Projet 2 - BSQ101

Tomographie d'état quantique

Christopher Sicotte

Présenté à
Maxime Dion et Karl Thibault

Département des Sciences
Université de Sherbrooke
20 février 2024

1 Introduction

La tomographie est une technique utilisée dans plusieurs domaines, notamment, en imagerie médicale et en astrophysique. Cette technique consiste à effectuer plusieurs mesures partielles d'une entité pour estimer son tout. La tomographie peut aussi être utilisée en informatique quantique. Effectivement, Dans un système quantique, la tomographie peut être utilisée pour estimer l'état inconnu d'un système quantique.

2 Théorie

2.1 Structure

Pour trouver le vecteur d'état d'un système quantique, on doit d'abord trouver la matrice de densité qui représente le système. Pour ce faire, on doit mesurer le circuit dans la bonne base de calcul plusieurs fois pour estimer la valeur moyenne des différentes chaînes de Pauli. Ces valeurs moyennes vont permettre d'estimer la matrice de densité du système et trouver le vecteur d'état.

2.2 Chaînes de Pauli

Les chaînes de Pauli sont une combinaison tensorielle de chaque opérateur de Pauli. Pour un système à n qubits, nous avons donc, 4^n chaînes de Pauli possible. Les chaînes de Pauli constituent une base de tout observable à n qubits. Elle peut donc être exprimée par une combinaison linéaire des chaînes de Pauli. La valeur moyenne d'une observable peut se représenter en évaluant les valeurs moyennes des chaînes de Pauli la constituant.

$$\langle\psi|\hat{A}|\psi\rangle = \sum_i a_i \langle\psi|\hat{\mathcal{P}}_i|\psi\rangle$$

2.3 Diagonalisation

Pour mesurer les valeurs moyennes des différentes chaînes de Pauli, nous devront les diagonaliser. Effectivement, les mesures d'un système quantique sont effectuées dans la base Z à cause de son architecture. Nous devons donc transformer toutes les chaînes de Pauli non diagonales en fonction d'opérateurs Z et I. Il suffit donc d'appliquer des transformations au circuit avant de mesurer. Pour chaque opérateur associé à un qubit, on applique les transformations suivantes : pour un opérateur $\hat{Y}$, on applique $S^\dagger H$. Pour un opérateur $\hat{X}$, on applique H.

2.4 Mesure et valeur moyenne

Une fois les chaînes Pauli diagonalisées, on peut effectuer plusieurs mesures pour chacune des chaînes. Pour chaque chaîne de Pauli, on prend la valeur du "count" de la mesure divisée par le nombre de mesures effectué. Ce processus va nous donner la valeur moyenne de la chaîne de Pauli. Cette valeur moyenne va être primordiale dans l'approximation de la matrice de densité.

2.5 Matrice de densité et vecteur d'état

Comme mentionné plus haut, un système quantique peut être représenté par une matrice de densité. Cette matrice de densité est une combinaison linéaire des chaînes de Pauli.

$$\hat{\rho} = |\psi\rangle\langle\psi| = \sum_i a_i \hat{\mathcal{P}}_i.$$

Nous n'avons qu'à trouver le coefficient a_i pour trouver la matrice. Il s'avère que ce coefficient est en fait la valeur moyenne associée à la chaîne de Pauli à un facteur de $\frac{1}{2^n}$. Pour trouver le vecteur d'état, il ne reste plus qu'à trouver le vecteur propre associé à la valeur propre la plus grande de la matrice de densité.

3 Implémentation

Le code se divise en 4 fichiers contenant des fonctions utiles à l'algorithme et un fichier principal (main.py) qui sert d'interface utilisateur. À noter que lors de l'explication, j'omettrai les paramètres des fonctions par souci de simplicité et puisqu'ils sont bien écrits dans le code et le diagramme.

3.1 main.py

C'est ce fichier qui sera lancé par l'utilisateur pour lancer l'algorithme.

