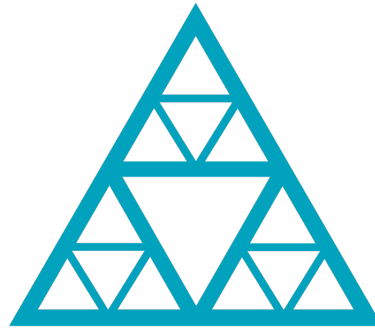




PARCOURS RECHERCHE - 2024

RÉSOLUTION QUANTIQUE DE L'ÉQUATION DE POISSON

Clément Mazzocchi et Paul Tritz-Heid
sous la supervision d'Eric Cances

Résumé

L'informatique quantique est vouée à produire une puissance de calcul nettement supérieure à celles que nous connaissons aujourd'hui. Une de ses applications est la résolution de grands systèmes linéaires. Nous étudions deux algorithmes traitant de ce problème, et les appliquons à la résolution de l'équation de Poisson. L'algorithme Harrow-Hassidim-Lloyd donne une première solution avec une augmentation exponentielle de la vitesse de résolution, sous certaines conditions, par-rapport aux meilleurs algorithmes classiques. Nous nous intéressons ensuite aux algorithmes variationnels, qui reposent sur l'optimisation classique d'un circuit quantique paramétré permettant la convergence vers la solution. Cet algorithme nécessite d'établir préalablement la décomposition de la matrice du système en chaînes de Pauli. Nous effectuons des simulations quantiques permettant d'évaluer l'efficacité de ces algorithmes et de comparer leurs solutions aux techniques classiques.

Abstract

Quantum computing is doomed to produce a much greater computing power than the ones known today. One of its applications is the resolution of big linear systems. We study two algorithms that deal with this issue before using them to solve Poisson's equation. The Harrow-Hassidim-Lloyd algorithm provides under certain assumptions, a first solution with an exponential speedup, compared to the best classical algorithms. Then, we focus on variational algorithms, especially the Variational Quantum Linear Solver. They rely on the classical optimisation of a parameterised quantum circuit. This process enables to converge towards the solution. This algorithm requires to compute the Pauli decomposition of the matrix of the system before running it. We simulate the quantum circuits obtained to assess the efficiency of these algorithms and compare their solutions to the ones of classical techniques.

Mots-clés

- Informatique quantique
- Équation de Poisson
- Algorithme Harrow-Hassidim-Lloyd
- Estimation de phase quantique
- Transformée de Fourier quantique
- Algorithme variationnel
- Décomposition en chaînes de Pauli

Keywords

- Quantum computing
- Poisson's equation
- Harrow-Hassidim-Lloyd algorithm
- Quantum phase estimation
- Quantum Fourier transform
- Variational algorithm
- Tensor product decomposition

Table des matières

1	Introduction	1
2	Préliminaires de calcul quantique	1
2.1	Postulats de la mécanique quantique	1
2.1.1	Postulat d'espace d'états : le qubit	1
2.1.2	Postulat d'opérateur quantique : la porte quantique	2
2.1.3	Postulat de mesure quantique	2
2.1.4	Postulat du produit tensoriel	3
2.2	Opérateur de densité	3
2.3	Circuit quantique	3
2.4	Opérations de copie et théorème de non-clonage	4
2.5	Mesure	4
2.6	Accroissement linéaire de l'erreur	4
2.7	Ensembles universels de portes et calcul réversible	4
2.8	Représentation des nombres en virgule fixe et opérations arithmétiques classiques	5
3	Outils pour la résolution de systèmes linéaires	5
3.1	Test de Hadamard	5
3.2	<i>Swap test</i>	6
3.3	Transformée de Fourier quantique	7
3.4	Estimation de phase quantique	7
3.4.1	Avec le test de Hadamard	7
3.4.2	Avec la transformée de Fourier quantique	8
4	Algorithme Harrow-Hassidim-Lloyd	8
5	Algorithmes variationnels	9
5.1	Principe général	9
5.2	Variational Quantum Eigensolver	10
5.3	Variational Quantum Linear Solver	11
5.3.1	Hamiltonien et fonction coût associés	11
5.3.2	Calcul de $\langle \psi(\boldsymbol{\theta}) \psi(\boldsymbol{\theta}) \rangle$	12
5.3.3	Calcul de $ \langle b \psi(\boldsymbol{\theta}) \rangle ^2$	13
6	Résultats	14
7	Conclusion et ouverture	15
8	Annexes	I
8.1	Implémentation de la transformée de Fourier quantique	I
8.2	Détails techniques pour la résolution de l'équation de Poisson	II
8.2.1	Discretisation	II
8.2.2	Diagonalisation de A pour l'algorithme HHL	II
8.2.3	Décomposition de A en chaînes de Pauli pour l'algorithme VQLS	III
8.2.4	Décomposition de A^2 en chaînes de Pauli pour l'algorithme VQLS	III
8.3	Détail du calcul de l'algorithme HHL	IV
8.3.1	Préparation de l'état	IV
8.3.2	Estimation de phase quantique	IV
8.3.3	Rotation du registre auxiliaire et mesure	V
8.3.4	Estimation de phase quantique inverse	VI
8.4	Préparation de l'ansatz des algorithmes variationnels	VI
8.5	Code pour l'algorithme HHL	VII
8.6	Code pour l'algorithme VQLS	VIII
8.6.1	Importation des bibliothèques nécessaires	VIII
8.6.2	Préparation de l'ansatz	VIII
8.6.3	Décompositions en chaînes de Pauli	VIII
8.6.4	Observables	XIV
8.6.5	Optimisation de la fonction coût	XV

1 Introduction

Ce rapport est réalisé dans le cadre du "Parcours recherche", dispositif mis en place par l'École des Ponts permettant à des étudiants volontaires de s'investir durant plusieurs mois sur une question spécifique. Le but de cette démarche est de s'approprier les méthodes de la recherche.

Nous avons choisi de nous engager dans cette modalité. Parmi les projets de recherche qui nous étaient proposés, celui portant sur l'informatique quantique, porté par Eric Cances, enseignant à l'École et chercheur au CERMICS, nous a complètement convaincus. Nous nous sommes donc intéressés, avec un autre binôme, à ce champ actif de la recherche contemporaine, qui fascine autant qu'il soulève de questions, d'ordre scientifique certes, mais également stratégique, économique voire géopolitique.

L'informatique quantique tire profit de phénomènes quantiques tels que la superposition et l'intrication pour résoudre des problèmes complexes en un temps réduit par-rapport à un traitement classique. Elle repose sur des technologies de pointe, comme les ordinateurs quantiques, encore en développement.

L'élément central de l'informatique quantique est le qubit, l'unité élémentaire d'information quantique, analogue du bit classique. Tout l'intérêt du calcul quantique repose sur le fait qu'un qubit donné peut être dans un état de superposition des deux qubits élémentaires $|0\rangle$ et $|1\rangle$. Cette propriété n'a bien sûr pas d'équivalent en informatique classique, où un bit est soit dans l'état 0, soit dans l'état 1, mais jamais dans une superposition de ces deux états. Un autre élément d'importance est le phénomène d'intrication quantique, qui permet de lier des qubits distincts. Là encore, ce phénomène est purement quantique.

Après nous être familiarisés avec les bases de l'informatique quantique et avoir appris à manipuler les circuits quantiques, qui permettent de représenter de manière visuelle un algorithme quantique au moyen de portes quantiques agissant sur les différents qubits, nous nous sommes centrés sur l'approche d'un problème spécifique. Sur les conseils de notre encadrant, nous avons orienté nos travaux vers la résolution de systèmes linéaires par des algorithmes quantiques. Rapidement, nous avons souhaité utiliser nos résultats pour résoudre des équations différentielles linéaires. C'est pourquoi notre rendu final porte sur la résolution quantique de l'équation de Poisson $-\Delta u = f$.

Nous avons étudié deux algorithmes permettant cette résolution. Le premier est l'algorithme de Harrow-Hassidim-Lloyd (HHL), algorithme de référence pour la résolution de systèmes linéaires sur un ordinateur quantique. Un autre type d'algorithmes nous a été présenté : il s'agit des algorithmes variationnels qui optimisent une quantité classique en fonction de résultats quantiques pour converger vers la solution. Nous nous sommes plus spécifiquement intéressés à l'algorithme Variational Quantum Linear Solver (VQLS).

Les résultats que nous avons obtenus nous permettent de comparer ces méthodes quantiques avec les solutions numériques classiques.

2 Préliminaires de calcul quantique

Nous commençons par présenter les éléments fondamentaux permettant de maîtriser les bases de l'informatique quantique.

2.1 Postulats de la mécanique quantique

L'informatique quantique est construite autour des postulats de la mécanique quantique. Rappeler ces postulats permet du même coup d'introduire les briques de base du calcul quantique.

2.1.1 Postulat d'espace d'états : le qubit

À tout système quantique est associé un espace de Hilbert $\mathcal{H}$ appelé espace d'états. Un état de $\mathcal{H}$ est appelé un "ket" et représenté avec la notation $|\cdot\rangle$. Ces états se décomposent sur les états de base qui sont les qubits. Les bits 0 et 1 de l'informatique classique ont leurs analogues quantiques $|0\rangle$ et $|1\rangle$. On prendra ici $\mathcal{H} = \mathbb{C}^N$ avec $N = 2^n$, où n est le nombre de qubits.

Tout état se décompose sur les qubits de base. Ainsi, pour le système à un qubit ($\mathcal{H} = \mathbb{C}^2$), on a $|\psi\rangle = \alpha_\psi |0\rangle + \beta_\psi |1\rangle$ où $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Deux états $|\psi\rangle$ et $c|\psi\rangle$ pour $c \in \mathbb{C}^*$ représentent le même état. On supposera donc que tout état $|\psi\rangle$ est normalisé. On a donc

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle, \quad 0 \leq \theta \leq \pi, \quad 0 \leq \phi \leq 2\pi.$$

Ainsi, on peut identifier chaque état avec un unique point $\mathbf{a}$ de la sphère unité à trois dimensions (appelée

sphère de Bloch) :

$$\mathbf{a} = \begin{pmatrix} \sin(\theta) \cos(\phi) \\ \sin(\theta) \sin(\phi) \\ \cos(\theta) \end{pmatrix}.$$

2.1.2 Postulat d'opérateur quantique : la porte quantique

L'évolution entre deux états quantiques fait intervenir un opérateur unitaire $U \in \mathbb{C}^{N \times N}$:

$$|\phi'\rangle = U |\phi\rangle, \quad U^\dagger U = I_N.$$

On appellera porte quantique un tel opérateur unitaire.

Les matrices de Pauli sont des opérateurs unitaires de première importance. Avec l'identité, elles forment une base des opérateurs linéaires sur $\mathbb{C}^2$.

$$\sigma_x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Voici quelques autres portes quantiques couramment utilisées pour le système à un qubit :

- La porte de Hadamard

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

- La porte de phase

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

- La porte T

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$

Une évolution temporelle est décrite ainsi :

$$\psi(t_2) = U(t_2, t_1) \psi(t_1)$$

où $U(t_2, t_1)$ est unitaire $\forall t_1, t_2$. Pour un système évoluant sous l'influence d'un hamiltonien H indépendant du temps

$$i\partial_t |\psi(t)\rangle = H |\psi(t)\rangle,$$

l'opérateur unitaire correspondant est

$$U(t_2, t_1) = e^{-iH(t_2-t_1)} \quad \forall t_1, t_2.$$

Réciproquement, tout opérateur unitaire U peut s'exprimer sous la forme $U = e^{iH}$ où H est une matrice hermitienne.

2.1.3 Postulat de mesure quantique

Si M est la matrice hermitienne associée à l'observable que l'on mesure, sa décomposition spectrale est $M = \sum_m \lambda_m P_m$ où les λ_m sont les valeurs propres et les P_m les projecteurs sur les différents espaces propres.

Le résultat d'une mesure est toujours un point du spectre de M , avec probabilité

$$p_m = \langle \psi | P_m | \psi \rangle.$$

Après la mesure, l'état devient

$$|\psi\rangle \rightarrow \frac{P_m |\psi\rangle}{\sqrt{p_m}}.$$

L'espérance de la mesure vaut quant à elle

$$\mathbb{E}_\psi(M) = \langle \psi | M | \psi \rangle.$$

2.1.4 Postulat du produit tensoriel

Si un état quantique est constitué de m composantes d'espaces d'états respectifs $\{\mathcal{H}_i\}_{i=0}^{m-1}$, alors l'espace d'états correspondant est $\mathcal{H} = \bigotimes_{i=0}^{m-1} \mathcal{H}_i$.

Pour $|\psi_i\rangle$ des états quantiques des espaces $\mathcal{H}_i$, on a $|\psi\rangle = |\psi_0\rangle \otimes \dots \otimes |\psi_{m-1}\rangle \in \mathcal{H}$. Pour autant, tout état quantique de $\mathcal{H}$ ne peut s'exprimer sous cette forme factorisée.

L'espace d'états $\mathcal{H}$ d'un système à n qubits est $(\mathbb{C}^2)^{\otimes n} \equiv \mathbb{C}^{2^n}$. On utilise par ailleurs la notation $|01\rangle \equiv |0, 1\rangle \equiv |0\rangle |1\rangle \equiv |0\rangle \otimes |1\rangle$ ou plus généralement $|0^{\otimes n}\rangle \equiv |0\rangle^{\otimes n}$.

Prenons l'exemple du système à deux qubits. L'espace d'états est $\mathcal{H} = (\mathbb{C}^2)^{\otimes 2} \equiv \mathbb{C}^4$. La base canonique est

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

L'état de Bell n'est pas factorisable. Il s'écrit

$$\psi = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Un opérateur important sur les systèmes à deux qubits est la porte CNOT :

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

qui agit ainsi sur la base canonique :

$$\text{CNOT} \begin{cases} |00\rangle &= |00\rangle \\ |01\rangle &= |01\rangle \\ |10\rangle &= |11\rangle \\ |11\rangle &= |10\rangle \end{cases}.$$

On peut définir les opérateurs de Pauli multi-qubits : l'opérateur de Pauli qui agit sur le qubit i s'écrit X_i , Y_i ou Z_i . Par exemple, $X_i = I^{\otimes(i-1)} \otimes X \otimes I^{\otimes(n-i)}$.

