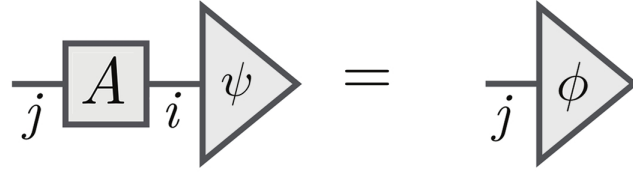


FIGURE 2.3 – Tenseurs connectés - contraction d'une matrice et d'un vecteur

On peut aussi, voir les réseaux de tenseurs comme des graphes, dont les sommets sont des tenseurs, et les arêtes les câbles qui les connectent.

Remarque 15. On remarque que la représentation des circuits quantiques en diagramme est un cas particulier de réseaux de tenseurs. Où les portes sont symbolisées par des quadrilatères (\square), la mesure par des triangles à droite (\triangleright), et les ket par des triangles à gauche (\triangleleft)

2.2.2 Marche à suivre pour la simulation

Avec ces outils en main, nous pouvons enfin rentrer dans le vif du sujet.

De même que dans la section 2.1, où nous avons défini le d-SQC, nous allons définir une procédure de calcul sur lequel notre raisonnement pourra s'appuyer. On la définit par les étapes suivante :

1. L'initialisation des n qubits dans l'état $|0\rangle$.
2. L'action d'un circuit C avec m portes.
3. La mesure de chaque qubits dans la base canonique.
4. Enfin un traitement classique du résultat qu'on assimile à une fonction $f : \{0, 1\}^n \rightarrow [-1, 1]$.

On appelle une telle procédure est appelé *un algorithme quantique-classique* abrégé algorithme QC. On peut aussi noter une telle procédure comme le couple (C, f) .

on note la moyenne du résultat en sortie sur un grand nombre de mesures $\mathbb{E}_y f(y)$.

Définition 14. Soit S une simulation d'un algorithme QC (C, f) , on dit que S est une bonne simulation, si la moyenne des résultats en sortie sur un grand nombre de mesures qu'on note M , vérifie pour un petit $\epsilon > 0$ donné :

$$||M - \mathbb{E}_y f(y)|| \leq \epsilon$$

Maintenant, nous allons donner une marche à suivre en plusieurs pour créer une bonne simulation d'un algorithme QC quelconque.

Phase 1 - Partitionnement du réseau de tenseurs associé

On a vu dans la section précédente qu'un circuit quantique pouvait être vu comme un réseau de tenseur. De même un algorithme QC peut être évidemment vu comme un réseau de tenseur, qu'on note (G, A) , où G est le graphe associé au réseau, et A un l'ensemble des tenseurs. On note $T(G, A)$ la valeur du réseau de tenseur qui est égale à $\mathbb{E}_y f(y)$.

Une idée naturel pour réduire le nombre de qubits utilisés, est de découper le réseau de tenseurs en plusieurs réseaux différents. Cependant, il faut que ces différents réseaux interagissent le moins possible entre eux. C'est pourquoi on donne la définition suivante :

Définition 15. Un algorithme QC est dit (K, d) - partitionné si son réseau de tenseur, s'il existe une partition S_1, \dots, S_r de tout les tenseurs, sauf des derniers tenseurs effectuant la mesure. Tel que K soit le nombre total de qubits communicant entre plusieurs partitions, et d le nombre de qubits minimum pour simuler chaque partition.

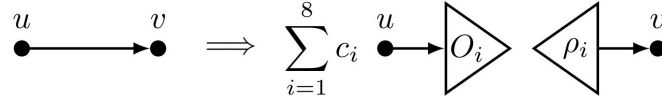


FIGURE 2.4 – On coupe l'arête en deux

Ainsi, dans un premier temps on considère le réseau de tenseur associé au circuit quantique. Puis on le partitionne en plusieurs réseaux différents tout en essayant de minimiser K et d .

Phase 2 - Découpage des arêtes communicantes entre les différentes partitions

Maintenant que nous avons donné une (K, d) -partition de notre circuit, on se retrouve face à un problème.

« Comment simuler l'interaction entre les différentes partitions, afin que chaque partition puissent être simulé indépendamment des autres sur d qubits ? »

Soit une arête $u \rightarrow v$ entre deux partitions des réseaux de tenseurs. Idéalement, on cherche on voudrait appliquer un opérateur de mesure⁴ d'un coté de l'arête et initialisé dans un certain état de l'autre coté pour pouvoir couper l'arête, afin qu'il n'y ai plus de communication entre les deux partitions. Cette méthode est possible, cependant elle rajoute un coût. En effet, cela nécessite d'exécuter plusieurs fois le circuit pour différentes bases de mesure et différentes initialisation (cf. figure 2.4).

Ce résultat nous est donné par le lemme suivant :

Lemme 2. *Soit (G, A) un réseau de tenseurs d'un algorithme QC. Alors pour toutes arêtes $e \in G$, il existe un opérateur mesure O_i et un état dans lequel on initialise le qubit ρ_i tel que :*

$$T(G, A) = \sum_{i=1}^8 c_i T(G', A_i) \quad (2.6)$$

où G' est égale à G sans l'arête e et où l'on a ajouté O_i (\triangleright) à gauche de l'arête et ρ_i (\triangleleft) à droite, pour $c_i \in \{\frac{1}{2}, -\frac{1}{2}\}$.

Nous n'allons pas rentrer dans les détails de la démonstration qui peut être lu dans l'article [39]. Cependant, nous allons donner une rapide idée de la construction des O_i et ρ_i . Comme nous l'avons déjà vu les matrices de Pauli normalisée forment une bases orthonormales des opérateurs. Ainsi on se sert de ça pour déterminer les O_i , les ρ_i et les c_i :

$$\begin{aligned} O_1 &= I, \quad \rho_1 = |0\rangle\langle 0|, \quad c_1 = +\frac{1}{2} \\ O_2 &= I, \quad \rho_2 = |1\rangle\langle 1|, \quad c_1 = +\frac{1}{2} \\ O_3 &= X, \quad \rho_3 = |+\rangle\langle +|, \quad c_1 = +\frac{1}{2} \\ O_4 &= X, \quad \rho_4 = |-\rangle\langle -|, \quad c_1 = -\frac{1}{2} \\ O_5 &= Y, \quad \rho_5 = |+i\rangle\langle +i|, \quad c_1 = +\frac{1}{2} \\ O_6 &= Y, \quad \rho_6 = |-i\rangle\langle -i|, \quad c_1 = -\frac{1}{2} \\ O_7 &= Z, \quad \rho_7 = |0\rangle\langle 0|, \quad c_1 = +\frac{1}{2} \\ O_8 &= Z, \quad \rho_8 = |1\rangle\langle 1|, \quad c_1 = -\frac{1}{2} \end{aligned}$$

où $|\pm\rangle = \frac{|0\rangle \pm |1\rangle}{\sqrt{2}}$ et $|\pm i\rangle = \frac{|0\rangle \pm i|1\rangle}{\sqrt{2}}$.

4. c'est à dire une mesure dans une certaine base, formellement en mécanique quantique on appelle ça un observable.

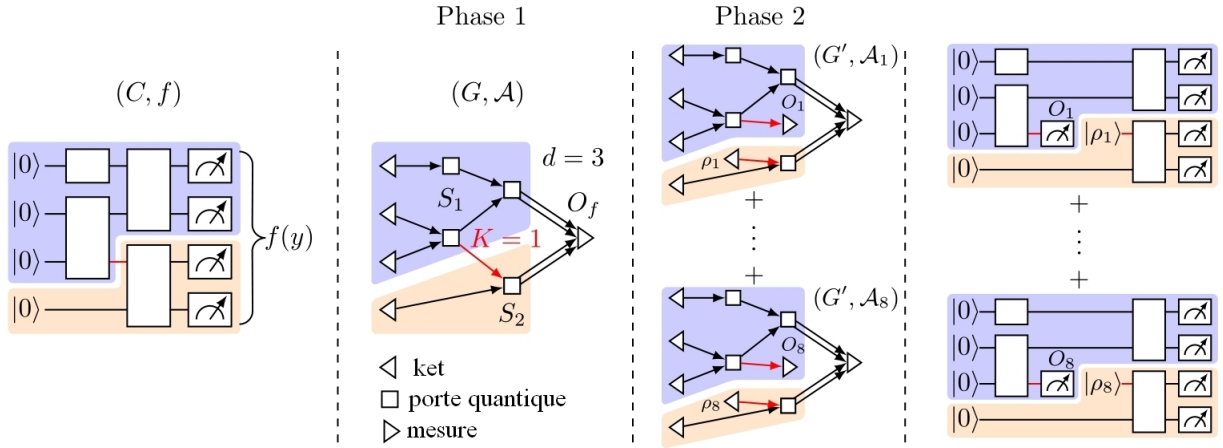


FIGURE 2.5 – Les 4 phases de la simulations

Remarque 16. Il nous faut pas être étonné de voir que les O_i (les opérateurs de mesure) soit des matrices de Pauli. En effet, chaque matrices de Pauli peut être vu comme un opérateur de mesure. Par exemple, appliquer l'opérateur de mesure X correspond à faire une mesure dans la base $(|+\rangle, |-\rangle)$. De façon plus générale appliquer l'opérateur de mesure O_i correspond à faire une mesure dans la base de ses vecteurs propres.

