BSQ 110 - Introduction aux sciences quantiques et à leurs applications

Rapport d'évaluation - Premier projet de programmation

Simulation d'un circuit quantique

Guillaume Blouin

1 Rétroaction

1.1 Fonctionnalités

1.1.1 Construction des portes

Toutes les matrices des portes quantiques sont correctes. Par contre, les matrices pour les portes contrôlées sont construites «à la main» sans exploiter les capacités de l'ordinateur. Un commentaire similaire s'applique à la réorganisation de l'ordre des qubits pour les portes concernées.

1.1.2 Construction de circuits

Les matrices représentant les circuits quantiques au complet n'ont pas été construites. Une approche où chaque matrice de chaque section de circuit est appliquée successivement sur l'état quantique a plutôt été utilisée.

1.1.3 Préparation des états initial et final

Aucune fonction qui permet de préparer l'état initial n'a été programmée. L'état final est obtenu par l'application de la matrice représentant le circuit sur l'état initial.

1.1.4 Simulation de la mesure

La simulation de la mesure est effectuée par les méthodes `sample_state_3qubits` et `sample_state_4qubits`. Les implémentations fonctionnent comme attendu, mais auraient pu être beaucoup plus compactes. L'utilisation de nombreux `elseif` auraient pu être évité avec une utilisation judicieuse d'une `list` ou d'un `ndarray`. L'utilisation des lettres grecques à l'intérieur d'un dictionnaire est très encombrant. Cela aurait également permis de définir une seule fonction pour l'échantillonnage d'un état quantique, peu importe le nombre de qubits. Plus sur ça à la section *Encapsulation et réutilisabilité*.

1.2 Qualité du code

1.2.1 Lisibilité

Les noms de variables sont informatifs. Il y a parfois des utilisations non cohérentes de différentes conventions (par exemple, les fonctions `tensorProductOnDoors` et `ry_gate_function`) ou des passages du français à l'anglais. L'énumération de variables (par exemple `circuit_1er_palier`) est à éviter. L'utilisation d'une `list` ou d'un `ndarray` aurait permis d'automatiser la construction des matrices représentant les circuits quantiques. Le code m'apparait inutilement compliqué étant donnée la simplicité des tâches à accomplir.

1.2.2 Encapsulation et réutilisabilité

De nombreuses fonctions et méthodes sont définies. Plusieurs ne sont pas utilisées. Certaines ne font que remplacer une opération déjà existante. Au final, la construction des circuits quantiques est entièrement faite par l'utilisateur, étape par étape, en n'exploitant pas l'encapsulation ou la réutilisabilité. Aucune fonction qui permet de définir des portes contrôlées ou encore de réorganiser l'ordre des qubits n'est implémentée. Les matrices pour les portes contrôlées sont toutes construites «à la main».

Les méthodes pour l'échantillonnage auraient pu être remplacées par une seule méthode plus générale.

1.2.3 Originalité

La présentation des résultats sous la forme d'histogrammes est un ajout intéressant.

1.3 Commentaires généraux

Tu as visiblement de l'expérience en programmation. Ton utilisation de plusieurs fonctionnalités de Python le démontre. Cependant, tu n'as pas vraiment exploiter l'encapsulation et la réutilisabilité, malgré ton expérience. La construction des matrices représentant des circuits quantiques a été faite «à la main» sans utiliser des fonctions qui permettent de simplifier ce travail.

2 Évaluation

L'évaluation est en trois parties. D'abord, les fonctionnalités sont évaluées. L'unique critère est que votre implémentation produisent les résultats attendus et que chaque fonction se comporte correctement dans toutes les situations, même des situations que vous n'avez pas nécessairement rencontrées durant le projet.

Ensuite, la qualité de votre code est évaluée en fonction de sa lisibilité et au niveau de sa structure. Un code lisible est un code qui est d'abord bien présenté avec des noms de variable et de fonctions qui sont descriptifs et informatifs. L'utilisation assidue de convention de notation aide également à la lisibilité. Un code bien structuré permet une utilisation facile et flexible de ses fonctionnalités. Un code bien structuré est également plus lisible.

Finalement, l'originalité de vos implémentations peut vous permettre d'obtenir des points bonus. Les implémentations qui sont différentes de l'approche habituelle, particulièrement efficace ou avec une application beaucoup plus générale que nécessaire peuvent être considérées comme originales. L'originalité ne doit cependant pas se faire au détriment de l'efficacité. Il se peut qu'une implémentation originale nuise à la lisibilité. Cela n'est justifiable que s'il y a un gain considérable en efficacité et devrait alors être accompagné d'explications sous la forme de commentaires et d'un document explicatif.

Fonctionnalités			Commentaires
Circuit 1	Portes quantiques	4/6	Voir 1.1.1
	Circuit quantiques	1/2	Voir 1.1.2
	États quantiques	1/2	Voir 1.1.3
Circuit 2	Portes quantiques	4/6	Voir 1.1.1
	Circuit quantiques	1/2	Voir 1.1.2
	États quantiques	1/2	Voir 1.1.3
Commun	Mesure	8/10	Voir 1.1.4
Total		20/30	

Qualité du code		Commentaires
Lisibilité	2/4	Voir 1.2.1
Structure	2/6	Voir 1.2.2
Total	4/10	

Bonus		Commentaires
Orignalité	1/(2)	
Total	1/(2)	

Note finale : 25/40