2.2 Opérateur de densité

Si un système quantique peut être dans des états $|\psi_i\rangle$ de probabilités respectives p_i , on définit son opérateur de densité comme

$$\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|.$$

Notons que pour un état pur, $\rho = |\psi\rangle \langle \psi|$.

Lors d'une mesure, la probabilité $p(m)$ de retourner la valeur propre λ_m de l'observable M et l'espérance de la mesure sont

$$p(m) = \text{Tr}(P_m \rho), \quad \mathbb{E}_\rho(M) = \text{Tr}(M \rho).$$

Tous les postulats de la section précédente peuvent être énoncés en termes d'opérateurs de densité. En particulier, la transformation de $|\psi\rangle$ en $U|\psi\rangle$ se traduit par

$$\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i| \longrightarrow \sum_i p_i U |\psi_i\rangle \langle \psi_i| U^\dagger = U \rho U^\dagger.$$

2.3 Circuit quantique

Les algorithmes quantiques opèrent en général sur des systèmes multi-qubits. On utilise l'écriture visuelle des circuits quantiques.

Les états des qubits sont représentés par des lignes, sur lesquelles des portes quantiques peuvent être appliquées. Par exemple, le circuit ci-dessous permet d'obtenir le premier état de Bell $|\phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$:

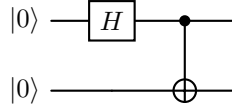


Figure 1: Circuit quantique pour $|\phi^+\rangle$

2.4 Opérations de copie et théorème de non-clonage

Le théorème de non-clonage énonce qu'il n'existe pas d'opérateur unitaire U tel que $\forall |x\rangle, |s\rangle$,

$$U |x\rangle \otimes |s\rangle = |x\rangle \otimes |x\rangle.$$

Autrement dit, il n'est pas possible d'effectuer de recopie d'un état quelconque $|x\rangle$. Ce résultat rend presque impossible la réalisation de tout algorithme itératif, utilisant des variables intermédiaires qui sont des copies des variables précédentes.

Notons que malgré ce théorème, on sait comment un état quantique est préparé : $|x\rangle = U_x |s\rangle$ avec U_x et $|s\rangle$ connus. On peut donc réaliser la copie de cet état à l'aide de l'opération $(I \otimes U_x) |x\rangle \otimes |s\rangle = |x\rangle \otimes |x\rangle$.

De plus, on peut quand même réaliser la copie d'une information classique. En effet, la porte CNOT permet, pour $x \in \{0, 1\}$, de réaliser l'opération $\text{CNOT} |x, 0\rangle = |x, x\rangle$. Cependant en vertu du théorème de non-clonage, la porte CNOT est incapable de recopier une information du type $|x\rangle = a|0\rangle + b|1\rangle$. Il est aussi possible de copier de l'information classique contenue dans plusieurs qubits grâce à une porte CNOT multi-qubit.

Une conséquence du théorème de non-clonage quantique est l'inexistence d'un opérateur unitaire U de suppression, ce qui reviendrait à copier l'état $|0\rangle$.

2.5 Mesure

Le principe de mesure différée affirme que les opérations de mesure peuvent toujours être déplacées d'un état intermédiaire du circuit à l'état final, sans changer la mesure. Attention : lorsque l'on utilise la mesure différée, il est nécessaire de stocker l'information intermédiaire dans des qubits supplémentaires.

Le principe de mesure implicite affirme que le résultat de la mesure d'un ou plusieurs qubits ne dépend pas de la mesure des autres qubits du système. Plus spécifiquement, l'opérateur de densité d'un sous-système n'est pas modifié par la mesure de l'autre sous-système.

2.6 Accroissement linéaire de l'erreur

Considérons un algorithme quantique, représenté par un opérateur unitaire U . En supposant que l'on peut décomposer $U = U_K \dots U_1$ avec les U_i unitaires, et que chaque U_i peut être approché par $\tilde{U}_i$, alors

$$\|U_i - \tilde{U}_i\| \leq \epsilon \quad \forall i \in \llbracket 1, k \rrbracket \implies \|U_K \dots U_1 - \tilde{U}_K \dots \tilde{U}_1\| \leq K\epsilon.$$

Cela garantit un certain contrôle sur les erreurs commises par les algorithmes quantiques utilisés dans la suite.

2.7 Ensembles universels de portes et calcul réversible

Un ensemble de portes est universel lorsque toute porte peut être représentée à l'aide d'une combinaison de portes de cet ensemble.

Tout opérateur unitaire multi-qubits peut être représenté en utilisant des portes à un ou deux qubits. Afin de s'autoriser à manipuler des ensembles universels de portes discrètes, on s'affranchit de l'exactitude de la représentation. On dit donc qu'un ensemble de portes quantiques $\mathcal{S}$ est universel si pour tout opérateur unitaire U et pour toute précision ϵ , on peut trouver des portes $U_1, \dots, U_m \in \mathcal{S}$ telles que

$$\|U - U_m \dots U_1\| \leq \epsilon.$$

Par exemple, l'ensemble $\{H, T, \text{CNOT}\}$ est universel.

Toute porte classique $x \mapsto f(x)$ peut être rendue réversible en la transformant en $(x, y) \mapsto (x, y \oplus f(x))$, car alors $(x, 0) \mapsto (x, f(x))$. Ainsi en stockant toutes les variables intermédiaires, le calcul est rendu réversible. Toutefois, utiliser cette idée en calcul quantique présente deux inconvénients : l'utilisation de grandes ressources quantiques pour le stockage et l'intrication entre les variables intermédiaires et les états quantiques d'intérêt. Un procédé appelé "décalcul" permet de résoudre ces deux problèmes.

Pour implémenter une fonction booléenne $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, on suppose qu'il existe un oracle

$$U_f : |0^m\rangle |x\rangle \mapsto |f(x)\rangle |x\rangle.$$

Il prend en réalité une forme plus complexe qui fait intervenir des registres de travail supplémentaires.

Finalement, tout processus classique peut être implémenté efficacement sur un ordinateur quantique, en le changeant en processus réversible puis en utilisant la technique du décalcul. On obtient

$$U_f : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle.$$

2.8 Représentation des nombres en virgule fixe et opérations arithmétiques classiques

La représentation binaire du nombre $k \in \llbracket 1, N \rrbracket$, où $N = 2^n$, est

$$k = \sum_{i=0}^{n-1} k_i 2^i := k_{n-1} \dots k_0.$$

La représentation du nombre k divisé par 2^m , où $0 \leq m \leq n$, est

$$\frac{k}{2^m} = \sum_{i=0}^{n-1} k_i 2^{i-m} := k_{n-1} \dots k_m . k_{m-1} \dots k_0.$$

3 Outils pour la résolution de systèmes linéaires

Nous présentons dans cette section différents outils qui seront utilisés dans les algorithmes que nous mettons en avant pour résoudre l'équation de Poisson.

3.1 Test de Hadamard

Le test de Hadamard est un moyen d'estimer $\langle \psi | V | \psi \rangle$. V est supposé unitaire mais pas nécessairement hermitien ; on doit donc mesurer séparément la partie réelle et la partie imaginaire de $\langle \psi | V | \psi \rangle$.

Le circuit pour la partie réelle est le suivant :

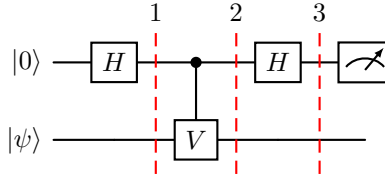


Figure 2: Circuit quantique pour le test de Hadamard réel

Suivons le déroulement du circuit pas-à-pas.

Porte de Hadamard

$$|\psi_1\rangle = \frac{|0\rangle \otimes |\psi\rangle}{\sqrt{2}} + \frac{|1\rangle \otimes |\psi\rangle}{\sqrt{2}}.$$

Circuit principal

$$|\psi_2\rangle = \frac{|0\rangle \otimes |\psi\rangle}{\sqrt{2}} + \frac{|1\rangle \otimes V|\psi\rangle}{\sqrt{2}}.$$

Seconde porte de Hadamard

$$\begin{aligned} |\psi_3\rangle &= \frac{|0\rangle \otimes |\psi\rangle}{2} + \frac{|1\rangle \otimes |\psi\rangle}{2} + \frac{|0\rangle \otimes V|\psi\rangle}{2} - \frac{|1\rangle \otimes V|\psi\rangle}{2} \\ &= \frac{|0\rangle \otimes (I + V)|\psi\rangle}{2} + \frac{|1\rangle \otimes (I - V)|\psi\rangle}{2}. \end{aligned}$$

La probabilité p_0 de récupérer l'état $|0\rangle$ en sortie de circuit vaut

$$\begin{aligned} p_0 &= \frac{1}{4} \langle \psi | (I + V^\dagger)(I + V) | \psi \rangle \\ &= \frac{1}{4} \langle \psi | 2I + V + V^\dagger | \psi \rangle \\ &= \frac{1}{4} (\langle \psi | 2I | \psi \rangle + \langle \psi | V | \psi \rangle + \langle \psi | V^\dagger | \psi \rangle) \\ &= \frac{1 + \text{Re}(\langle \psi | V | \psi \rangle)}{2}. \end{aligned}$$

Le circuit pour la partie imaginaire utilise la porte de phase $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$:

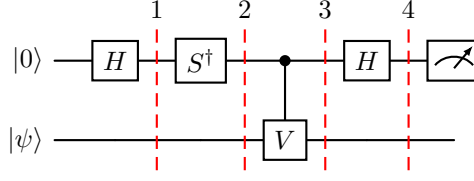


Figure 3: Circuit quantique pour le test de Hadamard imaginaire

On peut là encore suivre le déroulement du circuit :

Porte de Hadamard

$$|\psi_1\rangle = \frac{|0\rangle \otimes |\psi\rangle}{\sqrt{2}} + \frac{|1\rangle \otimes |\psi\rangle}{\sqrt{2}}.$$

Porte S

$$|\psi_2\rangle = \frac{|0\rangle \otimes |\psi\rangle}{\sqrt{2}} - \frac{i|1\rangle \otimes |\psi\rangle}{\sqrt{2}}.$$

Circuit principal

$$|\psi_3\rangle = \frac{|0\rangle \otimes |\psi\rangle}{\sqrt{2}} - \frac{i|1\rangle \otimes V|\psi\rangle}{\sqrt{2}}.$$

Seconde porte de Hadamard

$$\begin{aligned} |\psi_3\rangle &= \frac{|0\rangle \otimes |\psi\rangle}{2} + \frac{|1\rangle \otimes |\psi\rangle}{2} - \frac{i|0\rangle \otimes V|\psi\rangle}{2} + \frac{i|1\rangle \otimes V|\psi\rangle}{2} \\ &= \frac{|0\rangle \otimes (I - iV)|\psi\rangle}{2} + \frac{|1\rangle \otimes (I + iV)|\psi\rangle}{2}. \end{aligned}$$

La probabilité p'_0 de récupérer l'état $|0\rangle$ en sortie de circuit vaut

$$\begin{aligned} p'_0 &= \frac{1}{4} \langle \psi | (I + iV^\dagger)(I - iV) | \psi \rangle \\ &= \frac{1}{4} \langle \psi | 2I - iV + iV^\dagger | \psi \rangle \\ &= \frac{1}{4} (\langle \psi | 2I | \psi \rangle - i \langle \psi | V | \psi \rangle + i \langle \psi | V^\dagger | \psi \rangle) \\ &= \frac{1 + \text{Im}(\langle \psi | V | \psi \rangle)}{2}. \end{aligned}$$

En combinant les résultats des deux circuits, on obtient $\langle \psi | V | \psi \rangle$:

$$\boxed{\langle \psi | V | \psi \rangle = 2p_0 - 1 + i(2p'_0 - 1)}.$$

3.2 Swap test

Le *swap test* est une application du test de Hadamard, permettant d'évaluer l'*overlap* $|\langle \phi | \psi \rangle|$ de deux états quantiques $|\phi\rangle$ et $|\psi\rangle$. Voici le circuit quantique permettant de le réaliser :

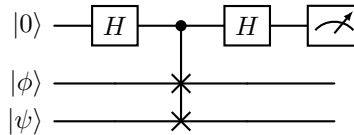


Figure 4: Circuit quantique pour le *swap test*

On remarque qu'il s'agit d'un test de Hadamard réel avec comme circuit principal V une porte *swap*. Or, $\langle \phi \psi | \text{SWAP} | \phi \psi \rangle = \langle \phi \psi | \psi \phi \rangle$. On a donc immédiatement

$$p_0 = \frac{1 + \text{Re}(\langle \phi \psi | \psi \phi \rangle)}{2} = \frac{1 + |\langle \phi | \psi \rangle|^2}{2}$$

d'où finalement

$$|\langle \phi | \psi \rangle| = \sqrt{2p_0 - 1}.$$

3.3 Transformée de Fourier quantique

La transformée de Fourier quantique est une opération mathématique agissant sur les qubits que l'on retrouve dans de nombreux algorithmes, notamment dans l'algorithme HHL.

La transformée de Fourier quantique d'un vecteur $|j\rangle$ est définie par

$$U_{\text{FT}} |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i2\pi \frac{kj}{N}} |k\rangle.$$

En particulier,

$$U_{\text{FT}} |0^{\otimes n}\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle = H^{\otimes n} |0^{\otimes n}\rangle.$$

La transformée de Fourier quantique inverse de $|j\rangle$ est quant à elle définie par

$$U_{\text{FT}}^\dagger |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-i2\pi \frac{kj}{N}} |k\rangle.$$

En utilisant la représentation binaire des entiers j et k , on peut écrire

$$e^{i2\pi \frac{kj}{N}} = e^{i2\pi k_0(\cdot j_{n-1} \dots j_0)} e^{i2\pi k_1(\cdot j_{n-2} \dots j_0)} \dots e^{i2\pi k_{n-1}(\cdot j_0)}.$$

On obtient alors facilement que

$$U_{\text{FT}} |j_{n-1} \dots j_0\rangle = \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{i2\pi(\cdot j_0)} |1\rangle \right) \otimes \left(|0\rangle + e^{i2\pi(\cdot j_1 j_0)} |1\rangle \right) \otimes \dots \otimes \left(|0\rangle + e^{i2\pi(\cdot j_{n-1} \dots j_0)} |1\rangle \right).$$

L'implémentation de la transformée de Fourier quantique est proposée en annexe 8.1.