Maintenant que nous avons correctement découpé les liaisons entre nos réseaux de tenseurs, nous pouvons exécuter les 8 partitions sur d qubits.

Exemple

Illustrons la simulation à travers un exemple, le circuit est donné figure 2.5.

Phase 1 Après avoir donné le réseau de tenseur associé au circuit. On divise le circuit en deux partitions qu'on note S_1 et S_2 . Un seul qubit communique entre les deux partitions, donc $K = 1$. De plus, 2 qubits sont nécessaires pour simuler S_2 , et 3 qubits sont nécessaires pour simuler S_1 , on a donc $d = 3$. L'algorithme QC, est donc $(1, 3)$ – partitionné.

Phase 2 On va appliquer le lemme 2 sur les partitions, on a donc i itérations pour i allant de 1 à 8 du réseau de tenseurs en rajoutant l'opérateur de mesure O_i dans S_1 et on initialise le premier qubit de S_2 comme ρ_i .

La complexité de la simulation

La complexité de la simulation quant à elle dépend des paramètres (K, d) , du nombre de qubits n , du nombre de portes m du circuit quantique initial, et de ϵ la précision désiré. Elle est donnée par le théorème suivant, dont on donnera uniquement l'énoncé :

Théorème 4. Soit un algorithme QC (K, d) – partitionné, alors le temps d'exécution classique et quantique de la simulation est $\mathcal{O}(2^{4K \frac{(n+m)}{\epsilon^2}})$.

Ainsi, on remarque le temps d'exécution de la simulation est exponentielle en K . De ce fait, la simulation est vraiment pratique pour des circuits pouvant être partitionné sans trop de liaisons entre les différentes partitions.

Chapitre 3

Application résolution de système linéaire hybride

En 2009 les chercheurs Aram W. Harrow, Avinatan Hassidim, et Seth Lloyd ont publié un article [6] donnant un nouvel algorithme avec une amélioration exponentielle par rapport à son équivalent classique. Cette algorithme qu'on appellera *HHL* est un algorithme qui permet de donner une approximation de la solution de certain système linéaire.

La résolution de système linéaire, est un algorithme omniprésent dans toutes les mathématiques appliqués. Que ce soit à travers la méthode des éléments finis, ou encore le machine learning. C'est pourquoi l'algorithme HHL est d'une importance primordiale et jouera une place capitale dans les futures applications du calcul quantique.

Ce chapitre se décompose en deux grandes section, une première où on étudiera l'algorithme HHL, et une seconde dans la continuité du chapitre précédent où l'on développera une alternative hybride de l'algorithme pour le NISQ.

3.1 L'algorithme HHL□

Comme nous l'avons dit plus haut l'algorithme HHL, est un algorithme qui permet de donner une approximation de la solution de certain système linéaire. En effet, nous verrons après avoir présenté l'algorithme et donné un exemple de résolution dans la première section et seconde section, que l'algorithme impose un certain nombres de contraintes sur le système.

Cependant si ces contraintes sont vérifiées l'algorithme a une complexité logarithmique par rapport à la taille de la matrice. Comparé au meilleur algorithme classique qui est linéaire par rapport à la taille de la matrice.

Pour écrire cette section, nous nous sommes en grandes partis inspirés du travail séminale [6], mais aussi de l'article plus détaillé [7].

3.1.1 Un rapide aperçu de l'algorithme

On souhaite résoudre le système linéaire :

$$Ax = b \tag{3.1}$$

Supposons d'abord que b et x sont des vecteurs de norme 1, qui peuvent donc être représentés par des ket $|b\rangle$ et $|x\rangle$. Supposons que A est une matrice carré $N \times N$ hermitienne de rang N ¹.

1. de tel sorte que le système 3.1 possède une unique solution

Dans ce cas on peut écrire le système 3.1 comme :

$$A|x\rangle = |b\rangle \quad (3.2)$$

L'équation 3.2 admet une unique solution donnée par :

$$|x\rangle = \frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|}$$

La question est donc de savoir « comment peut-on réécrire cette solution avec des termes que l'on sait calculer à l'aide d'algorithme quantique ? ».

Un algorithme que nous avons déjà étudié dans [21] l'estimation de phase quantique (qu'on abrégera QPE pour Quantum Phase Estimation), permet d'estimer les valeurs propres d'un opérateur à partir de ces vecteurs propres. De ce fait ré-exprimons l'équation 3.1.1 en fonction des valeurs propres de A .

On se place dans la base des vecteurs propres de A qu'on note $(|u_j\rangle)_{j=1\dots N}$, de plus on note λ_j la valeur propre associé à $|u_j\rangle$.

Dans cette base le vecteur $|b\rangle$ est donné par :

$$\sum_{j=1}^N \alpha_j |u_j\rangle \text{ pour } \alpha_j \in \mathbb{C} \quad (3.3)$$

On en déduit une réécriture de l'équation 3.1.1 comme :

$$|x\rangle = \frac{1}{N} \sum_{j=1}^N \frac{1}{\lambda_j^{-1}} \alpha_j |u_j\rangle \quad (3.4)$$

Où N est un coefficient de normalisation. Ainsi on cherche un circuit qui à $|b\rangle$ et A associe $|x\rangle = \frac{1}{N} \sum_{j=1}^N \frac{1}{\lambda_j^{-1}} \alpha_j |u_j\rangle$.

Ainsi on procède en plusieurs étape :

Tout d'abord on pose un premier registre de qubits dans l'état $|0\rangle$. Dans un soucis de pédagogie on notera par l'indice b les qubits de se registre. On va appliquer des transformations afin d'initialiser se registre dans l'état $|b\rangle_b$.

Remarque 17. *Initialiser un registre de qubits dans un état arbitraire est une opération complexe et coûteuse, nous reviendrons dessus dans la section 3.1.3.*

L'estimation de phase quantique

Ensuite on pose un second registre de qubits qu'on notera cette fois par l'indice c . Celui-ci sera le registre en «contrôle» du second pour QPE. Plus se registre possède de qubits plus l'estimation de la valeur propre est «bonne». Au lieu d'initialiser les registres dans l'état $H^{\otimes n}$. On va initialiser se registre dans l'état :²

$$|\Psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin \frac{\pi(\tau + \frac{1}{2})}{T} |\tau\rangle \quad (3.5)$$

Dans QPE si on veut chercher les valeurs propres d'un opérateur U , on applique la porte conditionnelle au puissance de U . Cependant on fait face à un premier problème, dans QPE l'opérateur U est un opérateur

2. Sans rentrer dans les détails, l'approximation de l'estimation de phase quantique sera plus optimale avec cette initialisation. Le lecteur curieux pourra lire la réponse de DaftWullie sur <https://quantumcomputing.stackexchange.com/questions/2393/quantum-algorithm-for-linear-systems-of-equations-hhl09-step-2-what-is-p/2462#2462>.

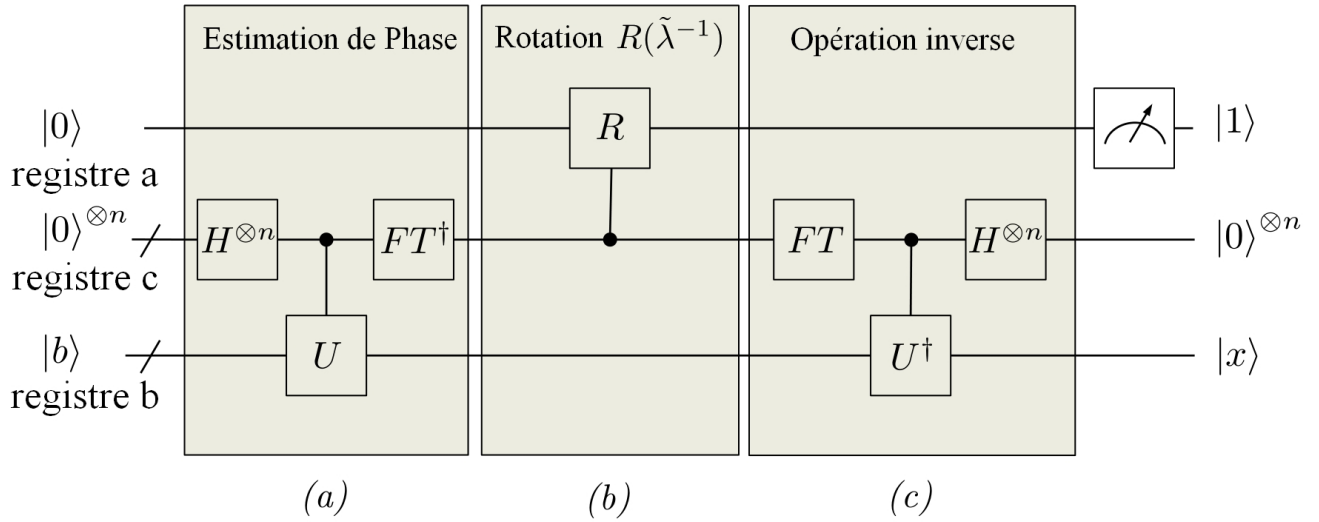


FIGURE 3.1 – Circuit HHL

unitaire. Or dans notre cas, on souhaiterait appliquer QPE à l'opérateur A qui n'est pas forcément unitaire.