3.2 State_tomography.py

Ce fichier contient toutes fonctions qui gèrent la tomographie.

state_tomography() : Gère la structure de la tomographie. Appel create_all_pauli(), estimate_expectation_values(), calculate_density_matrix() et calculate_state_vector().

calculate_density_matrix() : Calcule la matrice de densité en multipliant la valeur moyenne des chaînes de Pauli avec la chaîne de Pauli associée.

calculate_state_vector() : Calcule le vecteur d'état en trouvant le vecteur propre associé à la valeur propre la plus grande de la matrice de densité du système.

3.3 Pauli_operations.py

Ce fichier contient toutes fonctions qui gèrent les opérations à utiliser sur les Pauli et chaînes de Pauli utilisés dans la tomographie.

estimate_expectation_values() : Calcule les valeurs moyennes pour chacune des chaînes de Pauli.
Appel diagonalize_pauli_with_circuit(), mesure_pauli_circuit() et diag_pauli_expectation_value().

diagonalize_pauli_with_circuit() : Applique les transformations nécessaires au circuit mystère pour mesurer dans la bonne base de calcul selon la chaîne de Pauli en entrée et diagonalise la Pauli en entrée.
Appel diagonalize_pauli().

mesure_pauli_circuit() : Ajoute une chaîne de Pauli diagonale au circuit mystère et le mesure.

diag_pauli_expectation_value() : Calcule la valeur moyenne d'une chaîne de Pauli diagonale en trouvant sa valeur propre multipliée par son "count" et divisé par le total de "count". Appel bitstring_to_bits().

diagonalize_pauli() : Diagonalise une chaîne de Pauli selon sa représentation zx.

3.4 Utils.py

Ce fichier contient des fonctions accessoires à l'algorithme de tomographie.

execute_job() : Transpile et exécute la liste de circuit à mesurer sur le backend.

bitstring_to_bits() : Convertie des chaines de caractères en tableau de booléens.

create_all_pauli() : Crée les 4^n chaînes de Pauli et les insère dans une PauliList().

create_random_quantum_circuit() : Applique des portes aléatoires sur le circuit.

4 Résultats et Analyse

Les résultats obtenus devraient être une approximation du vecteur d'état du circuit mystère. Effectivement, mes résultats sont concluants. J'ai créé une fonction validate_state_vector() qui permet de déterminer le taux de fidélité du vecteur d'état espéré et du vecteur d'état estimé. Cette fonction donne une valeur entre 0 et 1. Plus la valeur est près de 0 plus les deux vecteurs se rapprochent de l'orthogonalité et plus la valeur de sortie s'approche de 1, les vecteurs se rapprochent de la colinéarité. Voici la fonction de fidélité utilisée:

$$F(\rho, \sigma) = |\langle \psi_\rho | \psi_\sigma \rangle|^2.$$

Voici quelques résultats:

Vecteur d'état espéré	vecteur d'état estimé	Valeur de fidélité
[0, 0, i, 0]	[0.020, -0.001, -0.28 -0.98i, 0.01 - 0.02i]	0.998
[0.707, 0.707, 0, 0]	[-0.798, -0.704, 0.002, 0.01]	0.999
[0, 0.707, 0, 0.707]	[-0.005, -0.698, -0.110, -0.691]	0.999

En analysant ces résultats, on peut voir qu'il semble y avoir une plus grande erreur lorsqu'une porte Y est appliquée et qu'il y a une rotation dans les Y. Mais somme toute, les résultats semblent concluants. De plus, on peut remarquer que la topographie quantique est possible sur un ordinateur quantique grâce à l'architecture de celle-ci. Si on veut parler d'un avantage quantique, je ne sais pas si c'est le cas, car il s'avère que ce problème est directement relié à cette architecture. Pour moi, c'est comme comparer ce qui fait mieux des trous entre une perceuse et un marteau. Effectivement, effectuer un calcul semblable sur un ordinateur classique serait moins performant à cause que ce problème est directement lié à un circuit quantique, ses propriétés et des chaînes de Pauli.