3.4 Estimation de phase quantique

Soit U un opérateur unitaire et $|\psi\rangle$ un de ses vecteurs propres. Les valeurs propres de U étant nécessairement de module 1, on a donc

$$U |\psi\rangle = e^{i2\pi\phi} |\psi\rangle, \quad \phi \in [0, 1[.$$

Le but est d'estimer ϕ avec une certaine précision.

Avec un ordinateur classique, on peut estimer ϕ par le moyen suivant :

$$\frac{\langle j | U | \psi \rangle}{\langle j | \psi \rangle} = e^{i2\pi\phi}.$$

Mais ce procédé ne peut pas être efficacement implémenté sur un ordinateur quantique.

3.4.1 Avec le test de Hadamard

Le test de Hadamard permet de réaliser l'estimation de la phase d'un unique qubit. La probabilité de mesurer le qubit $|0\rangle$ dans l'état $|1\rangle$ est

$$p(1) = \frac{1}{2}(1 - \text{Re} \langle \psi | U \psi \rangle) = \frac{1}{2}(1 - \cos(2\pi\phi))$$

d'où

$$\phi = \pm \frac{\arccos(1 - 2p(1))}{2\pi} \mod 1.$$

Étant donné que le circuit du test de Hadamard pour la partie réelle ne permet pas d'accéder au signe de ϕ , on peut utiliser le circuit pour la partie imaginaire en remplaçant $S^\dagger$ par S . Ainsi, on a

$$p(1) = \frac{1}{2}(1 + \sin(2\pi\phi))$$

ce qui donne

$$\begin{cases} \phi \in \left[0, \frac{1}{2}\right[, & p(1) \geq \frac{1}{2} \\ \phi \in \left[\frac{1}{2}, 1\right[, & p(1) < \frac{1}{2} \end{cases}$$

3.4.2 Avec la transformée de Fourier quantique

On écrit la phase $\phi = .\phi_{d-1} \dots \phi_0$ et on définit l'opérateur unitaire

$$\mathcal{U} = \sum_{j=0}^{2^d-1} |j\rangle \langle j| \otimes U^j.$$

À noter que l'on a $U^j = U^{\sum_{i=0}^{d-1} j_i 2^i} = \prod_{i=0}^{d-1} U^{j_i 2^i}$.

En utilisant la transformée de Fourier quantique et $\mathcal{U}$, on peut mesurer ϕ . On procède ainsi :

$$\begin{aligned} |0^d\rangle |\psi_0\rangle &\xrightarrow{U_{\text{FT}} \otimes I} \frac{1}{\sqrt{2^d}} \sum_{j=0}^{2^d-1} |j\rangle |\psi_0\rangle \\ &\xrightarrow{\mathcal{U}} \frac{1}{\sqrt{2^d}} \sum_{j=0}^{2^d-1} |j\rangle U^j |\psi_0\rangle = \frac{1}{\sqrt{2^d}} \sum_{j=0}^{2^d-1} |j\rangle e^{i2\pi\phi j} |\psi_0\rangle \\ &\xrightarrow{U_{\text{FT}}^\dagger \otimes I} \sum_{k'=0}^{2^d-1} \left(\frac{1}{2^d} \sum_{j=0}^{2^d-1} e^{i2\pi j \left(\phi - \frac{k'}{2^d} \right)} \right) |k'\rangle |\psi_0\rangle. \end{aligned}$$

Or, $\exists k \in \llbracket 0, 2^d - 1 \rrbracket \mid \phi = \frac{k}{2^d}$. On peut donc, en mesurant les qubits, obtenir l'information sur la phase. On obtient plus précisément l'état

$$|k\rangle |\psi_0\rangle = 2^d |\phi\rangle |\psi_0\rangle,$$

les autres termes s'annulant par interférences destructives.

4 Algorithme Harrow-Hassidim-Lloyd

L'algorithme HHL permet la résolution de systèmes linéaires sur un ordinateur quantique. Plus précisément, on souhaite résoudre l'équation

$$Ax = b$$

où $A \in \mathcal{M}_{N_b}(\mathbb{C})$ et $x, b \in \mathbb{C}^{N_b}$. On suppose qu'il existe $n_b \in \mathbb{N}$ tel que $N_b = 2^{n_b}$. N_b est le nombre d'inconnues et n_b est le nombre de qubits utilisé pour résoudre le système.

On suppose dans la suite que A est hermitienne. Si tel n'est pas le cas, il suffit de considérer $\tilde{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$.

La résolution se fera au moyen du circuit quantique suivant, auquel nous nous référerons tout au long du processus :

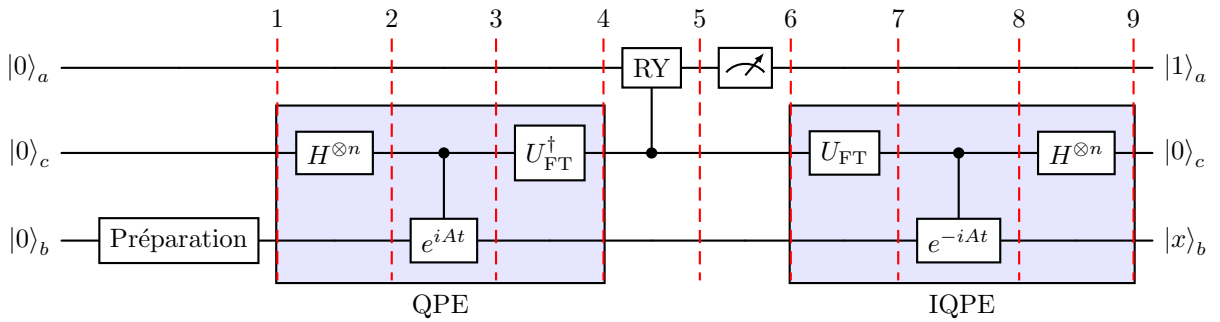


Figure 5: Circuit quantique pour l'algorithme HHL

L'algorithme HHL se divise en quatre phases principales :

1. La préparation de l'état ;
2. L'estimation de phase quantique (*quantum phase estimation*, QPE dans la suite) ;
3. La rotation du registre auxiliaire et la mesure;
4. L'estimation de phase quantique inverse (*inverse quantum phase estimation*, IQPE dans la suite).

Les significations des différentes portes seront précisées par la suite.

On remarque que ce circuit fait appel à trois registres distincts :

- Le registre b , de taille n_b , stocke $|b\rangle$ au-début du circuit et sera progressivement modifié pour finalement accueillir $|x\rangle$ en sortie de circuit.
- Le registre c , de taille n , stocke les valeurs des phases des valeurs propres de A après la QPE. Il fait référence à l'horloge (*clock*) dans la phase de rotation contrôlée. On pose $N = 2^n$.
- Le registre a , pour *ancilla*, est un registre auxiliaire dont le rôle sera précisé plus tard. Il ne contient qu'un qubit.

A étant hermitienne, elle possède une décomposition spectrale sous la forme

$$A = \sum_{j=0}^{2^{n_b}-1} \lambda_j |u_j\rangle \langle u_j|$$

où les λ_j sont les valeurs propres et les $|u_j\rangle$ les vecteurs propres de A . $|b\rangle$ peut être décomposé dans la base propre de A :

$$|b\rangle = \sum_{j=0}^{2^{n_b}-1} b_j |u_j\rangle.$$

Le problème que l'on cherche à résoudre

$$A|x\rangle = |b\rangle \tag{E}$$

admet pour solution

$$|x\rangle = A^{-1} |b\rangle = \sum_{j=0}^{2^{n_b}-1} \frac{b_j}{\lambda_j} |u_j\rangle. \tag{1}$$

Notre but sera donc d'approcher cette solution à l'aide du circuit quantique présenté précédemment. Il est important de noter que la solution obtenue ne sera pas égale à la solution analytique, mais proportionnelle à celle-ci.

Le détail du calcul de l'algorithme HHL est présenté dans l'annexe 8.3.

5 Algorithmes variationnels

5.1 Principe général

Un algorithme variationnel est un algorithme hybride, qui repose à la fois sur l'utilisation d'un ordinateur classique et d'un circuit quantique. Plus précisément, l'utilisation d'un algorithme variationnel repose sur l'implémentation de circuits quantiques paramétrés, dont les paramètres sont optimisés classiquement à chaque itération de l'algorithme, ce qui permet de converger vers la solution.

Le circuit quantique comporte une première partie qui construit un état initial $|\psi(\theta)\rangle$, appelé *ansatz*, qui dépend de $\theta \in \mathbb{R}^M$. Cet état est créé à partir d'un état de référence $|\psi_{\text{ref}}\rangle$ par un unitaire U qui se décompose de la manière suivante :

$$U(\theta) = U_1(\theta_1) \dots U_M(\theta_M)$$

où les U_i sont des unitaires représentant des portes quantiques à un ou deux qubits.

Puis la deuxième partie du circuit quantique évalue une fonction coût, qui dépend de θ . Vient ensuite une optimisation classique de cette fonction coût, par exemple par descente de gradient, qui conduit à la modification du paramètre θ , avant de rentrer à nouveau dans le circuit quantique.

Finalement, la routine de l'algorithme variationnel peut être décomposée de la manière suivante :

1. Préparation de l'état de référence ψ_{ref} .
2. Préparation de l'ansatz $|\psi(\theta)\rangle = U(\theta) |\psi_{\text{ref}}\rangle$.
3. Mesure de la fonction coût.
4. Optimisation des paramètres de θ .

On répète ces étapes jusqu'à atteindre le critère d'arrêt choisi.

La forme générale du circuit quantique d'un algorithme variationnel est la suivante :

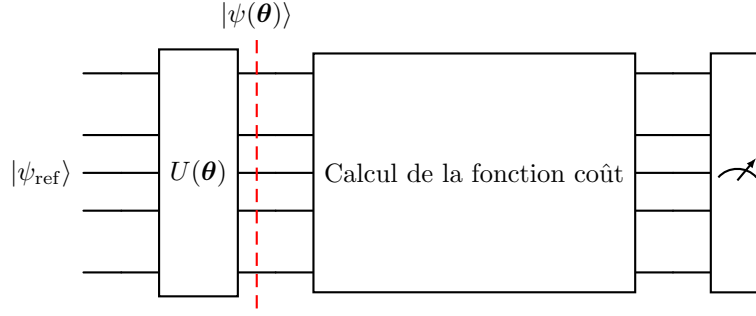


Figure 6: Circuit quantique pour un algorithme variationnel

Les différentes techniques permettant de préparer l'ansatz sont décrites dans l'annexe 8.4.

5.2 Variational Quantum Eigensolver

L'algorithme Variational Quantum Eigensolver (VQE) est un algorithme variationnel dont le but est de déterminer les valeurs propres et vecteurs propres associés d'un certain hamiltonien $\hat{H}$, c'est-à-dire de résoudre le problème aux valeurs propres suivant :

$$\hat{H} |\psi\rangle = E |\psi\rangle$$

où $\hat{H}$ est l'hamiltonien du système $\mathcal{H}$ et $|\psi\rangle$ l'état propre associé à l'énergie E .

Dans sa formulation la plus basique, VQE nous donne une approximation de l'énergie de l'état fondamental et l'état correspondant. Il est basé sur le théorème d'énergie minimale et repose sur le principe variationnel de Rayleigh-Ritz :

$$\forall |\psi\rangle \in \mathcal{H}, \quad \langle \psi | \hat{H} | \psi \rangle \geq E_0$$

où E_0 est l'énergie de l'état fondamental. L'idée est d'approcher de manière itérative le $|\psi\rangle$ solution en minimisant la fonction du coût du système

$$E : \theta \in \mathbb{R}^n \mapsto \langle \psi(\theta) | \hat{H} | \psi(\theta) \rangle$$

$|\psi(\theta)\rangle$ est l'état ansatz du système. Il est préparé à l'aide du circuit ansatz $U(\theta)$. On cherche donc à résoudre

$$\text{Min}_{\theta} E(\theta) = \langle \psi(\theta) | \hat{H} | \psi(\theta) \rangle$$

Le θ^* solution permet alors d'approcher la valeur propre E_0 solution du problème par $E_0 \approx \langle \psi(\theta^*) | \hat{H} | \psi(\theta^*) \rangle$. On aura alors de plus $|\psi(\theta^*)\rangle \approx |\psi_0\rangle$, le vecteur propre associé à E_0 . L'algorithme permet donc d'obtenir l'énergie fondamentale et l'état associé.

Il s'agit alors de fabriquer un circuit quantique qui permet, à chaque itération k , de calculer θ_k et le coût $E(\theta_k)$ associé. Ces valeurs sont stockées dans un ordinateur classique afin qu'il minimise (à l'aide par exemple d'une descente de gradient) la fonction coût.