De ce fait, on cherche une façon de transformer notre opérateur A en opérateur unitaire de préférence possédant les mêmes valeurs propres que A . Par chance cette transformation existe, et est donnée par :

$$A \mapsto e^{iAt}$$

En effet, l'exponentielle d'un opérateur hermitienne est un opérateur unitaire et il possède les même valeurs propres. C'est pourquoi on va, non pas appliquer QPE sur l'opérateur A , mais sur l'opérateur e^{iAt} .

Remarque 18. *Ce qui peut paraître simple théoriquement ici. Est l'une des choses les plus complexes de l'algorithme. En effet pour pouvoir transformer A en e^{iAt} , on utilise l'algorithme de la simulation hamiltonienne, un algorithme qui peut être très couteux en fonction de la matrice. Nous reviendrons sur se point dans la section 3.1.3.*

Une fois qu'on a l'opérateur e^{iAt} , on l'applique en contrôle au registre $|b\rangle_b$, puis on applique l'inverse de transformée de Fourier quantique (cf. figure 3.1).

Ainsi après la QPE notre système est approximativement dans une superposition des valeurs propres de A . Plus formellement la QPE correspond à :

$$|0\rangle_c |b\rangle_b \mapsto \sum \alpha_j |\tilde{\lambda}_j\rangle_c |u_j\rangle_b \quad (3.6)$$

où $|\tilde{\lambda}_j\rangle_c$ est une représentation binaire de λ_j .

Ce circuit est représenté figure 3.1 (a).

Rotation contrôlée en fonction des valeurs propres

La seconde étape de l'algorithme est d'appliquer un opérateur sur nos registres afin de déterminer les valeurs propres de A^{-1} en fonction des valeurs propres de A que nous avons déterminé à l'étape précédente.

Pour cela on va utiliser le lemme suivant :

Lemme 3. Soit $\theta \in \mathbb{R}$ et soit $\tilde{\theta}$ une représentation binaire sur d -bit de θ . Alors, il existe un opérateur unitaire

$$U_\theta : |\tilde{\theta}\rangle |0\rangle \mapsto |\tilde{\theta}\rangle (\cos\tilde{\theta} |0\rangle + \sin\tilde{\theta} |1\rangle) \quad (3.7)$$

Démonstration. Posons

$$U_\theta = \sum_{\tilde{\theta} \in \{0,1\}^d} |\tilde{\theta}\rangle \langle \tilde{\theta}| \otimes \exp(-i\tilde{\theta}Y)$$

où Y est la porte de Pauli donné par la matrice $\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$. La matrice $\exp(-i\theta Y)$ est donné par $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$, ainsi quand on applique U_θ à $|\tilde{\theta}\rangle |0\rangle$, on obtient bien le résultat de l'équation 3.7. \square

En conséquence, on rajoute un 3ème registre d'un qubits auxiliaire dans l'état $|0\rangle$ qu'on notera par l'indice a. On utilise le lemme 3 en posant $\theta = 2 \arcsin(\frac{C}{\lambda})$ où $C = \mathcal{O}(1/\kappa)$ est une constante de normalisation (on rappelle que κ est le conditionnement de la matrice). De ce fait en appliquant les opérateurs U_θ sur le registre a et sur le registre c en contrôle. On a ³ :

$$\sum \alpha_j |\tilde{\lambda}_j\rangle_c |u_j\rangle_b |0_A\rangle \mapsto \sum \alpha_j \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} |0\rangle_a + \frac{C}{\tilde{\lambda}_j} |1\rangle_a \right) |\tilde{\lambda}_j\rangle_c |u_j\rangle_b$$

Ce qui correspond à la partie (b) du circuit 3.1.

Opération inverse

Pour finir, on souhaite se débarrasser de toutes les informations superflus, de ce fait on va procéder au opération inverse de la QPE effectué précédemment. On laisse donc le système dans l'état :

$$\sum \alpha_j \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} |0\rangle_a + \frac{C}{\tilde{\lambda}_j} |1\rangle_a \right) |\tilde{\lambda}_j\rangle_c |u_j\rangle_b \mapsto |0\rangle_c \otimes \sum \alpha_j \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} |0\rangle_a + \frac{C}{\tilde{\lambda}_j} |1\rangle_a \right) |u_j\rangle_b$$

On remarque que si on obtient 1 après la mesure du registre a on obtient :

$$\sqrt{\frac{1}{\sum_{j=1}^N C^2 |\alpha_j|^2}} \sum_{j=1}^N \alpha_j \frac{C}{\tilde{\lambda}_j} |u_j\rangle_b$$

On obtient $|x\rangle$ (cf. équation 3.4) à un facteur de normalisation près.

Ainsi on répète les étapes précédentes jusqu'à obtenir 1 après la mesure du registre a. Pour augmenter cette probabilité on peut utiliser l'algorithme d'amplification des amplitudes que nous ne détaillerons pas ici, mais qui est très bien décrit dans [10].

Ce qui correspond à la partie (c) du circuit 3.1.

Récapitulons

Récapitulons les différentes étapes de l'algorithme :

1. On initialise le registre b comme vecteur $|b\rangle$
2. On initialise le registre a comme vecteur $|\psi_0\rangle$.
3. On rappelle la formule trigonométrique $\cos(\arcsin(\theta)) = \sqrt{1 - \theta^2}$

3. On applique sur le registre b en contrôle du registre c les simulation hamiltonienne formellement on applique $U = \sum_{k=0}^{2^t-1} |k\rangle_c \langle k|_c \otimes e^{2i\pi A \frac{k}{2^t}}$ où t est le nombre de qubits du registre c. Plus vulgairement on applique sur l'ensemble des qubits du registre b en contrôle du k-ième qubit de c l'opérateur $e^{2i\pi A \frac{k}{2^t}}$.
4. On applique la QFT* sur le registre c.
5. On applique les rotations $R_y(\theta_j)$ où $\theta_j = 2 \arcsin \frac{C}{\lambda_j}$ sur le registre a en contrôle du registre c.
6. On inverse les étapes 2 à 4.
7. On réitère jusqu'à mesuré 1 sur le registre a.

3.1.2 Exemple de résolution linéaire

Pour cet exemple, nous avons utilisé l'excellent article regroupant de nombreuses implémentations d'algorithmes quantique [27].

On va appliquer l'algorithme HHL au système $Ax = b$ suivant :

$$\begin{pmatrix} 3/2 & 1/2 \\ 1/2 & 3/2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \quad (3.8)$$

Où l'on testera numériquement sur différents couples (α_0, α_1) . Cependant, pour la théorie on pose $(\alpha_0, \alpha_1) = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ afin que l'état soit facile à initialiser.

La diagonalisation de A est donnée par :

$$A = P^{-1}DP = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} -1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix} \quad (3.9)$$

Remarque 19. La matrice du système 3.8 n'a pas été choisi inopinément, en effet les valeurs propres de la matrices sont deux puissances de deux. Cela permet lors de l'estimation de phase quantique d'avoir une estimation exacte des valeurs propres.

De plus, on peut aussi remarqué que la matrice P est égale à l'opérateur $R_y^{\pi/2}$.