Étant donné que $\hat{H}$ est un opérateur hermitien, mais n'est pas nécessairement unitaire, on ne peut pas directement l'intégrer au circuit en tant que porte quantique. On utilise donc une décomposition du hamiltonien en chaîne de Pauli. En effet, les opérateurs de Pauli I , X , Y et Z forment une base des matrices hermitiennes de $\mathbb{C}^2 \times \mathbb{C}^2$. Ainsi, les produits $\{I, X, Y, Z\}^{\otimes N}$ forment une base des observables à N qubits. On rappelle

$$X = \sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \sigma_Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Une chaîne de Pauli est de la forme $P_1 \otimes P_2 \otimes \dots \otimes P_N$ où $P_i \in \{I, X, Y, Z\}$. On peut donc écrire le hamiltonien sous la forme d'une combinaison linéaire d'unitaires :

$$\hat{H} = \sum_{\alpha=1}^L h_{\alpha} P_{\alpha}$$

avec $(P_{\alpha})_{\alpha}$ des chaînes de Pauli et $(h_{\alpha})_{\alpha}$ des réels. On peut alors réécrire la fonction coût :

$$E(\theta) = \sum_{\alpha} h_{\alpha} \langle \psi(\theta) | P_{\alpha} | \psi(\theta) \rangle.$$

À chaque itération k , c'est-à-dire à chaque passage dans le circuit :

$$E(\boldsymbol{\theta}_k) = \sum_{\alpha} h_{\alpha} \langle \psi(\boldsymbol{\theta}_k) | P_{\alpha} | \psi(\boldsymbol{\theta}_k) \rangle.$$

On va mesurer cette grandeur à chaque itération.

Pour minimiser l'énergie et trouver $\boldsymbol{\theta}_{k+1}$ à partir de $\boldsymbol{\theta}_k$, on réalise une optimisation sur un ordinateur classique. On pourra par exemple utiliser une descente de gradient, et on aura dans ce cas

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \rho_k \nabla E(\boldsymbol{\theta}_k)$$

où $\rho_k > 0$.

En résumé, on commence avec un état quantique de référence $|\psi_{\text{ref}}\rangle$. On applique à cet état une transformation classique à l'aide de la porte multi-qubits $U(\boldsymbol{\theta})$. Puis, on mesure avec un circuit quantique la fonction coût énergie $E(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | \hat{H} | \psi(\boldsymbol{\theta}) \rangle$, en utilisant la décomposition de l'hamiltonien en chaînes de Pauli. On optimise $\boldsymbol{\theta}$ à l'aide par exemple d'une descente de gradient. Enfin, une fois le minimum local atteint, on obtient l'énergie fondamentale et l'état associé.

5.3 Variationnal Quantum Linear Solver

5.3.1 Hamiltonien et fonction coût associés

On s'intéresse maintenant au problème matriciel

$$A\mathbf{x} = \mathbf{b}.$$

On peut utiliser VQE pour résoudre ce système linéaire. Il faut donc trouver un hamiltonien $\hat{H}$ permettant de réécrire le problème sous la forme

$$\hat{H} |\psi(\boldsymbol{\theta})\rangle = E |\psi(\boldsymbol{\theta})\rangle$$

On a besoin :

- d'un hamiltonien ;
- d'une fonction coût ;
- d'un unitaire qui prépare $|b\rangle$.

On suppose disposer d'un unitaire U_b permettant de préparer $|b\rangle$:

$$U_b |0\rangle^{\otimes n} = |b\rangle.$$

On fait l'hypothèse que A est hermitienne. Elle s'écrit donc comme une combinaison linéaire d'unitaires :

$$A = \sum_{l=1}^L c_l A_l$$

où A_l est une chaîne de Pauli et $c_l \in \mathbb{C}$. On suppose que l'on connaît aussi l'unitaire $U(\boldsymbol{\theta})$ qui prépare $|x(\boldsymbol{\theta})\rangle$ par $U(\boldsymbol{\theta}) |0\rangle^{\otimes n} = |x(\boldsymbol{\theta})\rangle$.

On pose $|\psi(\boldsymbol{\theta})\rangle = A |x(\boldsymbol{\theta})\rangle$. On a donc $|\psi(\boldsymbol{\theta})\rangle = A |x(\boldsymbol{\theta})\rangle = AU(\boldsymbol{\theta}) |0\rangle^{\otimes n}$. On définit le hamiltonien

$$\hat{H} = A^\dagger (I - |b\rangle \langle b|) A$$

On peut vérifier que $\hat{H}$ est bien hermitien. De plus, il a pour état fondamental la solution $|x^*\rangle$ avec une énergie propre associée nulle. Il permet de définir la fonction coût

$$C : \boldsymbol{\theta} \mapsto \langle x(\boldsymbol{\theta}) | \hat{H} | x(\boldsymbol{\theta}) \rangle.$$

Or,

$$\begin{aligned} C(\boldsymbol{\theta}) &= \langle x(\boldsymbol{\theta}) | \hat{H} | x(\boldsymbol{\theta}) \rangle \\ &= \langle x(\boldsymbol{\theta}) | A^\dagger (I - |b\rangle \langle b|) A | x(\boldsymbol{\theta}) \rangle \\ &= \langle x(\boldsymbol{\theta}) | A^\dagger A | x(\boldsymbol{\theta}) \rangle - |\langle b | A | x(\boldsymbol{\theta}) \rangle|^2 \\ &= \langle \psi(\boldsymbol{\theta}) | \psi(\boldsymbol{\theta}) \rangle - |\langle b | \psi(\boldsymbol{\theta}) \rangle|^2 \end{aligned}$$

Pour la mesurer, deux quantités sont donc nécessaires :

- L'overlap $|\langle b|\psi(\theta)\rangle|^2$;
- $\langle\psi(\theta)|\psi(\theta)\rangle$.

En réalité, on mesure plutôt une fonction coût normalisée

$$\mathcal{C}(\theta) := \frac{C(\theta)}{\langle\psi(\theta)|\psi(\theta)\rangle} = 1 - \frac{|\langle b|\psi(\theta)\rangle|^2}{\langle\psi(\theta)|\psi(\theta)\rangle}$$

mais la mesure des deux quantités mentionnées précédemment reste nécessaire.

5.3.2 Calcul de $\langle\psi(\theta)|\psi(\theta)\rangle$

On dispose essentiellement de deux possibilités pour le calcul de cette quantité :

Décomposition en chaîne de Pauli La décomposition en chaîne de Pauli de A permet d'écrire

$$|\psi(\theta)\rangle = A|x(\theta)\rangle = \sum_{l=1}^L c_l A_l |x(\theta)\rangle$$

On en déduit que

$$\langle\psi(\theta)|\psi(\theta)\rangle = \sum_{l=1}^L \sum_{p=1}^L c_l c_p^\dagger \langle x(\theta)|A_p^\dagger A_l|x(\theta)\rangle$$

Or, on prépare $x(\theta)$ avec $U(\theta)|0^{\otimes n}\rangle = |x(\theta)\rangle$. Ainsi,

$$\langle\psi(\theta)|\psi(\theta)\rangle = \sum_{l=1}^L \sum_{p=1}^L c_l c_p^\dagger \langle 0^{\otimes n}|U^\dagger(\theta)A_p^\dagger A_l U(\theta)|0^{\otimes n}\rangle$$

On pose $\omega_{lp}(\theta) = \langle 0^{\otimes n}|U^\dagger(\theta)A_p^\dagger A_l U(\theta)|0^{\otimes n}\rangle$. Les c_l et A_l étant supposé connus, l'accès à $\omega_{lp}(\theta)$ permet de récupérer la valeur de $\langle\psi(\theta)|\psi(\theta)\rangle$ associée. On utilise pour cela un test de Hadamard.

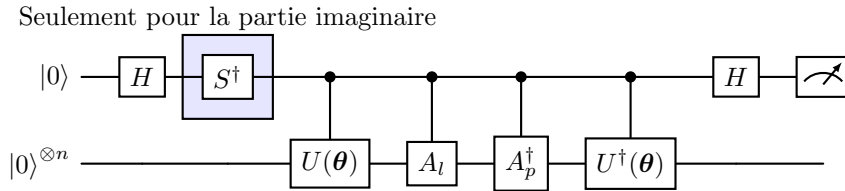


Figure 7: Circuit quantique pour la mesure de $\langle\psi(\theta)|\psi(\theta)\rangle$

Le circuit ci-dessus est un test de Hadamard avec $V = U^\dagger(\theta)A_p^\dagger A_l U(\theta)$.

La probabilité de mesurer 0 pour l'ancilla est une fonction de ω_{lp} . En particulier :

- La probabilité de mesurer $|0\rangle$ en sortie du circuit sans la porte $S^\dagger$ est

$$p_0 = \frac{1 + \text{Re}(\omega_{lp})}{2}.$$

- La probabilité de mesurer $|0\rangle$ en sortie du circuit avec la porte $S^\dagger$ est

$$p_0 = \frac{1 + \text{Im}(\omega_{lp})}{2}.$$

D'où

$$\omega_{lp} = 2p_0 - 1 + i(2p'_0 - 1).$$

Grâce à cette expression, on peut déterminer $\langle\psi(\theta)|\psi(\theta)\rangle$. Notons que $2L^2$ mesures sont nécessaires, et ce à chaque itération de l'algorithme variationnel.

Utilisation d'une observable Dans le cas où A est symétrique réelle, une autre possibilité s'offre à nous. En effet, on peut aussi voir $\langle\psi(\theta)|\psi(\theta)\rangle$ comme la quantité $\langle x(\theta)|A^\dagger A|x(\theta)\rangle$. Étant donné que l'on suppose A symétrique réelle, on a

$$\langle\psi(\theta)|\psi(\theta)\rangle = \langle x(\theta)|A^2|x(\theta)\rangle.$$

Ainsi, le calcul de $\langle\psi(\theta)|\psi(\theta)\rangle$ revient à l'évaluation de la valeur moyenne de l'observable A^2 dans l'état $x(\theta)$.

Dans la pratique, la décomposition de A^2 en chaîne de Pauli est nécessaire pour mener ce calcul à bien.

5.3.3 Calcul de $|\langle b|\psi(\theta)\rangle|^2$

On dispose là encore de deux possibilités pour calculer $|\langle b|\psi(\theta)\rangle|^2$. Éliminons d'emblée la possibilité du *swap test*, qui pourrait sembler convenir étant donné que ce type de circuit permet de mesurer un *overlap*. Cependant, le circuit suivant serait nécessaire :

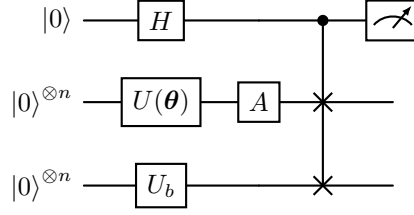


Figure 8: Circuit quantique inutilisable pour la mesure de $|\langle b|\psi(\theta)\rangle|^2$

En effet, $\psi(\theta) = AU(\theta)|0^{\otimes n}\rangle$. Or, A n'est pas supposée unitaire mais seulement hermitienne. Elle ne peut donc pas jouer le rôle de porte quantique en général, et un tel circuit est impossible à réaliser. On pourrait vouloir décomposer A en chaîne de Pauli et appliquer le circuit avec les A_l à la place de A , mais la non-linéarité de la quantité $|\langle b|\psi(\theta)\rangle|^2$ empêche ces manipulations. Le *swap test* n'est donc pas une solution adaptée. Heureusement, deux autres solutions sont possibles.

Décomposition en chaîne de Pauli Comme précédemment, la décomposition de A en chaîne de Pauli permet d'utiliser des tests de Hadamard pour aboutir au résultat :

$$\begin{aligned} |\langle b|\psi(\theta)\rangle|^2 &= \langle b|\psi(\theta)\rangle \langle \psi(\theta)|b\rangle \\ &= \sum_{l=1}^L \sum_{p=1}^L c_l c_p^* \underbrace{\langle 0^{\otimes n}|U_b^\dagger A_l U(\theta)|0^{\otimes n}\rangle \langle 0^{\otimes n}|U(\theta)^\dagger A_p^\dagger U_b|0^{\otimes n}\rangle}_{\gamma_{lp}} \end{aligned}$$

On définit alors

$$\Gamma_l(\theta) := U_b^\dagger A_l U(\theta)$$

et donc

$$\Gamma_l^\dagger(\theta) = U(\theta)^\dagger A_l U_b.$$

Γ_l est unitaire. Pour mesurer γ_{lp} , on utilise un test de Hadamard :

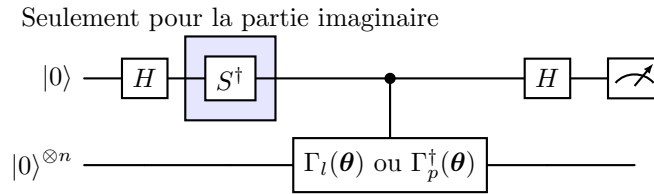


Figure 9: Circuit quantique pour la mesure de $|\langle b|\psi(\theta)\rangle|^2$

Ainsi pour obtenir γ_{lp} , il faut effectuer quatre mesures : deux pour le terme $\langle 0^{\otimes n}|\Gamma_l(\theta)|0^{\otimes n}\rangle$ et deux pour le terme $\langle 0^{\otimes n}|\Gamma_p^\dagger(\theta)|0^{\otimes n}\rangle$. Cela donne $4L^2$ mesures à chaque itération de l'algorithme variationnel.