Débutons l'algorithme :

1. D'abord on a un premier registre de 1 qubits dans l'état $|0\rangle_b$ (suffisant pour stocker un vecteur à deux coefficients). On veut initialiser se registre dans le vecteur $b = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ pour cela on applique une porte H. On a :

$$|0\rangle_b \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle_b + |1\rangle_b)$$

2. On va utiliser 2 qubits pour le registre c. Pour simplifier ici on va négliger l'optimisation de l'initialisation à $|\Psi_0\rangle$ et on va uniquement initialiser le registre c en appliquant des portes H à chaque qubits.
3. On souhaite appliquer les simulations hamiltonienne plus précisément les opérateurs $e^{2i\pi A \frac{k}{2^t}}$, où $t = 2$ le nombres de qubits du registre c, et pour k variant entre 0 et $2^t - 1$. Ainsi on souhaite appliquer $e^{i\pi A}$ et $e^{i\pi A/2}$ sur le registre b en contrôle respectivement du premier et du second qubit du registre c. On va les calculer à la "main" ici, cependant dans la pratique il faudrait procéder à

Vecteur \mathbf{b}	Solution HHL	Solution Exacte	$\ \mathbf{x}_{\text{exact}} - \mathbf{x}_{\text{HHL}}\ $
$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$	$\begin{pmatrix} 0.7155589 \\ 0.69855241 \end{pmatrix}$	$\begin{pmatrix} 0.70710678 \\ 0.70710678 \end{pmatrix}$	0.0120256
$\begin{pmatrix} \frac{1}{\sqrt{3}} \\ \frac{\sqrt{2}}{\sqrt{3}} \end{pmatrix}$	$\begin{pmatrix} 0.48703705 \\ 0.87338131 \end{pmatrix}$	$\begin{pmatrix} 0.43932091 \\ 0.8983302 \end{pmatrix}$	0.05384494
$\begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{\sqrt{4}} \end{pmatrix}$	$\begin{pmatrix} 0.36910931 \\ 0.92938599 \end{pmatrix}$	$\begin{pmatrix} 0.28925258 \\ 0.95725281 \end{pmatrix}$	0.084579293

FIGURE 3.2 – Solution HHL - Les vecteurs ont tous été normé

une simulation hamiltonienne.

$$\begin{aligned}
e^{i\pi A} &= P^{-1} \begin{pmatrix} e^{i\pi} & 0 \\ 0 & e^{2i\pi} \end{pmatrix} P \\
&= P^{-1} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} P \\
&= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\
e^{i\pi A} &= X
\end{aligned}$$

Donc on applique d'abord une porte CNOT sur le registre b avec le premier qubit du registre c en contrôle.

$$\begin{aligned}
e^{i\pi A \frac{1}{2}} &= \frac{1}{2} \begin{pmatrix} -1+i & -1-i \\ -1-i & -1+i \end{pmatrix} \text{ ce qui est approximativement égale à} \\
&\simeq (R_y^{\pi/2})^* (R_x^{\pi})^* R_z^{\pi/2} R_x^{\pi} R_y^{\pi/2}
\end{aligned}$$

Puis on applique cette approximation⁴ sur le registre b avec le second qubit du registre c en contrôle.

4. On applique la QFT* sur le registre c.

C'est à dire, sur 2 qubits, cela correspond à une porte d'Hadamard sur le second qubit, la porte S^* ⁵ sur le premier qubit et en contrôle sur le second qubit, et enfin une porte d'Hadamard sur le premier qubit.

5. On sait que $C = \mathcal{O}(1/\kappa)$, d'où on prend $C = 1/2$, ainsi on applique des rotations $\theta_j = 2 \arcsin(\frac{1}{2\lambda_j})$.

D'où on applique $R_y^{\theta_1}$ sur le qubit du registre a avec le premier qubit du registre c en contrôle, où $\theta_1 = \pi/3$. $R_y^{\theta_2}$ sur le qubit du registre a avec le second qubit du registre c en contrôle, où $\theta_2 = 0.505$.

6. On inverse les étapes 2 à 4.

Après implémentation dans qiskit on obtient les solutions représentées dans la figure 3.2.

Remarque 20. Avec un meilleur choix de la constante C , on peut obtenir des solutions bien plus précise. Néanmoins cela demande une étude précise comme dans [43].

4. On utilise cette approximation, car elle est simple à implémenter sur Qiskit.

5. On rappelle que la porte S est définie comme la matrice $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix}$

3.1.3 Contraintes de l'algorithme

Cette section est en grandes partie inspiré de l'introduction de l'article de Scott Aaronson [1].

L'algorithme HHL n'est pas un algorithme de résolution de système linéaire au sens *stricto sensu* du terme. En effet, il ne permet pas de résoudre un système linéaire quelconque, car l'algorithme possède différentes contraintes que nous allons lister dans cette section.

Supposons que l'on souhaite résoudre le système 3.1.

La matrice A doit être hermitienne

Nous avons vu dans la section précédente que pour que l'opérateur e^{iAt} soit unitaire, il fallait que la matrice A hermitienne.

Cette restriction peut être contournée en considérant le système suivant :

$$\begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix} \begin{pmatrix} 0 \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

où cette fois la matrice sera bien hermitienne. Cependant le registre b nécessitera 2 fois de qubits pour stocker le vecteur $\begin{pmatrix} b \\ 0 \end{pmatrix}$.

Le vecteur b doit être unitaire

Afin de pouvoir encoder le registre b dans l'état $|b\rangle$, il faut que le vecteur b soit un vecteur unitaire.

De même que dans la section ci-dessus cette restriction peut être contournée en définissant le vecteur :

$$|c\rangle = \frac{b}{\|b\|}$$

Et de fait la matrice $B = \frac{1}{\|b\|}A$, pour résoudre le système :

$$B|x\rangle = |c\rangle$$

Initialisation d'un registre de qubits dans un état arbitraire

Dans l'algorithme il est nécessaire d'initialiser le registre b dans l'état $|b\rangle$. Comme nous l'avons mentionné dans la remarque 17, l'initialisation d'un registre de qubits dans un état arbitraire est une opération bien plus complexes qu'elle ne semble l'être.

Une des méthodes pour permettre cette initialisation est l'utilisation d'une RAM quantique ou QRAM. Nous ne rentrerons pas dans les détails de cette méthode, cependant de nombreuses sources d'erreurs sont à prendre en compte. Et aujourd'hui encore il n'est pas évident que l'implémentation d'une telle méthode puisse se faire avec une complexité et un taux d'erreur raisonnable.

Le lecteur souhaitant en apprendre plus sur la QRAM pourra lire [29], sur les erreurs qu'elle entraîne [37]. Et plus précisément sur les techniques pour initialiser un registre de qubits dans un état arbitraire [20] ou encore [14].

On en conclut que si le vecteur b n'est pas simple à initialiser, par exemple si son implémentation nécessite une complexité polynomiale. L'algorithme perd tout son intérêt à peine après avoir été commencé.

La simulation Hamiltonienne

Donnons tout d'abord une définition précise de la simulation Hamiltonienne.

Définition 16. *On dit qu'un Hamiltonien (une matrice hermitienne) qui agit sur n qubits peut être efficacement simulé si pour tout $t > 0$ et pour tout $\epsilon > 0$, il existe un circuit quantique U_H tel que*

$$\|U_H - e^{iHt}\| < \epsilon \quad (3.10)$$

Tout comme l'initialisation d'état arbitraire, la simulation hamiltonienne est une opération complexe et qui peut en fonction de la matrice hermitienne être très coûteuse. Par exemple un algorithme naïf pour une matrice $N \times N$ a une complexité en temps de $\mathcal{O}(N^3)$. Une telle complexité ruinerait l'intérêt de l'algorithme HHL.

Un des cas que l'on sait traiter rapidement est le cas d'une matrice s -creuse⁶. En effet pour une matrice s -creuse, on peut simuler son Hamiltonien dans un temps linéaire par rapport à s .

Il existe de nombreuses pour résoudre le problème de la simulation hamiltonienne, décrite notamment dans les articles suivant [9], [13] et [4].

Le conditionnement de la matrice

Pour que l'algorithme soit efficace, la matrice ne doit pas être juste inversible, elle doit être bien conditionnée.

Plus précisément si on note $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$ le conditionnement donné par le ratio entre la plus grande et la plus petite valeur propre. Alors le temps nécessaire pour exécuter l'algorithme grandit linéairement par rapport à κ . Ainsi pour des valeurs de κ trop grande l'algorithme perd son intérêt.

L'accès au résultat

L'inconvénient majeur de l'algorithme réside dans le fait que la donnée de la solution x n'est pas directement accessible.

Lorsque l'algorithme a terminé de s'exécuter le résultat n'est pas x , mais un état quantique $|x\rangle$ de $\log_2 n$ qubits qui encode les coefficients de x dans leurs amplitudes. C'est pourquoi si on mesure directement on obtiendra une information partielle de x , qui est inutile toute seule.

Si l'on souhaite récupérer complètement $|x\rangle$, il faut procéder à ce qu'on appelle une *tomographie*. La tomographie est une technique de reconstitution d'état quantique après de multiple mesure⁷. Cependant en faisant ça il faut procéder à un grand nombre polynomial ou exponentielle de mesure, mais donc l'algorithme HHL perd une nouvelle fois son intérêt.

Ainsi, une façon d'utiliser le résultat d'une unique mesure serait par exemple de s'intéresser au produit scalaire entre un vecteur z et x . Dans ce cas toutes les informations se déduisent après une mesure.

3.1.4 Analyse de la complexité et amélioration

Dans cette section, nous allons donner la complexité de l'algorithme HHL à travers le théorème 5 que nous ne démontrerons pas.