Notons que pour les termes où $p = l$, il est possible d'utiliser un *swap test* afin de simplifier les calculs. En effet,

$$\begin{aligned} c_l c_p^* \langle 0^{\otimes n}|U_b^\dagger A_l U(\theta)|0^{\otimes n}\rangle \langle 0^{\otimes n}|U(\theta)^\dagger A_p^\dagger U_b|0^{\otimes n}\rangle &= c_p c_p^* \langle b|A_p x(\theta)\rangle \langle x(\theta)A_p^\dagger|b\rangle \\ &= |c_p|^2 |\langle b|A_p x(\theta)\rangle|^2 \end{aligned}$$

Le circuit pertinent est alors le suivant :

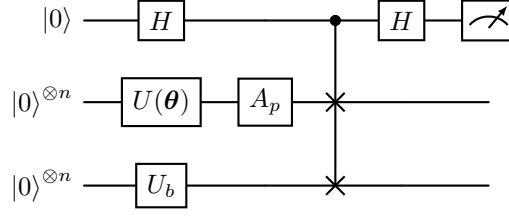


Figure 10: Circuit quantique pour la mesure de γ_{pp}

Utilisation d'une observable Dans le cas où A est symétrique réelle, on peut simplifier le calcul de $|\langle b|\psi(\theta)\rangle|^2$. Notons que $|\langle b|\psi(\theta)\rangle|^2 = |\langle b|A|x(\theta)\rangle|^2$. On peut alors exprimer cette quantité comme le carré de la valeur moyenne de l'observable $X \otimes A$ dans l'état $|bx(\theta)\rangle := \frac{|0\rangle|b\rangle + |1\rangle|x(\theta)\rangle}{\sqrt{2}}$:

$$\begin{aligned}
\langle bx(\theta)|X \otimes A|bx(\theta)\rangle &= \frac{1}{2} \langle (\langle 0| \langle b| + \langle 1| \langle x(\theta)|) | X \otimes A | (\langle 0| |b\rangle + |1\rangle |x(\theta)\rangle) \rangle \\
&= \frac{1}{2} \langle (\langle 0| \langle b| + \langle 1| \langle x(\theta)|) | (|1\rangle A |b\rangle + |0\rangle A |x(\theta)\rangle) \rangle \\
&= \langle b|Ax(\theta)\rangle \\
&= \langle b|\psi(\theta)\rangle
\end{aligned}$$

6 Résultats

Nous présentons maintenant l'application des algorithmes quantiques présentés précédemment à la résolution de l'équation de Poisson. Ces algorithmes permettant de résoudre des systèmes linéaires, il faut tout d'abord convertir l'équation $-\Delta u = f$ en un problème matriciel $Ax = b$. Nous utilisons pour cela la méthode des différences finies, les détails étant présentés dans l'annexe 8.2.1.

L'algorithme HHL nécessitant la décomposition spectrale de A , nous donnons la diagonalisation de la matrice en annexe 8.2.2.

Enfin, comme nous l'avons vu, l'algorithme VQLS nécessite la décomposition en chaînes de Pauli de A , que nous présentons en annexe 8.2.3. L'utilisation d'observables pour la mesure des quantités intéressantes fait intervenir la matrice A^2 , dont nous donnons la décomposition en chaînes de Pauli dans l'annexe 8.2.4.

Nous présentons ci-après les résultats obtenus grâce à l'algorithme HHL. En effet, bien que nous ayons réalisé l'implémentation quantique complète de l'algorithme VQLS, l'étape d'optimisation nous a posé des problèmes techniques trop complexes à résoudre pour le temps qui nous était imparti.

Un élément important à noter est le fait que les codes que nous développons ne sont pas implémentés sur des ordinateurs quantiques, mais sur des simulateurs quantiques reposant sur un processeur classique. Le gain de temps de calcul espéré n'est donc pas perceptible, ce qui limite nos capacités de calcul.

Les calculs ont donc été effectués pour $n = 3$, ce qui correspond à un découpage du segment $[0, 1]$ en $2^n = 8$ éléments. Au delà, la puissance de calcul de nos ordinateurs n'était pas suffisante pour simuler les circuits quantiques. Les résultats de la méthode HHL sont les suivants.

Résultat pour $f : x \mapsto x$ Comme nous pouvons le constater sur la figure 11a, à première vue les résultats ne sont pas satisfaisants. Or, nous avons vu que l'algorithme HHL donne une solution proportionnelle à la solution exacte. En ajoutant le coefficient de proportionnalité approprié, nous obtenons

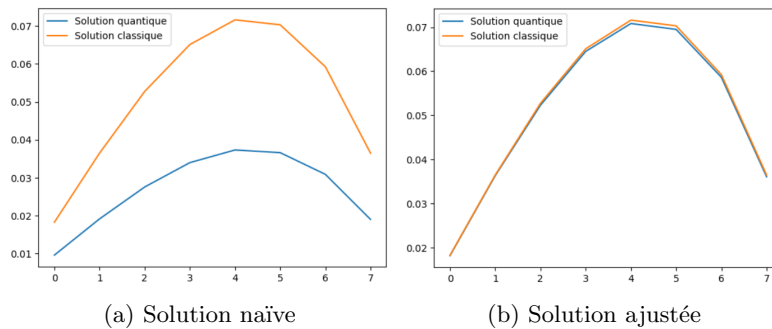


Figure 11: Solutions quantiques et classiques pour $f : x \mapsto x$

Résultats pour $f : x \mapsto \sin(x)$ Là encore, nous pouvons constater l'important du facteur de proportionnalité pertinent.

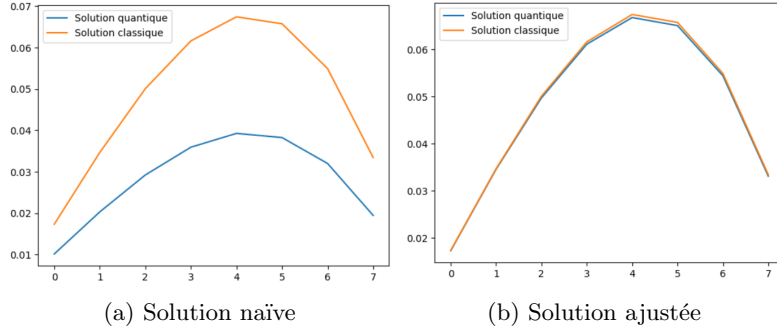


Figure 12: Solutions quantiques et classiques pour $f : x \mapsto \sin(x)$

Résultats pour $f : x \mapsto e^{-x^2}$ Ce dernier test permet de mettre en évidence l'intérêt des résolutions quantiques, permettant de trouver rapidement des solutions faisant intervenir, dans le cas analytique, des éléments complexes tels que la fonction d'erreur.

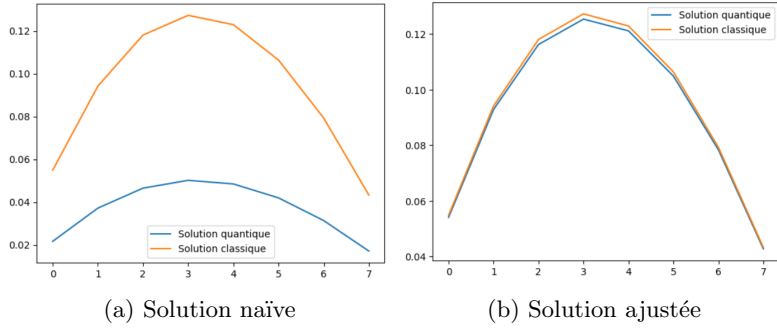


Figure 13: Solutions quantiques et classiques pour $f : x \mapsto e^{-x^2}$

Tous les codes Python ayant permis la production de ces résultats sont présents dans l'annexe 8.5 pour l'algorithme HHL et dans l'annexe 8.6 pour l'algorithme VQLS. Nous avons utilisé le langage Quantum Learning Machine (QLM), proposé par Atos, pour VQLS et le langage Qiskit, développé par IBM, pour HHL.

7 Conclusion et ouverture

Conclusion Nous avons ainsi exploré deux méthodes distinctes, l'algorithme Harrow-Hassidim-Lloyd et le Variational Quantum Linear Solver, permettant de résoudre de grands systèmes linéaires. Grâce à la méthode des différences finies, nous les avons appliqués à la résolution de l'équation de Poisson. Nous avons développé les programmes sur QLM et Qiskit permettant d'implémenter ces algorithmes. Les résultats pour la méthode HHL sont concluants, bien que nous soyons limités par la puissance de calcul de nos processeurs classiques, étant donné que nous travaillons avec des simulateurs quantiques. Ces démarches ont nécessité la compréhension de plusieurs outils fondamentaux de l'informatique quantique, tels que le test de Hadamard, le *swap test*, la transformée de Fourier quantique ou encore l'estimation de phase quantique. Nous avons aussi eu besoin d'effectuer la diagonalisation de la matrice A correspondant à l'équation de Poisson et la décomposition en chaînes de Pauli de A et A^2 .

Ouverture Ce projet de recherche nous a permis de découvrir l'informatique quantique et d'approfondir nos connaissances en mécanique quantique. Les circuits théoriques et les calculs sous-jacents ont été étudiés et compris. Avec plus de temps, il nous aurait été possible de mieux maîtriser le langage QLM, et ainsi obtenir des résultats plus convaincants pour l'algorithme VQLS. De plus, le code que nous proposons simule un circuit quantique sans prendre en compte le bruit. En effet, les portes utilisées sont supposées parfaites. Il faudrait ainsi rajouter artificiellement dans le code du bruit quantique pour se rapprocher des résultats que nous pourrions obtenir avec un véritable ordinateur quantique. Par ailleurs, le code proposé peut se généraliser pour $n > 3$. Il pourrait également se généraliser à des fonctions f et u définies dans $\mathbb{R}^q$ avec $q > 1$. Enfin, il serait intéressant d'étudier la résolution quantique d'autres équations différentielles, plus sophistiquées, afin de tirer pleinement parti de la puissance de calcul des ordinateurs quantiques.

8 Annexes

8.1 Implémentation de la transformée de Fourier quantique

Commençons avec $n = 1$. Dans ce cas, on cherche une porte quantique qui transforme $|j_0\rangle$ en

$$U_{\text{FT}}^{(n=1)} |j_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_0)} |1\rangle)$$

Notons que $(\cdot j_0)$ représente $\frac{j_0}{2}$. Ainsi,

$$U_{\text{FT}}^{(n=1)} |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad U_{\text{FT}}^{(n=1)} |1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Cela correspond à la porte d'Hadamard. On en déduit finalement que

$$U_{\text{FT}}^{(n=1)} = H$$

Intéressons nous maintenant au cas $n = 2$. Définissons les matrices de rotation contrôlée. La fonction

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_{n-1} \dots j_0)} |1\rangle)$$

est une rotation contrôlée. On peut l'implémenter à l'aide des portes de rotation R_j définies par

$$R_j = R_z\left(\frac{\pi}{2^{j-1}}\right)$$

avec $R_z(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$. Montrons que le circuit quantique correspondant à $U_{\text{FT}}^{(n=2)}$ est celui de la figure ci-dessous :

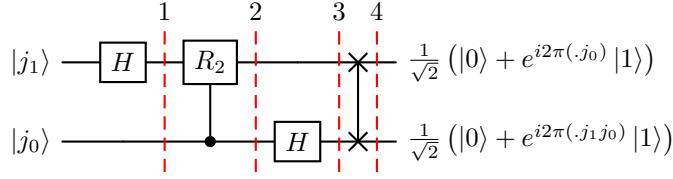


Figure 14: Circuit quantique pour $U_{\text{FT}}^{(n=2)}$

Porte H D'après le cas $n = 1$, on sait que

$$H |j_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_1)} |1\rangle).$$

Par conséquent,

$$|\psi_1\rangle = H |j_1\rangle \otimes |j_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_1)} |1\rangle) \otimes |j_0\rangle.$$

Porte R_2 La porte R_2 est définie par

$$R_2 |0\rangle = |0\rangle, \quad R_2 |1\rangle = i |1\rangle.$$

Ainsi,

$$|\psi_2\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_1)} |1\rangle) \otimes |0\rangle, & |j_0\rangle = |0\rangle \\ \frac{1}{\sqrt{2}}(|0\rangle + ie^{i2\pi(\cdot j_1)} |1\rangle) \otimes |1\rangle, & |j_0\rangle = |1\rangle \end{cases}$$

Porte H

$$\begin{aligned} |\psi_3\rangle &= \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_1)} |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), & |j_0\rangle = |0\rangle \\ \frac{1}{\sqrt{2}}(|0\rangle + ie^{i2\pi(\cdot j_1)} |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), & |j_0\rangle = |1\rangle \end{cases} \\ &= \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_1 j_0)} |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_0)} |1\rangle) \end{aligned}$$

Porte $swap$

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_0)} |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi(\cdot j_1 j_0)} |1\rangle).$$

On obtient bien $U_{\text{FT}}^{(n=2)} |j_1 j_0\rangle$ en sortie du circuit. La transformée de Fourier quantique inverse $U_{\text{FT}}^{\dagger (n=2)}$ est réalisée grâce au conjugué du circuit précédent :

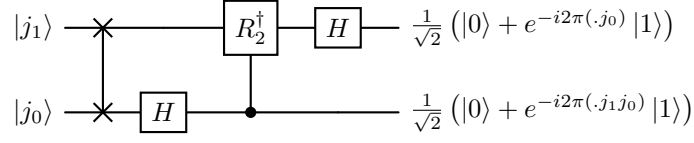


Figure 15: Circuit quantique pour $U_{\text{FT}}^{\dagger (n=2)}$

On peut alors généraliser ce schéma pour $n \in \mathbb{N}^*$ et obtenir le circuit suivant pour $U_{\text{FT}}^{(n)}$ (avant d'appliquer les opérations *swap*) :

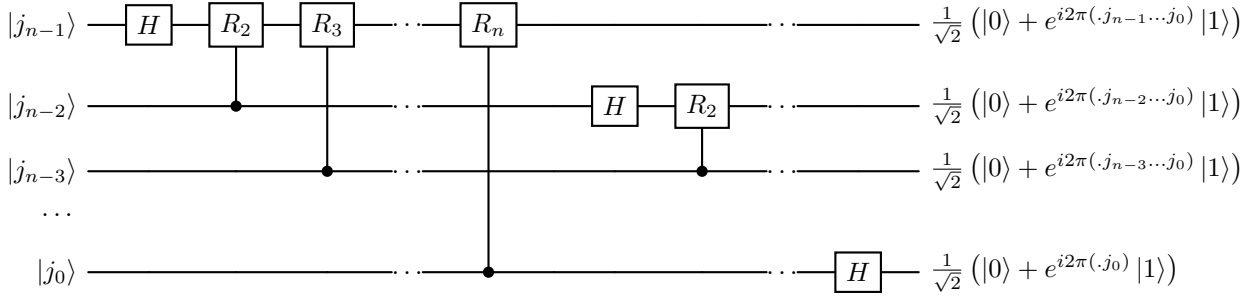


Figure 16: Circuit quantique pour $U_{\text{FT}}^{(n)}$

8.2 Détails techniques pour la résolution de l'équation de Poisson

On s'intéresse spécifiquement à l'équation de Poisson

$$-\Delta u = f$$

en 1D. On va donner une interprétation matricielle de ce problème afin d'utiliser les algorithmes de résolution de systèmes linéaires présentés précédemment pour trouver la solution.