Théorème 5. *Soit le système $Ax = b$, où A est une matrice hermitienne de taille $N \times N$, s -creuse, de conditionnement κ et b est un vecteur unitaire de taille N . Alors la complexité en temps de l'algorithme HHL est donné par*

$$\mathcal{O}(\log(N)s^2\kappa^2/\epsilon) \quad (3.11)$$

où ϵ est la précision désiré.

6. On rappelle qu'une matrice $N \times N$ est s -creuse si chaque lignes contient au plus s coefficients non nulles. Pour $s \ll N$

7. Pour plus de détail sur le sujet cf. [15].

Problème	Algorithme	Complexité
Quantique	HHL	$\mathcal{O}(\log(N)s^2\kappa^2/\epsilon)$
Quantique	VTAA-HHL	$\mathcal{O}(\log(N)s^2\kappa/\epsilon)$
Quantique	QLSA	$\mathcal{O}(\sqrt{n}\log(N)\kappa^2/\epsilon)$
Classique	Gradient Conjugué	$\mathcal{O}(Ns\kappa\log(1/\epsilon))^8$
Classique	Pivot de Gauss	$\mathcal{O}(N^3)$

FIGURE 3.3 – Comparaison de la complexité des différents algorithmes.

Ainsi pour un s et un κ suffisamment petit, l'algorithme HHL démontre une vitesse exponentielle par rapport aux meilleurs algorithmes classiques. (cf. figure 3.3)

De plus depuis 2009, l'algorithme a subi des améliorations comme ici dans [3] (l'algorithme VTAA-HHL). Mais aussi une alternative basée sur la décomposition en valeur singulière développée dans l'article, celui-ci permet de résoudre un système linéaire pour une matrice dense [18] (algorithme QLSA). (cf. figure 3.3)

3.2 Algorithme de résolution de système linéaire hybride

[INTROREF] L'algorithme HHL est bien mais problème car trop de qubit et de porte donc mauvais pour le NISQ. On cherche une alternative avec moins de porte quitte à être hybride.

3.2.1 L'algorithme de Neumann-Ulam

Considérons un système d'équation de la forme $Ax = b$, où $A \in \mathcal{M}_n$, on suppose que l'on peut réécrire le système comme :

$$A = I - \gamma P \quad (3.12)$$

où $\gamma \in]0, 1[$ et soit P une matrice tel que $P_{i,j} > 0$ et $\sum_j P_{i,j} = 1$. De plus on suppose que $\|P\| < 1$.

[INTERET]

Si on écrit la solution on a :

$$x = A^{-1}b \quad (3.13)$$

$$= (I - \gamma P)^{-1}b \quad (3.14)$$

Nous aimerions exprimer cette solution sous forme de sommes. C'est pourquoi nous allons utiliser le théorème suivant :

Théorème 6 (Neumann ([26])). *Soit \mathcal{H} un espace d'Hilbert et P un opérateur tel que $\|P\| < 1$, alors*

$$(I - P)^{-1} = \sum_s P^s$$

Remarque 21. *Avant la démonstration, on pourra remarquer l'analogie avec le développement limité de $\frac{1}{1-x} = \sum_k x^k$*

Démonstration. Soit $n \in \mathbb{N}$ et soit S_n la série géométrique partielle définit par :

$$S_n(u) = (1 + P + P^2 + \dots + P^n)(u)$$

Soit m un entier supérieur à n , alors :

$$\|S_m(u) - S_n(u)\| = \|(P^{n+1} + \dots + P^m)(u)\| \leq (\|P^{n+1}\| + \dots + \|P^m\|)(\|u\|) \quad (3.15)$$

De ce fait, la suite $(S_n(u))_{n \in \mathbb{N}}$ est une suite de Cauchy, or nous par définition d'un espace d'Hilbert la suite de Cauchy converge. En particulier, on pose $S_n(u) \rightarrow S$. De plus on rappelle que $(I - P)(I + P + \dots + P^n) = I - P^{n+1}$

Ainsi, on a les égalités suivante :

$$\begin{aligned} (I - P)S &= (I - P) \lim_{n \rightarrow \infty} S_n(u) \\ &= \lim_{n \rightarrow \infty} (I - P)S_n(u) \\ &= \lim_{n \rightarrow \infty} (I - P^{n+1})(u) \\ &= \lim_{n \rightarrow \infty} u - P^{n+1}(u) \\ &= u \end{aligned}$$

□

On déduit de ce théorème et de l'équation 3.14 l'équation suivant :

$$x = \sum_{s=0}^{\infty} P^s \gamma^s b \quad (3.16)$$

De plus la i -ème composante de x tronqué à partir du c -ième terme qu'on note x_i^c est donnée par :

$$x_i^c = b_i + \gamma \sum_{j=1}^n P_{ij} b_j + \gamma^2 \sum_{j_1=1}^n \sum_{j_2=1}^n P_{ij_1} P_{j_1 j_2} b_{j_2} + \dots + \gamma^c \sum_{j_1=1}^n \dots \sum_{j_c=1}^n P_{ij_1} \dots P_{j_{c-1} j_c} b_{j_c} \quad (3.17)$$

$$= \sum_{s=0}^c \gamma^s \sum_{i_1, \dots, i_s=1}^n P_{i_0 i_1} \dots P_{i_{s-1} i_s} b_{i_s} \quad (3.18)$$

On va construire une interprétation probabiliste de l'équation 3.17 en utilisant *des marches aléatoires*.

Tout d'abord faisons une rapide aparté sur ce qu'est une marche aléatoire :

Soit un système pouvant être dans un des n états possibles notés S_1, \dots, S_n , soit X_t l'état du système à l'étape t . On dit que X_t est une marche aléatoire, si la probabilité que le système à l'étape t soit dans l'état S_j dépend uniquement de l'état du système à l'état $t - 1$. C'est à dire $\mathbb{P}(X_t = S_j | X_{t-1} = S_{j-1})$.

Une façon visuelle de se l'imaginer et de prendre un graphe où chaque arrêtes possèdent un poids, tel que la somme des poids des arrêtes partant d'un sommet soit égale à 1.

Dans un tel graphe les sommets représentent les états S_i , et le poids d'une arrêtes allant de S_i, S_j la probabilité que le système passe de l'état S_i à l'état S_j . Ainsi, on peut représenter l'état du système par une fournis sur le graphe qui se baladerait de sommets en sommets de façon aléatoire en fonction des poids des arêtes (cf. figure 3.4).

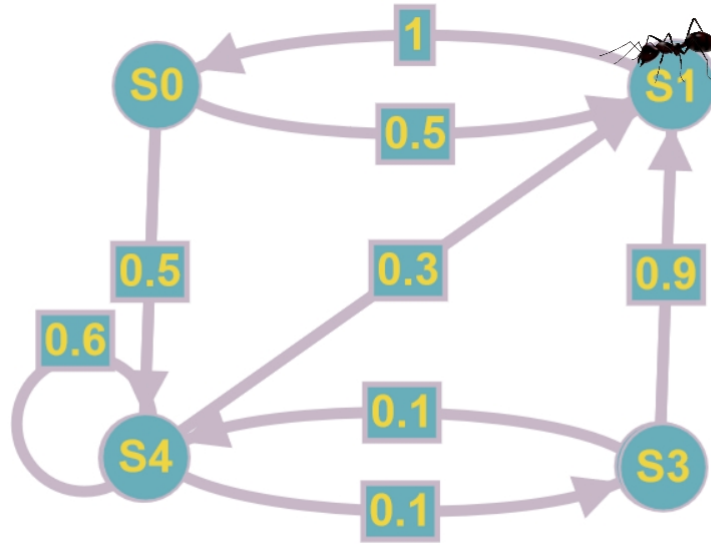


FIGURE 3.4 – Exemple de marche aléatoire

De plus, comme il s'agit d'un graphe, nous pouvons le représenter sous formes d'une matrice $H \in \mathcal{M}_n$, où H_{ij} est donné par le poids de l'arrête allant du sommet S_i au sommet S_j . Dans le cas des marches aléatoire cette matrice est appelée *matrice de transition*. Par exemple, la matrice de transition du graphe de la figure 3.4 est donné par :

$$H = \begin{pmatrix} 0 & 0.5 & 0.5 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.6 & 0.1 & 0 \\ 0 & 0.9 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Achevons cette aparté, en remarquant qu'une matrice de transition possède les mêmes propriétés que la matrice P définit dans l'équation 3.12. Ainsi on peut raisonner sur la matrice P comme une marche aléatoire dont les sommets sont notés $(1, \dots, n)$.

Maintenant, nous allons jouer à un jeu pour estimer $x_{i_0}^c$. Considérons un chemin d'un marche aléatoire I qui commence au sommet i_0 .

Avant chaque étapes on tire une pièce truqué qui est face avec une probabilité γ et pile avec une probabilité $1 - \gamma$.