8.2.1 Discrétisation

On commence par déterminer la matrice A associée. On utilise pour cela la méthode des différences finies. On discrétise le segment $[0, 1]$ en N points équidistants $(a_k)_{k \in \llbracket 0, N-1 \rrbracket}$. On discrétise de même les fonctions $\mathbf{x} = (u(a_k))_{k \in \llbracket 0, N-1 \rrbracket}$ et $\mathbf{b} = (f(a_k))_{k \in \llbracket 0, N-1 \rrbracket}$. Pour $h \ll 1$, on peut approcher x_k'' par

$$x_k'' \approx \frac{x_{k+1} - 2x_k + x_{k-1}}{h^2}.$$

Ainsi, en posant

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

on a la relation $-\mathbf{x}'' = A\mathbf{x}$. On en déduit donc que

$$-\Delta u = f \iff A\mathbf{x} = \mathbf{b}.$$

8.2.2 Diagonalisation de A pour l'algorithme HHL

La matrice A étant symétrique réelle, elle est diagonalisable en vertu du théorème spectral.

Ses valeurs propres sont de la forme

$$\lambda_k = \frac{2}{h^2} \left(1 + \cos \left(\frac{k\pi}{N+1} \right) \right), \quad k \in \llbracket 1, N \rrbracket$$

et ses vecteurs propres sont

$$\mathbf{v}_k = \sum_{j=1}^N \sin \left(\frac{jk\pi}{N+1} \right) e^{-ij} \mathbf{e}_j, \quad k \in \llbracket 1, N \rrbracket.$$

8.2.3 Décomposition de A en chaînes de Pauli pour l'algorithme VQLS

On cherche la décomposition en chaîne de Pauli de

$$A = \sum_{l=1}^L c_l A_l.$$

On raisonne par récurrence. On pose $N = 2^m$.

Pour $m = 1$:

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} = A^{(1)}$$

La décomposition en chaîne de Pauli s'écrit alors $A^{(1)} = 2I - X$. On a donc $c_1 = 2, A_1 = I, c_2 = -1$ et $A_2 = X$.

Pour $m = 2$:

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} = A^{(2)}$$

Pour simplifier les notations, on pose

$$\sigma_+ = \frac{X + iY}{2} \text{ et } \sigma_- = \frac{X - iY}{2}.$$

La décomposition donne $A^{(2)} = I \otimes A^{(1)} - \sigma_+ \otimes \sigma_- - \sigma_- \otimes \sigma_+ = 2I \otimes I - I \otimes X - \frac{1}{2}X \otimes X - \frac{1}{2}Y \otimes Y$.

Pour $m \in \mathbb{N}^*$: On peut montrer que

$$A^{(m)} = I \otimes A^{(m-1)} - \sigma_+ \otimes \sigma_-^{\otimes m-1} - \sigma_- \otimes \sigma_+^{\otimes m-1}$$

8.2.4 Décomposition de A^2 en chaînes de Pauli pour l'algorithme VQLS

La matrice A^2 s'écrit

$$\begin{aligned} A^2 &= \frac{1}{h^4} \begin{pmatrix} 5 & -4 & 1 & 0 & \cdots & 0 \\ -4 & 6 & -4 & \ddots & \ddots & \vdots \\ 1 & -4 & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & -4 & 1 \\ \vdots & \ddots & \ddots & -4 & 6 & -4 \\ 0 & \cdots & 0 & 1 & -4 & 5 \end{pmatrix} \\ &= \frac{1}{h^4} \underbrace{\begin{pmatrix} 6 & -4 & 1 & 0 & \cdots & 0 \\ -4 & 6 & -4 & \ddots & \ddots & \vdots \\ 1 & -4 & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & -4 & 1 \\ \vdots & \ddots & \ddots & -4 & 6 & -4 \\ 0 & \cdots & 0 & 1 & -4 & 6 \end{pmatrix}}_B - \frac{1}{h^4} \underbrace{\begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix}}_C \end{aligned}$$

On peut montrer que

$$B^{(m)} = I \otimes B^{(m-1)} + \sigma_- \otimes \sigma_+^{m-2} \otimes (I - 4\sigma_+) + \sigma_+ \otimes \sigma_-^{m-2} \otimes (I - 4\sigma_-)$$

et

$$C^{(m)} = (\sigma_+ \sigma_-)^{\otimes m} + (\sigma_- \sigma_+)^{\otimes m}.$$

8.3 Détail du calcul de l'algorithme HHL

Faisons deux remarques :

- Premièrement, les coordonnées du $|x\rangle$ solution ne sont pas les $\frac{b_j}{\lambda_j}$ étant donné que l'on travaille ici dans la base propre de A , alors que les coordonnées de $|x\rangle$ s'exprimeront dans la base des qubits $|0\rangle$ et $|1\rangle$.
- Deuxièmement, les vecteurs $|u_j\rangle$ et $|b\rangle$ doivent être normalisés pour être intégrés au circuit. On impose donc les conditions

$$\sum_{j=0}^{2^{n_b}-1} |b_j|^2 = 1, \quad \sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j}{\lambda_j} \right|^2 = 1. \quad (2)$$

Les quatre phases énumérées précédemment vont maintenant être décrites une à une.

8.3.1 Préparation de l'état

Au-début du circuit, l'état est

$$|\psi_0\rangle = |0\rangle_b |0\rangle_c |0\rangle_a.$$

Le registre b doit être modifié pour contenir $|b\rangle$. Comme b s'écrit $\begin{pmatrix} b_0 \\ \vdots \\ b_{N_b-1} \end{pmatrix}$, on a

$$|b\rangle = b_0 |0\rangle + \dots + b_{N_b-1} |N_b - 1\rangle$$

dans l'espace engendré par les n_b qubits. En définitive, b peut être écrit dans le registre b. Ainsi, on passe à l'état

$$|\psi_1\rangle = |b\rangle_b |0\rangle_c |0\rangle_a.$$

8.3.2 Estimation de phase quantique

Portes de Hadamard On applique l'opérateur $I^{\otimes n_b} H^{\otimes n} I$ à $|\psi_2\rangle$ ce qui donne l'état

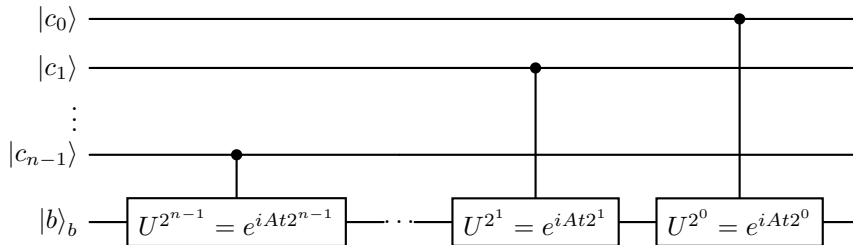
$$|\psi_2\rangle = |b\rangle_b \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right)^{\otimes n} |0\rangle_a.$$

Rotations contrôlées On définit la matrice de rotation unitaire

$$U = e^{iAt}.$$

Le but de la QPE est de déterminer les phases des valeurs propres de U , qui seront proportionnelles aux valeurs propres de A . Ainsi après la QPE, les valeurs propres de A seront stockées dans le registre c.

Des portes contrôlées sont appliquées à $|b\rangle$. Chaque qubit r du registre c contrôle une porte $U^{2^r} = e^{iAt2^r}$ qui agit sur le registre b. le circuit ci-dessous décrit précisément les actions de chacune de ces portes contrôlées.



Pour simplifier, commençons par supposer que $|b\rangle$ est un vecteur propre de U . Il existe alors une phase $\phi \in \mathbb{R}$ telle que $U|b\rangle = e^{i2\pi\phi}|b\rangle$. En effet, U étant unitaire, ses valeurs propres sont des racines de l'unité. La transformation est la suivante : lorsque $|c_r\rangle$ vaut $|1\rangle$, on applique la transformation $|b\rangle \rightarrow e^{i2\pi\phi 2^r}|b\rangle$; lorsque $|c_r\rangle$ vaut $|0\rangle$, on ne fait rien. On peut donc modéliser l'action du registre c sur le registre b par une multiplication des $|1\rangle$ du registre c :

$$|\psi_3\rangle = |b\rangle_b \frac{1}{2^{\frac{n}{2}}} \left(|0\rangle + e^{i2\pi\phi 2^{n-1}} |1\rangle \right) \dots \left(|0\rangle + e^{i2\pi\phi 2^0} |1\rangle \right) |0\rangle_a.$$

Finalement, en développant, on obtient

$$|\psi_3\rangle = |b\rangle_b \frac{1}{2^{\frac{n}{2}}} \sum_{k=0}^{2^n-1} e^{i2\pi\phi k} |k\rangle |0\rangle_a.$$

Transformée de Fourier quantique inverse Nous appliquons maintenant la transformée de Fourier quantique inverse à l'état ψ_3 . Rappelons que pour tout vecteur de base $|k\rangle$,

$$U_{\text{FT}}^\dagger(|k\rangle) = \frac{1}{2^{\frac{n}{2}}} \sum_{l=0}^{2^n-1} e^{-i2\pi\frac{l}{2^n}k} |l\rangle.$$

Ici, on a donc

$$U_{\text{FT}}^\dagger(|\psi_3\rangle) = |b\rangle_b \frac{1}{2^{\frac{n}{2}}} \sum_{k=0}^{2^n-1} e^{i2\pi\phi k} U_{\text{FT}}^\dagger(|k\rangle) |0\rangle_a = |b\rangle_b \frac{1}{2^n} \sum_{k=0}^{2^n-1} \sum_{l=0}^{2^n-1} e^{i2\pi k(\phi - \frac{l}{2^n})} |l\rangle |0\rangle_a.$$

Or, $\exists l_\phi \in \llbracket 0, 2^n - 1 \rrbracket$ tel que $\phi = \frac{l_\phi}{2^n}$. Pour $l = l_\phi$, le terme a une amplitude égale à 2^n . Sinon, l'amplitude vaut $\sum_{k=0}^{2^n-1} e^{i2\pi k(\phi - \frac{l}{2^n})} = 0$. Ainsi,

$$U_{\text{FT}}^\dagger(|\psi_3\rangle) = |b\rangle_b \frac{1}{2^n} \sum_{k=0}^{2^n-1} 2^n |l_\phi\rangle |0\rangle_a = |b\rangle_b |2^n\phi\rangle_c |0\rangle_a. \quad (3)$$

Or ici, $|b\rangle$ n'est pas nécessairement un vecteur propre de U . Mais on peut modifier la démarche effectuée pour adapter le résultat. En effet, supposons maintenant que $|b\rangle = |u_j\rangle$. Alors, on a $U|b\rangle = e^{iAt} |u_j\rangle = e^{i\lambda_j t} |u_j\rangle$. En confrontant $i\lambda_j t$ et $i2\pi\phi$, on obtient $\phi = \frac{\lambda_j t}{2\pi}$. En injectant cette expression de ϕ dans le résultat (3), on obtient

$$U_{\text{FT}}^\dagger(|\psi_3\rangle) = |u_j\rangle_b \left| \frac{2^{n-1}t}{\pi} \lambda_j \right\rangle_c |0\rangle_a.$$

Comme $|b\rangle = \sum_{j=0}^{2^{n_b}-1} b_j |u_j\rangle$, on arrive finalement à

$$U_{\text{FT}}^\dagger(|\psi_3\rangle) = \sum_{j=0}^{2^{n_b}-1} b_j |u_j\rangle \left| \frac{2^{n-1}t}{\pi} \lambda_j \right\rangle_c |0\rangle_a.$$

Les λ_j n'étant en général pas des entiers, on choisit t tel que les $\tilde{\lambda}_j = \frac{2^{n-1}t}{\pi} \lambda_j$ soient des entiers. Ainsi, on arrive au résultat

$$|\psi_4\rangle = \sum_{j=0}^{2^{n_b}-1} b_j |u_j\rangle |\tilde{\lambda}_j\rangle |0\rangle_a.$$

8.3.3 Rotation du registre auxiliaire et mesure

On effectue ensuite une rotation du registre a, contrôlée par le registre c, de manière à obtenir

$$|\psi_5\rangle = \sum_{j=0}^{2^{n_b}-1} b_j |u_j\rangle |\tilde{\lambda}_j\rangle \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} |0\rangle + \frac{C}{\tilde{\lambda}_j} |1\rangle \right)$$

où C est une constante à choisir la plus grande possible.

Puis on mesure le qubit du registre a jusqu'à obtenir $|1\rangle$. L'effondrement de la fonction d'onde donne alors l'état

$$|\psi_6\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j C}{\tilde{\lambda}_j} \right|^2}} \sum_{j=0}^{2^{n_b}-1} b_j |u_j\rangle |\tilde{\lambda}_j\rangle \frac{C}{\tilde{\lambda}_j} |1\rangle_a.$$

Ce résultat ressemble à (2). Mais on remarque que les opérations précédentes ont projeté le problème dans la base propre de A . Or, on a besoin de retourner dans la base des qubits $|0\rangle$ et $|1\rangle$ pour lire le résultat. C'est pourquoi on doit effectuer une QPE inverse afin de retourner dans cette base canonique.