- Si on obtient pile on arrête la marche.
- Si on obtient face on continue la marche.
- De plus si on a déjà fait c étape on arrête la marche aussi.

Après que la marche ait été arrêter on a un chemin $I = (i_1, \dots, i_k)$, donc k est la longueur du chemin.

A chaque chemin on associe un score :

$$\text{Score}(I) = \sum_j^k b_{i_j}$$

Théorème 7. Soit x la solution du système 3.12. Alors on a l'estimation de la i_0 -ième composante de x suivante :

$$x_{i_0}^c = \mathbb{E}(\sum_I \mathbb{P}(I) \text{Score}(I)) \quad (3.19)$$

où I est un chemin partant de i_0

Démonstration. Soit un chemin $I = (i_1, \dots, i_k)$, d'abord remarquons que

$$\mathbb{P}(\text{longueur}(I) \geq s) = \gamma^s$$

et

$$\mathbb{P}(i_1 = v_1, \dots, i_k = v_k) = \prod_j^k P_{v_{j-1}v_j}$$

où $v_0 = i_0$.

Revenons sur l'équation 3.19, on a :

$$\begin{aligned} \sum_I \mathbb{P}(I) \text{Score}(I) &= \sum_I \mathbb{P}(I) \sum_{s=0}^{\text{longueur de } I} b_{i_s} \\ &= \sum_{s=0}^c \sum_V \mathbb{P}(i_1 = v_1, \dots, i_s = v_s | \text{longueur}(I) \geq s) b_{v_s} \text{ où } V \text{ sont les chemins commençant à } i_0 \\ &= \sum_{s=0}^c \gamma^s \sum_{i_0, \dots, i_s} P_{i_0 i_1} \dots P_{i_{s-1} i_s} b_{i_s} \end{aligned}$$

Ainsi en prenant la moyenne sur un grands nombres de chemins on obtient bien l'égalité 3.19. \square

3.2.2 l'algorithme quantique

Le cube de Hamming

Une première amélioration par rapport à l'algorithme quantique vient du stockage de la matrice P .

Pour un ordinateur classique stocker la matrice P de taille N , nécessite au moins $\mathcal{O}(N)$. Alors, qu'un ordinateur quantique utilise uniquement $\mathcal{O}(\log_2(N))$ qubits.

En effet, soit un graphe à $N = 4$ noeuds comme sur la figure 3.4, alors on peut représenter les noeuds S_i en utilisant des chaînes de bit. Le noeud S_0 est représenté par le ket $|00\rangle$, le noeud S_1 par le ket $|01\rangle$, et ainsi de suite. Ainsi on aura utilisé uniquement $2 = \log_2(4)$ qubits pour stocker la matrice P . Cette façon de représenter les noeuds en utilisant des chaînes de bits, est appelée *le cube de Hamming*.

Cela peut être aisément généraliser pour un N quelconque.

Marche aléatoire quantique

Appendice

3.3 Théorème de non-clonage

Le théorème de non-clonage a été introduit en 1982 dans Nature [40].

Le théorème de non-clonage a de nombreuses conséquences, que se soit en calcul quantique, comme nous avons pu le voir dans la section 1.1.3. Mais aussi en philosophie, etc...

Théorème 8 (Non-clonage). *Il est impossible de faire une copie de l'état d'un qubit inconnue.*

Démonstration. Tout d'abord, commençons par traduire l'énoncé dans le formalisme de la mécanique quantique. On a un qubit⁹ dans un état $|\psi\rangle$, celui-ci correspondra à notre matériel de base que l'on souhaite copier. On a un second qubit dans un état $|\phi\rangle$ le qubit que l'on souhaite cloner dans l'état $|\psi\rangle$.

Une première idée, serait de faire une mesure de $|\psi\rangle$, cependant en faisant cela on perdrait une partie de l'information. Ce n'est donc pas la bonne méthode. De ce fait on cherche une porte quantique U tel que :

$$U(|\psi\rangle |\phi\rangle) = |\psi\rangle |\psi\rangle \text{ pour tout qubits}$$

En particulier on peut prendre comme "matériel de base" un qubit $|\Delta\rangle$ vérifiant aussi :

$$U(|\Delta\rangle |\phi\rangle) = |\Delta\rangle |\Delta\rangle$$

Comme U est un opérateur unitaire, on a $U^*U = I$, et donc :

$$\begin{aligned} \langle\phi| \langle\psi| U^*U |\Delta\rangle |\phi\rangle &= \langle\psi| \langle\psi| |\Delta\rangle |\Delta\rangle \\ \langle\psi| |\Delta\rangle &= \langle\psi| \langle\psi| |\Delta\rangle |\Delta\rangle \\ \langle\psi| |\Delta\rangle &= \langle\psi| |\Delta\rangle^2 \end{aligned}$$

D'où soit $\langle\psi| |\Delta\rangle = 0$ et donc les deux états sont orthogonaux ou soit $\langle\psi| |\Delta\rangle = 1$. Ce qui dans les deux cas engendre une contradiction, car les deux états ont été supposés quelconque.

Ainsi il n'existe pas d'opérateur U permettant de cloner l'état d'un qubit inconnue. \square

Remarque 22. *Il existe une autre démonstration, à l'aide de la théorie des catégories, où la notion de no-cloning est bien plus évidente.*¹⁰

3.4 État pure et état mixte

On va définir ici, la notion d'état mixte, pour cela nous avons utilisé le livre [28].

La plupart des états que nous utilisons du type $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, sont ce qu'on appelle des *états purs*. Cependant supposons qu'on ait un qubit dans l'état pure $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ avec une probabilité 1/3 et dans l'état pure $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ avec une probabilité 2/3. Dans ce cas on dit que c'est un *état mixte*.

9. On raisonne sur des qubits, cependant cela peut être très facilement généralisé à des états quantique quelconque

10. Pour une introduction à la théorie des catégories [22] et pour l'article sur le théorème de non-clonage [2].

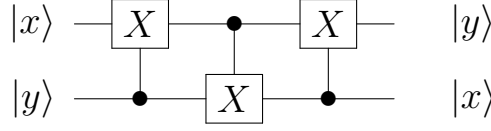


FIGURE 3.5 – Construction de porte SWAP

Une des façon de représenter un état mixte arbitraire de n qubits, est d'utiliser l'ensemble :

$$\{ (|\psi_1\rangle, p_1), (|\psi_2\rangle, p_2), \dots, (|\psi_k\rangle, p_k) \} \quad (3.20)$$

Pour dire que le qubit se trouve dans l'état $|\psi_i\rangle$ avec une probabilité p_i , où $i \in \{1, \dots, k\}$. Notons qu'un état pure, peut être vu comme un état mixte avec tout les $p_i = 0$ sauf un.

Cependant l'écriture ci-dessus, n'est pas très convenable. C'est pourquoi, on utilise une écriture sous forme d'opérateur. On appelle *opérateur de densité* pour un état mixte arbitraire représenté par ρ , l'opérateur ρ défini par la formule suivante :

$$\rho = \sum_{i=1}^k p_i |\psi_i\rangle \langle \psi_i| \quad (3.21)$$

Remarque 23. *Il ne faut pas s'étonner de voir utiliser dans la littérature un opérateur de densité, comme s'il s'agissait d'un ket.*

On voudrait savoir ce qu'il si on applique une opérateur unitaire à un état mixte. La première idée naturelle qui nous vient est d'appliquer l'opérateur à chaque état $|\psi_i\rangle$. Appliquons donc un opérateur U à la définition d'un état mixte comme ensemble (équation 3.20) :

$$\{ (U |\psi_1\rangle, p_1), (U |\psi_2\rangle, p_2), \dots, (U |\psi_k\rangle, p_k) \} \quad (3.22)$$

qui a comme opérateur de densité :

$$\begin{aligned} \rho &= \sum_{i=1}^k p_i U |\psi_i\rangle \langle \psi_i| U^* \\ &= U \left(\sum_{i=1}^k p_i |\psi_i\rangle \langle \psi_i| \right) U^* \\ &= U \rho U^* \end{aligned}$$

3.5 Porte Quantique SWAP

la porte SWAP est une porte sur deux qubits, qui échange deux qubits $|x\rangle |y\rangle \mapsto |y\rangle |x\rangle$, on la construit comme une suite de porte CNOT comme sur la figure 3.5.

De plus on la note avec des croix cf. figure 3.6.

3.6 Théorème de Schmidt

Cette section est entièrement inspiré de l'article [24].