8.3.4 Estimation de phase quantique inverse

Transformée de Fourier quantique On rappelle que pour tout vecteur de base $|k\rangle$,

$$U_{\text{FT}}(|k\rangle) = \frac{1}{2^{\frac{n}{2}}} \sum_{l=0}^{2^n-1} e^{i2\pi \frac{l}{2^n} k} |l\rangle.$$

Ici,

$$U_{\text{FT}}(|\psi_6\rangle) = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j C}{\tilde{\lambda}_j} \right|^2}} \sum_{j=0}^{2^{n_b}-1} C \frac{b_j}{\tilde{\lambda}_j} |u_j\rangle U_{\text{FT}}(|\tilde{\lambda}_j\rangle) |1\rangle_a$$

d'où

$$|\psi_7\rangle = \frac{1}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j C}{\tilde{\lambda}_j} \right|^2}} \sum_{j=0}^{2^{n_b}-1} C \frac{b_j}{\tilde{\lambda}_j} |u_j\rangle \left(\frac{1}{2^{\frac{n}{2}}} \sum_{l=0}^{2^n-1} e^{i2\pi \frac{l}{2^n} \tilde{\lambda}_j} |l\rangle \right) |1\rangle_a.$$

Rotations contrôlées On reproduit la même démarche que pour la QPE, sauf que les rotations sont cette fois de la forme $(U^{-1})^{2^r}$ pour la rotation contrôlée par le qubit r du registre c . Cela revient à multiplier chacun des termes de la somme sur l par $e^{-i\lambda_j t l}$, par le même raisonnement que précédemment. On obtient

$$|\psi_8\rangle = \frac{1}{2^{\frac{n}{2}} \sqrt{\sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j C}{\tilde{\lambda}_j} \right|^2}} \sum_{j=0}^{2^{n_b}-1} C \frac{b_j}{\tilde{\lambda}_j} |u_j\rangle \left(\sum_{l=0}^{2^n-1} e^{-i\lambda_j t l} e^{i2\pi \frac{l}{2^n} \tilde{\lambda}_j} |l\rangle \right) |1\rangle_a.$$

Or, comme on avait choisi $\tilde{\lambda}_j = \frac{2^{n-1}t}{\pi} \lambda_j$, le terme $e^{-i\lambda_j t l} e^{i2\pi \frac{l}{2^n} \tilde{\lambda}_j}$ vaut 1 et donc

$$|\psi_8\rangle = \frac{C}{2^{\frac{n}{2}} \sqrt{\sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j C}{\tilde{\lambda}_j} \right|^2}} \sum_{j=0}^{2^{n_b}-1} \frac{b_j}{\tilde{\lambda}_j} |u_j\rangle \sum_{l=0}^{2^n-1} |l\rangle |1\rangle_a.$$

On reconnaît $|x\rangle = \sum_{j=0}^{2^{n_b}-1} \frac{b_j}{\tilde{\lambda}_j} |u_j\rangle$ et finalement :

$$|\psi_8\rangle = \frac{C}{2^{\frac{n}{2}} \sqrt{\sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j C}{\tilde{\lambda}_j} \right|^2}} |x\rangle_b \sum_{l=0}^{2^n-1} |l\rangle |1\rangle_a.$$

Portes de Hadamard En appliquant les portes de Hadamard au registre c , on obtient

$$|\psi_9\rangle = \frac{C}{\sqrt{\sum_{j=0}^{2^{n_b}-1} \left| \frac{b_j C}{\tilde{\lambda}_j} \right|^2}} |x\rangle_b |0\rangle_c |1\rangle_a.$$

Si $C \in \mathbb{R}$ et en utilisant la condition (2), l'état final du circuit est

$$|\psi_9\rangle = |x\rangle_b |0\rangle_c |1\rangle_a.$$

Ainsi, $|x\rangle$ est accessible dans le registre b .

8.4 Préparation de l'ansatz des algorithmes variationnels

La porte $U(\theta)$ prépare l'ansatz avant de l'insérer dans le circuit quantique. Il s'agit d'un élément fondamental du circuit. On évalue l'efficacité de l'ansatz selon deux critères :

- expressivité (*expressivity*) : aptitude du circuit à balayer l'espace des solutions ;
- capacité d'entraînement (*trainability*) : aptitude à pouvoir optimiser (trouver un θ^* solution) le circuit en un temps environ polynomial avec la profondeur du circuit.

L'objectif est de trouver un bon compromis entre ces deux critères. Plusieurs techniques peuvent être utilisées.

Hardware efficient ansatz (HEA) On applique des portes faciles à implémenter sur l'ordinateur quantique comme R_X , R_Y , R_Z , que l'on répète plusieurs fois. Les circuits utilisés sont de ce type :

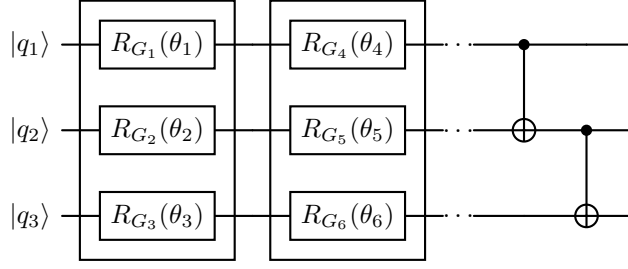


Figure 17: Un exemple d'ansatz

Les G_i sont dans $\{I, X, Y, Z\}$. C'est une ansatz difficile à entraîner, mais avec une bonne expressivité.

Unitary couple cluster simple double (UCCSD) On utilise des opérateurs bien connus pour la chimie sur l'ordinateur quantique :

$$U(\theta) = \exp \left(\sum_{p,q} \frac{\theta_{p,q}}{2} \left(\prod_k \hat{a}_{p,k}^\dagger \hat{a}_{q,k} - \prod_k \hat{a}_{q,k}^\dagger \hat{a}_{p,k} \right) \right)$$

construit à partir des opérateurs d'annihilation et de création. L'état ansatz est construit à partir de l'état de référence de Hartree-Fock :

$$|\psi(\theta)\rangle = U(\theta) |\psi_{\text{HF}}\rangle.$$

Adaptive Derivative-Assembled Pseudo-Trotter - VQE (ADAPT-VQE)

$$\mathcal{A} = \{G_{iq}(\theta_{iq})\}, \quad G_{iq}(\theta_{iq}) = \exp \left(-i \frac{\theta_{iq}}{2} \hat{P}_{iq} \right), \quad A = |\mathcal{A}|.$$

À chaque itération, $\psi^{(k)}(\theta) = G_k(\theta_k) \dots G_1(\theta_1) |\psi_{\text{ref}}\rangle$. On adopte la stratégie suivante :

$$\begin{aligned} \tilde{\psi}_{i1}^{(k)} &= G_{i1}(\theta_{i1}) |\psi^{(k)}(\theta)\rangle \\ &\vdots \\ \tilde{\psi}_{iA}^{(k)} &= G_{iA}(\theta_{iA}) |\psi^{(k)}(\theta)\rangle \\ \frac{\partial \tilde{E}}{\partial \theta_{ij}}(\theta) &= \langle \tilde{\psi} | [\hat{H}, \hat{P}_{ij}] | \tilde{\psi} \rangle \forall i, j. \end{aligned}$$

On choisit le G_{ij} qui maximise le gradient de $\tilde{E}(\theta)$. Une fois choisi, on l'applique :

$$U^{(k+1)}(\theta) = G_{k+1} U^{(k)}(\theta)$$

et

$$|\psi^{(k+1)}(\theta)\rangle = G_{k+1}(\theta_{k+1}) \dots G_1(\theta_1) |\psi_{\text{ref}}\rangle$$

et on continue le procédé algorithmique de VQE en optimisant les $k+1$ paramètres.

8.5 Code pour l'algorithme HHL

```
import numpy as np
from qiskit.quantum_info import Statevector

n = ...

#Matrice pour la solution quantique
tridi_matrix = TridiagonalToeplitz(n, 2, -1)
```

```

#Matrice pour la solution classique
a = [-1 for i in range(2**n-1)]
b = [2 for i in range(2**n)]
A = np.diag(a, -1) + np.diag(b, 0) + np.diag(a, 1)
matrix = np.array(A)

#Création du vecteur b
vector = np.array(...)

#Solution quantique
tridi_solution = HHL().solve(tridi_matrix, vector)
tridi_sv = Statevector(tridi_solution.state).data
tridi_full_vector = np.array([tridi_sv[2**(4*n)+i] for i in range(2**n)])

#Solution classique
matrix_inv = np.linalg.inv(matrix)
x = np.dot(matrix_inv, vector)

```

8.6 Code pour l'algorithme VQLS

Nous utilisons le langage Quantum Learning Machine (QLM) développé par Atos. Il permet de réaliser des simulations quantiques.

8.6.1 Importation des bibliothèques nécessaires

```

from qat.lang.AQASM import *
from qat.core import Observable, Term
from qat.qpus import PyLinalg, get_default_qpu
from qat.lang.models import KPTree
import numpy as np
from copy import deepcopy
import random as rand

```

8.6.2 Préparation de l'ansatz

Nous utilisons une Hardware Efficient Ansatz à `n_blocs` blocs. Elle utilise des portes aléatoirement choisies parmi R_X , R_Y et R_Z , paramétrées par les composantes de la variable à optimiser `theta`.

```

def ansatz(qbits, theta, n_blocs):
    portes=[RX,RY,RZ]
    n=len(qbits)
    for i in range(n_blocs):
        for j in range(n):
            prog.apply(portes[rand.randint(0,2)](theta[i*n+j]),qbits[j])
        for j in range(n-1):
            prog.apply(CNOT,qbits[j],qbits[j+1])

```

Nous définissons aussi sa version contrôlée, qui n'agira que lorsque le qbit ancilla sera dans l'état $|0\rangle$:

```

def ansatz_ctrl(qbits, theta, n_blocs):
    portes=[RX,RY,RZ]
    n=len(qbits)-1
    for i in range(n_blocs):
        for j in range(n):
            prog.apply(portes[rand.randint(0,2)](theta[i*n+j]).ctrl(),qbits[0],qbits[j+1])
        for j in range(n-1):
            prog.apply(CNOT.ctrl(),qbits[0],qbits[j+1],qbits[j+2])

```

8.6.3 Décompositions en chaînes de Pauli

Nous avons vu que la décomposition de A et A^2 en chaînes de Pauli nécessite les produits $\sigma_-^{\otimes n}$ et $\sigma_+^{\otimes n}$, que nous donnons ci-après.

```

def sigma_m(n):
    if n==0:
        return [[]]
    if n==1:
        a=[[X],[Y]]
        c=[1/2,-1j/2]
        a.append(c)
        return(a)
    else:
        a=[[X]+sigma_m(n-1)[i] for i in range(2**(n-1))]+[[Y]+sigma_m(n-1)[i] for i in
        ↪ range(2**(n-1))]
        c=[]
        for el in sigma_m(n-1)[-1]:
            c.append(el/2)
            c.append(-1j*el/2)
        a.append(c)
        return(a)
def sigma_m_app(qbits):
    n=len(qbits)
    sigma_m_n=sigma_m(n)
    for i in range(2**n):
        for j in range(n):
            prog.apply(sigma_m_n[i][j],qbits[j])

def sigma_p(n):
    if n==0:
        return [[]]
    if n==1:
        a=[[X],[Y]]
        c=[1/2,1j/2]
        a.append(c)
        return(a)
    else:
        a=[[X]+sigma_p(n-1)[i] for i in range(2**(n-1))]+[[Y]+sigma_p(n-1)[i] for i in
        ↪ range(2**(n-1))]
        c=[]
        for el in sigma_p(n-1)[-1]:
            c.append(el/2)
            c.append(1j*el/2)
        a.append(c)
        return(a)
def sigma_p_app(qbits):
    n=len(qbits)
    sigma_p_n=sigma_p(n)
    for i in range(2**n):
        for j in range(n):
            prog.apply(sigma_p_n[i][j],qbits[j])

```

Nous définissons ensuite une fonction de transformation du type porte de QLM au type chaîne de caractères de Python, et inversement.

```

def trans(G):
    if G==I:
        return "I"
    elif G==X:
        return "X"
    elif G==Y:
        return "Y"
    else:
        return "Z"
def trans_(G):
    if G=="I":

```

```

    return I
elif G=="X":
    return X
elif G=="Y":
    return Y
else:
    return Z

```

Les fonctions définies précédemment permettant de coder la transformée de Pauli de l'observable associée à la matrice A :

```

def pauli_A(n):
    if n==1:
        a=[[I],[X]]
        c=[2,-1]
        a.append(c)
        return(a)
    else:
        pauli_A_1=pauli_A(n-1)
        sigma_m_1=sigma_m(n-1)
        sigma_p_1=sigma_p(n-1)
        sigma_p_m=[[X]+sigma_m_1[i] for i in range(len(sigma_m_1)-1)]+[Y]+sigma_m_1[i] for
        ↪ i in range(len(sigma_m_1)-1)]
        sigma_m_p=[[X]+sigma_p_1[i] for i in range(len(sigma_p_1)-1)]+[Y]+sigma_p_1[i] for
        ↪ i in range(len(sigma_p_1)-1)]
        c=pauli_A_1[-1]
        a=[[I]+pauli_A_1[i] for i in range(len(pauli_A_1)-1)]+sigma_p_m+sigma_m_p
        for el in sigma_m(n-1)[-1]:
            c.append(-el/2)
            c.append(-1j*el/2)
        for el in sigma_p(n-1)[-1]:
            c.append(-el/2)
            c.append(1j*el/2)
        a_=[]
        for el in a:
            t=[]
            for i in range(n):
                t.append(trans(el[i]))
            a_.append(tuple(t))
        dict={}
        for i in range(len(c)):
            if a_[i] in dict.keys():
                dict[a_[i]]+=c[i]
            else:
                dict[a_[i]]=c[i]
        a__=[]
        for el in dict.keys():
            if dict[el]!=0:
                s=[]
                for i in range(n):
                    s.append(trans_(el[i]))
                a__.append(s)
        c_=[]
        for el in dict.values():
            if el!=0:
                c_.append(el)
        a__.append(c_)
        return a__
def pauli_A_app(qbits):
    n=len(qbits)
    pauli_A_n=pauli_A(n)

```

```

print("coefficients : ",pauli_A_n[-1])
for i in range(2**n):
    for j in range(n):
        prog.apply(pauli_A_n[i][j],qbits[j])

```

Nous codons de même l'observable correspondant à $X \otimes A$:

```

def pauli_AX(n):
    if n==1:
        a=[[X,I],[X,X]]
        c=[2,-1]
        a.append(c)
        return(a)
    else:
        pauli_A=pauli_A(n)
        a=[[X]+pauli_A[i] for i in range(len(pauli_A)-1)]
        c=pauli_A[-1]
        a.append(c)
        return a
def pauli_AX_app(qbits):
    n=len(qbits)-1
    pauli_AX_n=pauli_AX(n)
    print("coefficients : ",pauli_AX_n[-1])
    for i in range(2**n):
        for j in range(n+1):
            prog.apply(pauli_AX_n[i][j],qbits[j])