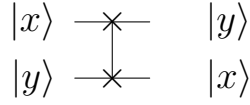


FIGURE 3.6 – Notation porte SWAP

Soit $|\psi\rangle$ un vecteur d'un produit tensoriel d'espace $\mathcal{H}_A \otimes \mathcal{H}_B$, où $\dim(\mathcal{H}_A) = m$ et $\dim(\mathcal{H}_B) = m$. Alors, il existe une base orthonormée $|f_i^A\rangle$ et $|f_i^B\rangle$ de respectivement \mathcal{H}_A et \mathcal{H}_B , il est évident qu'on peut décomposer $|\psi\rangle$ comme :

$$|\psi\rangle = \sum_{1 \leq i \leq n, 1 \leq j \leq m} \beta_{ij} |f_i^A\rangle \otimes |f_j^B\rangle \quad (3.23)$$

L'équation 3.23, nous donne une décomposition avec une somme sur deux indices. Dans cette section, nous allons donner un théorème qui enlève les termes croisés de l'équation 3.23.

Théorème 9 (Décomposition de Schmidt). *Soit $|\psi\rangle$ un vecteur d'un produit tensoriel d'espace $\mathcal{H}_A \otimes \mathcal{H}_B$, où $\dim(\mathcal{H}_A) = m$ et $\dim(\mathcal{H}_B) = m$. On suppose sans perte de généralité que $n \geq m$. Alors, il existe une base orthonormée $|e_i^A\rangle$ et $|e_i^B\rangle$ de respectivement \mathcal{H}_A et \mathcal{H}_B , et des scalaires positif $p_i \in \mathbb{C}$ tel que :*

$$|\psi\rangle = \sum_i^n \sqrt{p_i} |e_i^A\rangle \otimes |e_i^B\rangle \quad (3.24)$$

Démonstration. Partons de l'équation 3.23, on peut réécrire $|\psi\rangle$ sous forme matricielle comme $M_{|\psi\rangle} \in \mathcal{M}_{n \times m}(\mathbb{C})$

$$M_{|\psi\rangle} = (\beta_{ij})_{1 \leq i \leq m, 1 \leq j \leq n} \quad (3.25)$$

On utilise la décomposition en valeur singulière de $M_{|\psi\rangle}$.

On sait qu'il existe $U \in \mathcal{M}_n(\mathbb{C})$, $V \in \mathcal{M}_m(\mathbb{C})$ des matrices unitaires, et $\Sigma \in \mathcal{M}_m(\mathbb{C})$ une matrice semi-définie positive tel que :

$$M_{|\psi\rangle} = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^* \quad (3.26)$$

On note $U = \begin{pmatrix} U_1 & U_2 \end{pmatrix}$ où $U_1 \in \mathcal{M}_{n \times m}(\mathbb{C})$. Dans ce cas, le terme U_2 s'annule, d'où on peut réécrire l'équation 3.26 comme

$$M_{|\psi\rangle} = U_1 \Sigma V^* \quad (3.27)$$

D'où si on note $|e_i^A\rangle$ les m premières colonnes de U_1 , $|e_i^B\rangle$ les colonnes de V , soit α_i les coefficients de la diagonales de Σ , alors l'équation 3.27 s'écrit finalement :

$$M_{|\psi\rangle} = \sum_i^n \alpha_i |e_i^A\rangle \langle e_i^B|$$

Comme les α_i peuvent être vu comme la racine carré des valeurs propres de la matrice $M_{|\psi\rangle} M_{|\psi\rangle}^*$. On abouti bien au résultat de l'équation 3.24. \square

Formulaire

Analyse Hilbertienne

Définition (Espace de Hilbert). On appelle espace de Hilbert complexe un espace vectoriel complexe muni d'un produit hermitien $\langle . | . \rangle$, qui est complet pour la norme associée $x \mapsto \|x\| = \sqrt{\langle x | x \rangle}$.

Notation (de Dirac). Les notations de Dirac consistent, entre autres, à représenter les vecteurs d'un espace de Hilbert sous forme de ket : $|x\rangle$.

Définition (Base orthonormée). Soit \mathcal{H} un espace de Hilbert complexe de dimension N . Une famille de vecteurs de \mathcal{H} $(|x_i\rangle)_{i \in \{1, \dots, N\}}$ est une base orthonormée de \mathcal{H} lorsque :

- ces vecteurs sont deux à deux orthogonaux : $\forall i \neq j, \langle x_i | x_j \rangle = 0$;
- ces vecteurs sont normés : $\forall i \in \{1, \dots, N\}, \langle x_i | x_i \rangle = 1$.

Proposition (Opérateur adjoint). Soient \mathcal{H} un espace de Hilbert et $U : \mathcal{H} \rightarrow \mathcal{H}$ un opérateur linéaire. Il existe un unique opérateur U^* , appelé adjoint de U , tel que $\langle Ux | y \rangle = \langle x | U^*y \rangle$ pour tous $x, y \in \mathcal{H}$

Définition (Opérateur unitaire). Soit \mathcal{H} un espace de Hilbert. Un opérateur linéaire $U : \mathcal{H} \rightarrow \mathcal{H}$ est dit unitaire lorsqu'il vérifie $UU^* = U^*U = I$, où I désigne l'identité. Il est équivalent de dire que U conserve le produit scalaire, ou bien la norme : $\forall x, y \in \mathcal{H}, \langle Ux | Uy \rangle = \langle x | y \rangle$ et $\|Ux\| = \|x\|$.

Postulats

Postulat 1 (Espace des états). L'état d'un système quantique peut être décrit par une droite dans un espace de Hilbert.











Postulat 2 (Évolution). L'évolution entre deux instants d'un système quantique isolé est décrit par un opérateur unitaire.



Postulat 3 (Mesure - première version). Étant donnée une base orthonormée $(|e_i\rangle)_{i \in I}$ d'un espace de Hilbert, il est possible d'effectuer une mesure relativement à cette base qui, pour un système dans l'état $|\psi\rangle = \sum_i a_i |e_i\rangle$ va renvoyer le résultat i avec probabilité $|a_i|^2$ et laisser le système dans l'état $|e_i\rangle$.

Postulat 4 (Composition de systèmes). Lorsque deux systèmes quantiques ayant respectivement comme espace d'états \mathcal{H}_1 et \mathcal{H}_2 sont regardés comme un seul système quantique, ce dernier a pour espace d'états l'espace $\mathcal{H}_1 \otimes \mathcal{H}_2$. Si le premier système est dans l'état $|\psi_1\rangle$ et le deuxième dans l'état $|\psi_2\rangle$, alors le système composite est dans l'état $|\psi_1\rangle \otimes |\psi_2\rangle$.

Postulat 5 (Mesure). Soit $|e_1\rangle, \dots, |e_n\rangle$ est une base orthonormée d'un espace \mathcal{H}_1 , et soit $|\psi_1\rangle, \dots, |\psi_n\rangle$ une famille de vecteurs normés (pas forcément orthogonaux) d'un espace \mathcal{H}_2 . Alors la mesure relativement à la base $(|e_i\rangle)_i$ d'un système dans l'état $|\psi\rangle = \sum_i a_i |e_i\rangle |\psi_i\rangle$, avec $\sum_i |a_i|^2 = 1$ va renvoyer i avec probabilité $|a_i|^2$ et laisser alors le système dans l'état $|e_i\rangle |\psi_i\rangle$.

Portes quantiques

Opérateur	Notation dans un circuit	Matrice
$I : \begin{cases} 0\rangle \mapsto 0\rangle \\ 1\rangle \mapsto 1\rangle \end{cases}$		$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
$X : \begin{cases} 0\rangle \mapsto 1\rangle \\ 1\rangle \mapsto 0\rangle \end{cases}$		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
$Z : \begin{cases} 0\rangle \mapsto 0\rangle \\ 1\rangle \mapsto - 1\rangle \end{cases}$		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
$Y : \begin{cases} 0\rangle \mapsto i 1\rangle \\ 1\rangle \mapsto -i 0\rangle \end{cases}$		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
$H : \begin{cases} 0\rangle \mapsto \frac{1}{\sqrt{2}}(0\rangle + 1\rangle) \\ 1\rangle \mapsto \frac{1}{\sqrt{2}}(0\rangle - 1\rangle) \end{cases}$		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
$T : \begin{cases} 0\rangle \mapsto 0\rangle \\ 1\rangle \mapsto e^{i\pi/4} 1\rangle \end{cases}$		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$
$S : \begin{cases} 0\rangle \mapsto 0\rangle \\ 1\rangle \mapsto i 1\rangle \end{cases}$		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
$R_k : \begin{cases} 0\rangle \mapsto 0\rangle \\ 1\rangle \mapsto e^{i\frac{2\pi}{2^k}} 1\rangle \end{cases}$		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{2\pi}{2^k}} \end{pmatrix}$
$\text{CNOT} : \begin{cases} 0\rangle x\rangle \mapsto 0\rangle x\rangle \\ 1\rangle x\rangle \mapsto 1\rangle X x\rangle \end{cases}$		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
$\text{SWAP} : x\rangle y\rangle \mapsto y\rangle x\rangle$		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

$\mathbf{c}\text{-}U : \begin{cases} 0\rangle x\rangle & \mapsto 0\rangle x\rangle \\ 1\rangle x\rangle & \mapsto 1\rangle U x\rangle \end{cases}$		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{pmatrix}$
$\text{QFT} : j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j \frac{k}{N}} k\rangle$		

Bibliographie

- [1] Scott Aaronson. Quantum machine learning algorithms : Read the fine print. *Nature* disponible sur <https://scottaaronson.com/papers/qml.pdf>, 2015.
- [2] Samson Abramsky. No-cloning in categorical quantum mechanics. *Arxiv* disponible sur <https://arxiv.org/pdf/quant-ph/0101034.pdf>, 2012.
- [3] Andris Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. *Arxiv* disponible sur <https://arxiv.org/pdf/1010.4458.pdf>, 2010.
- [4] Enrico Deotto Edward Farhi Sam Gutmann Daniel A. Spielman Andrew M. Childs, Richard Cleve. Exponential algorithmic speedup by quantum walk. *Arxiv* disponible sur <https://arxiv.org/pdf/quant-ph/0209131.pdf>, 2002.
- [5] Sarah Sheldon Paul D. Nation Jay M. Gambetta Andrew W. Cross, Lev S. Bishop. Validating quantum computers using randomized model circuits. *IBM*, 2018. Disponible sur <https://arxiv.org/pdf/1811.12926.pdf>.
- [6] Avinandan Hassidim Aram W. Harrow and Seth Lloyd. Quantum algorithm for linear systems of equations. *Arxiv* disponible sur <https://arxiv.org/pdf/0811.3171v3.pdf>, 2009.
- [7] Peter Mountney Simone Severini Nairi Usher Leonard Wossnig Danial Dervovic, Mark Herbster. Quantum linear systems algorithms : a primer. *Arxiv* disponible sur <https://arxiv.org/pdf/0811.3171v3.pdf>, 2009.
- [8] Karol Życzkowski David W. Kribs, Aron Pasiaka. Entropy of a quantum error correction code. *World Scientific Publishing Company* - disponible sur <http://chaos.if.uj.edu.pl/~karol/pdf/KPZ08.pdf>, 2008.
- [9] Andrew M. Childs Dominic W. Berry. Black-box hamiltonian simulation and unitary implementation. *Arxiv* disponible sur <https://arxiv.org/pdf/0910.4157.pdf>, 2011.
- [10] Michele Mosca Alain Tapp Gilles Brassard, Peter Høyer. Quantum amplitude amplification and estimation. *Arxiv* disponible sur <https://arxiv.org/pdf/quant-ph/0005055.pdf>, 2000.
- [11] Patrick Gros. Introduction à l'algèbre tensorielle. *INRIA*, 2000.
- [12] Lov K. Grover. A fast quantum mechanical algorithm for database search. *Arxiv*, 1996.
- [13] Isaac L. Chuang Guang Hao Low. Hamiltonian simulation by uniform spectral amplification. *Arxiv* disponible sur <https://arxiv.org/pdf/1707.05391.pdf>, 2017.
- [14] Yang Sun Gui-Lu Long. Efficient scheme for initializing a quantum register with an arbitrary superposed state. *Arxiv* disponible sur <https://arxiv.org/pdf/quant-ph/0104030.pdf>, 2001.
- [15] P.G. Kwia J. B. Altepeter, D.F. V. James. Quantum state tomography. Disponible sur http://research.physics.illinois.edu/QI/Photonics/tomography-files/tomo_chapter_2004.pdf, 2004.
- [16] Ville Bergholm Jacob Biamonte. Quantum tensor networks in a nutshell. *Arxiv* disponible sur <https://arxiv.org/pdf/1708.00006.pdf>, 2017.
- [17] Eric Tannier Sébastien Varrette Jean-Guillaume Dumas, Jean-Louis Roch. *Théorie des codes : Compression, cryptage, correction*. Dunod, 2006.
- [18] Anupam Prakash Leonard Wossnig, Zhikuan Zhao. A quantum linear system algorithm for dense matrices. *Arxiv* disponible sur <https://arxiv.org/pdf/1704.06174.pdf>, 2017.
- [19] Andrew Cross Jay M. Gambetta John Smolin Lev S. Bishop, Sergey Bravyi. Quantum volume. *IBM*, 2017. Disponible sur <https://pdfs.semanticscholar.org/650c/3fa2a231cd77cf3d882e1659ee14175c01d5.pdf>.

- [20] Terry Rudolph Lov Grover. Creating superpositions that correspond to efficiently integrable probability distributions. *Arxiv disponible sur* <https://arxiv.org/pdf/quant-ph/0208112.pdf>, 2002.
- [21] David Algis Léo Bois. Introduction au calcul quantique. *Disponible sur* https://drive.google.com/open?id=1NEHP3_Myh1eO1qH3vq3v4R-midHBWX3A, 2019.
- [22] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.
- [23] Isaac L.Chuang Michael A. Nielsen. *Quantum Computation and Quantum Information*. Cambridge, 2000.
- [24] Jaroslaw Adam Miszczak. Singular value decomposition and matrix reorderings in quantum information theory. *Arxiv disponible sur* <https://arxiv.org/pdf/1807.10651.pdf>, 2011.
- [25] Michael A. Nielsen. A simple formula for the average gate fidelity of a quantum dynamical operation. *Arxiv disponible sur* <https://arxiv.org/pdf/quant-ph/0205035.pdf>, 2008.
- [26] Sheehan Olver. Lesson 15 - functional analysis. *Disponible sur* <http://www.maths.usyd.edu.au/u/olver/teaching/NCA/15.pdf>, 2013.
- [27] Scott Pakin ... Patrick J. Coles, Stephan Eidenbenz. Quantum algorithm implementations for beginners. *Arxiv disponible sur* <https://arxiv.org/pdf/1804.03719.pdf>, 2018.
- [28] Michele Mosca Phillip Kaye, Raymon Laflamme. *An Introduction to Quantum Computing*. Oxford University Press Inc., New York, 2007.
- [29] Anupam Prakash. Quantum algorithms for linear algebra and machinelearning. *Berkeley disponible sur* <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.pdf>, 2014.
- [30] John Preskill. Quantum computing in the nisc era and beyond. *Arxiv*, 2018. Disponible sur <https://arxiv.org/pdf/1801.00862.pdf>.
- [31] Juan Pablo Paz Wojciech Hubert Zurek Raymond Laflamme, Cesar Miquel. Perfect quantum error correction code. *Arxiv disponible sur* <https://arxiv.org/pdf/quant-ph/9602019.pdf>, 2008.
- [32] R.Martinez C.Negrevergne R.Laflamme, E.Knill. Implementation of the five qubit error correction benchmark. *Arxiv disponible sur* <https://arxiv.org/pdf/quant-ph/0101034.pdf>, 2018.
- [33] John A. Smolin Sergey Bravyi, Graeme Smith. Trading classical and quantum computational resources. *Arxiv disponible sur* <https://arxiv.org/pdf/1506.01396.pdf>, 2015.
- [34] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. *School of Computer Science Carnegie Mellon University Pittsburgh disponible sur* <https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf>, 1994.
- [35] Kae Nemoto Simon J. Devitt, William J. Munro. Quantum error correction for beginners. *Arxiv disponible sur* <https://arxiv.org/pdf/0905.2794.pdf>, 2013.
- [36] Michael Sipser. *Introduction to the Theory of Computation*. Wadsworth Publishing, 2012.
- [37] Tomas Jochym-O'Connor Michele Mosca Priyaa Varshinee Srinivasan Srinivasan Arunachalam, Vlad Gheorghiu. On the robustness of bucket brigade quantumram. *New Journal of Physics disponible sur* <https://iopscience.iop.org/article/10.1088/1367-2630/17/12/123010/pdf>, 2015.
- [38] Moinuddin K. Qureshi Swamit S. Tannu. Not all qubits are created equal - a case for variability-aware policies for nisc-era quantum computers. *Arxiv*, 2018. Disponible sur <https://arxiv.org/ftp/arxiv/papers/1805/1805.10224.pdf>.
- [39] Maris Ozols-Xiaodi Wu Tianyi Peng, Aram W.Harrow. Simulating large quantum circuits on a small quantum computer. *Arxiv disponible sur* <https://arxiv.org/pdf/1904.00102.pdf>, 2019.
- [40] W. H. Zurek W. K. Wootters. A single quantum cannot be cloned. *Nature* 299 pp. 802-803., 1982.
- [41] Peter Wittek. *Quantum Machine Learning*. Elsevier, 2014.
- [42] Peter W.Shor. Proceedings of the 35th annual symposium on foundations of computer science pp. 124-134. 1994.
- [43] Steven Frankel-Sabre Kais Yudong Cao, Anmer Daskin. Quantum circuits for solving linear systems of equations. *Arxiv disponible sur* <https://arxiv.org/pdf/1110.2232.pdf>, 2013.