```

Pour la décomposition de A^2 en chaînes de Pauli, nous avons vu qu'il fallait utiliser l'expression $A^2 = B - C$, où B et C ont été définies précédemment. Nous donnons donc successivement les décompositions permettant d'obtenir B , C et A^2 .

```

def pauli_B(n):
    if n==1:
        a=[[I],[X]]
        c=[6,-4]
        a.append(c)
        return(a)
    else:
        sigma_m_1=sigma_m(n-1)
        sigma_p_1=sigma_p(n-1)
        sigma_m_2=sigma_m(n-2)
        sigma_p_2=sigma_p(n-2)
        sigma_p_m=[[X]+sigma_m_1[i] for i in range(len(sigma_m_1)-1)]+[Y]+sigma_m_1[i] for
        ↪ i in range(len(sigma_m_1)-1)]
        sigma_m_p=[[X]+sigma_p_1[i] for i in range(len(sigma_p_1)-1)]+[Y]+sigma_p_1[i] for
        ↪ i in range(len(sigma_p_1)-1)]
        sigma_p_m_2=[[X]+sigma_m_2[i]+[I] for i in
        ↪ range(len(sigma_m_2)-1)]+[Y]+sigma_m_2[i]+[I] for i in range(len(sigma_m_2)-1)]
        sigma_m_p_2=[[X]+sigma_p_2[i]+[I] for i in
        ↪ range(len(sigma_p_2)-1)]+[Y]+sigma_p_2[i]+[I] for i in range(len(sigma_p_2)-1)]
        pauli_B_1=pauli_B(n-1)
        c=pauli_B_1[-1]
        a=[[I]+pauli_B_1[i] for i in
        ↪ range(len(pauli_B_1)-1)]+sigma_p_m+sigma_m_p+sigma_p_m+sigma_m_p_
        for el in sigma_m_1[-1]:
            c.append(-4*el/2)
            c.append(-4*1j*el/2)
        for el in sigma_p_1[-1]:
            c.append(-4*el/2)
            c.append(4*1j*el/2)
        for el in sigma_m_2[-1]:
            c.append(el/2)

```

```

        c.append(1j*el/2)
    for el in sigma_p_2[-1]:
        c.append(el/2)
        c.append(-1j*el/2)
    a_=[]
    for el in a:
        t=[]
        for i in range(n):
            t.append(trans(el[i]))
        a_.append(tuple(t))
    dict={}
    for i in range(len(a)):
        if a_[i] in dict.keys():
            dict[a_[i]]+=c[i]
        else:
            dict[a_[i]]=c[i]
    a__=[]
    for el in dict.keys():
        if dict[el]!=0:
            s=[]
            for i in range(n):
                s.append(trans_(el[i]))
            a__.append(s)
    c_=[]
    for el in dict.values():
        if el!=0:
            c_.append(el)
    a__.append(c_)
    return a__

def pauli_C(n):
    if n==1:
        a=[[I]]
        c=[1]
        a.append(c)
        return(a)
    else:
        pauli_C_1=pauli_C(n-1)
        a=[[I]+pauli_C_1[i] for i in range(len(pauli_C_1)-1)]+[[Z]+pauli_C_1[i] for i in
        ↪ range(len(pauli_C_1))]
        c=pauli_C_1[-1]
        for i in range(len(pauli_C_1[-1])//2):
            c.append(pauli_C_1[-1][i]/2)
            c.append(pauli_C_1[-1][i]/2)
        for i in range(len(pauli_C_1[-1])//2,len(pauli_C_1[-1])):
            c.append(pauli_C_1[-1][i]/2)
            c.append(-pauli_C_1[-1][i]/2)
        a_=[]
        for el in a:
            t=[]
            for i in range(n):
                t.append(trans(el[i]))
            a_.append(tuple(t))
        dict={}
        for i in range(len(a)):
            if a_[i] in dict.keys():
                dict[a_[i]]+=c[i]
            else:
                dict[a_[i]]=c[i]
        a__=[]
        for el in dict.keys():

```

```

        if dict[el] != 0:
            s=[]
            for i in range(n):
                s.append(trans_(el[i]))
            a_.append(s)
c_=[]
for el in dict.values():
    if el != 0:
        c_.append(el)
a_.append(c_)
return a_

def pauli_A_2(n):
    if n==1:
        a=[[I],[X]]
        c=[6,-4]
        a.append(c)
        return(a)
    else:
        pauli_B_n=pauli_B(n)
        pauli_C_n=pauli_C(n)
        a=[pauli_B_n[i] for i in range(len(pauli_B_n)-1)]+[pauli_C_n[i] for i in
            ↪ range(len(pauli_C_n))]
        c=pauli_B_n[-1]
        c_C=pauli_C_n[-1]
        for el in c_C:
            c.append(-el)
        a_=[]
        for el in a:
            t=[]
            for i in range(n):
                t.append(trans(el[i]))
            a_.append(tuple(t))
        dict={}
        for i in range(len(c)):
            if a_[i] in dict.keys():
                dict[a_[i]]+=c[i]
            else:
                dict[a_[i]]=c[i]
        a_=[]
        for el in dict.keys():
            if dict[el] != 0:
                s=[]
                for i in range(n):
                    s.append(trans_(el[i]))
                a_.append(s)
        c_=[]
        for el in dict.values():
            if el != 0:
                c_.append(el)
        a_.append(c_)
        return a_

def pauli_A_2_app(qbits):
    n=len(qbits)
    pauli_A_2_n=pauli_A_2(n)
    print("coefficients : ",pauli_A_2_n[-1])
    for i in range(len(pauli_A_2_n)-1):
        for j in range(n):
            prog.apply(pauli_A_2_n[i][j],qbits[j])

```

8.6.4 Observables

Nous donnons enfin les codes permettant l'application des observables A^2 et $X \otimes A$ respectivement aux états $x(\theta)$ et $bx(\theta)$:

```
def observ1(n):
    obs=Observable(2*n+1,pauli_terms=[],constant_coeff=0)
    pauli_A_2_n=pauli_A_2(n)
    for l in range(len(pauli_A_2_n)-1):
        terme=pauli_A_2_n[l]
        portes=""
        qbits=[]
        for i in range(n):
            if terme[i]!=I:
                portes+=trans(terme[i])
                qbits.append(i)
        if len(qbits)==0:
            obs+=Observable(2*n+1,pauli_terms=[],constant_coeff=pauli_A_2_n[-1][1])
        else:
            ↪ obs+=Observable(2*n+1,pauli_terms=[Term(pauli_A_2_n[-1][1],portes,qbits)],constant_coef

return obs

def observ2(n):
    obs=Observable(2*n+1,pauli_terms=[],constant_coeff=0)
    pauli_AX_n=pauli_AX(n)
    for l in range(len(pauli_AX_n)-1):
        terme=pauli_AX_n[l]
        portes=""
        qbits=[]
        for i in range(n,2*n+1):
            if terme[i-n]!=I:
                portes+=trans(terme[i-n])
                qbits.append(i)
        if len(qbits)==0:
            obs+=Observable(2*n+1,pauli_terms=[],constant_coeff=pauli_AX_n[-1][1])
        else:
            ↪ obs+=Observable(2*n+1,pauli_terms=[Term(pauli_AX_n[-1][1],portes,qbits)],constant_coeff

return obs
```

Pour terminer, nous appliquons ces observables et simulons les circuits quantiques. Voici le programme pour évaluer $\langle x(\theta)|A^2|x(\theta)\rangle$:

```
def fonction(theta):
    n=2
    prog = Program()
    theta=[prog.new_var(float,'\\th_%s'%i) for i in range(n*n_blocs)]
    tree = KPTree(data)
    routine_state_prep = tree.get_routine(0)
    qbits = prog.qalloc(2*n+1)
    prog.apply(H,qbits[n])
    routine_state_prep.ctrl()(qbits[n],qbits[n+1:])
    prog.apply(X,qbits[n])
    for i in range(n_blocs):
        for j in range(n+1,2*n+1):
            #j=k-n-1
            RX(theta[i*n+j-n-1]).ctrl()(qbits[n],qbits[j])
        for j in range(n+1,2*n):
            prog.apply(CNOT.ctrl(),qbits[n],qbits[j],qbits[j+1])
    prog.apply(X,qbits[n])
    for i in range(n_blocs):
```

```

    for j in range(n):
        RX(theta[i*n+j])(qbits[j])
    for j in range(n-1):
        CNOT(qbits[j],qbits[j+1])
circuit = prog.to_circ()
value = {"\\theta": theta0}
qpu = get_default_qpu()
job1 = circuit.to_job(observable=observ1(n))
job = circuit.to_job(observable=observ2(n))
job1_=job1(**value)
job_=job(**value)
result = qpu.submit(job1_)
result1=result.value
result = qpu.submit(job_)
result2=result.value
energy=result1-abs(result2)**2
print(energy)
return energy

```

8.6.5 Optimisation de la fonction coût

```

from scipy.optimize import minimize
res = minimize(fonction, theta0)

```

Remerciements

Nous tenons tout d'abord à remercier l'École des Ponts, et plus particulièrement le Département de première année, pour la mise en place du parcours recherche qui nous a permis de nous intéresser à un sujet aussi passionnant que l'informatique quantique pendant quelques mois. Merci donc à François Chevoir qui, en tant que président du Département, a rendu l'existence de ce parcours possible.

Nous souhaitons évidemment adresser nos plus sincères remerciements à Eric Cances, chercheur au CERMICS et enseignant à l'École des Ponts, qui a été notre encadrant tout au long de ce projet de recherche. Nous le remercions pour son cours de mécanique quantique que nous avons suivi avec intérêt, et pour son soutien constant et son investissement à nos côtés pour que ce projet aboutisse.

Nous tenons aussi à remercier Alicia Negre, doctorante au CERMICS, pour son apport sur les algorithmes variationnels.

Enfin, nous remercions Henri Pinsolle, élève de l'École, pour son apport sur la programmation des circuits quantiques.

References

- [Bodin 2021] BODIN, Arnaud: *Quantum - Un peu de mathématiques pour l'informatique quantique*. Exo7, 2021 (Algorithmes et mathématiques). – URL <http://exo7.emath.fr/cours/livre-quantum.pdf>
- [Bravo-Prieto u. a. 2023] BRAVO-PRIETO, Carlos ; LAROSE, Ryan ; CEREZO, M. ; SUBASI, Yigit ; CINCIO, Lukasz ; COLES, Patrick J.: Variational Quantum Linear Solver. In: *Quantum* 7 (2023), November, S. 1188. – URL <http://dx.doi.org/10.22331/q-2023-11-22-1188>. – ISSN 2521-327X
- [Cances 2016] CANCES, Eric: *An elementary introduction to quantum entanglement and its applications in physics and quantum computer science*. 2016
- [Cilds 2022] CILDS, Andrew M.: *Lecture Notes on Quantum Algorithms*. 2022. – URL <https://www.cs.umd.edu/~amchilds/qa/qa.pdf>
- [Harrow u. a. 2009] HARROW, Aram W. ; HASSIDIM, Avinatan ; LLOYD, Seth: Quantum Algorithm for Linear Systems of Equations. In: *Phys. Rev. Lett.* 103 (2009), Oct, S. 150502. – URL <https://link.aps.org/doi/10.1103/PhysRevLett.103.150502>
- [Kay 2023] KAY, Alastair: *Tutorial on the Quantikz Package*. 2023. – URL <http://mirrors.ibiblio.org/CTAN/graphics/pgf/contrib/quantikz/quantikz.pdf>
- [Lin 2022] LIN, Lin: *Lecture Notes on Quantum Algorithms for Scientific Computation*. 2022. – URL <https://arxiv.org/pdf/2201.08309.pdf>
- [Liu u. a. 2021] LIU, Hai-Ling ; WU, Yu-Sen ; WAN, Lin-Chun ; PAN, Shi-Jie ; QIN, Su-Juan ; GAO, Fei ; WEN, Qiao-Yan: Variational quantum algorithm for the Poisson equation. In: *Physical Review A* 104 (2021), August, Nr. 2. – URL <http://dx.doi.org/10.1103/PhysRevA.104.022418>. – ISSN 2469-9934
- [Saha u. a. 2023] SAHA, Kamal K. ; ROBSON, Walter ; HOWINGTON, Connor ; SUH, In-Saeng ; WANG, Zhimin ; NABRZYSKI, Jaroslaw: Advancing Algorithm to Scale and Accurately Solve Quantum Poisson Equation on Near-term Quantum Hardware. (2023)
- [Stokes u. a. 2020] STOKES, James ; IZAAC, Josh ; KILLORAN, Nathan ; CARLEO, Giuseppe: Quantum Natural Gradient. In: *Quantum* 4 (2020), Mai, S. 269. – URL <http://dx.doi.org/10.22331/q-2020-05-25-269>. – ISSN 2521-327X
- [Vazquez u. a. 2022] VAZQUEZ, Almudena C. ; HIPTMAIR, Ralf ; WOERNER, Stefan: Enhancing the Quantum Linear Systems Algorithm Using Richardson Extrapolation. In: *ACM Transactions on Quantum Computing* 3 (2022), Januar, Nr. 1, S. 1–37. – URL <http://dx.doi.org/10.1145/3490631>. – ISSN 2643-6817
- [de Wolf 2023] WOLF, Ronald de: *Quantum Computing: Lecture Notes*. 2023. – URL <https://arxiv.org/pdf/1907.09415.pdf>
- [Zaman u. a. 2023] ZAMAN, Anika ; MORRELL, Hector J. ; WONG, Hiu Y.: A Step-by-Step HHL Algorithm Walkthrough to Enhance Understanding of Critical Quantum Computing Concepts. In: *IEEE Access* 11 (2023), S. 77117–77131. – URL <https://arxiv.org/pdf/2108.09004.pdf>