Introduction au Quantique
Année 3

TP2

Dans le TD 1 et 2 nous avons vu comment générer un job et comment le simuler. Cela nous donne des mesures sur l'ensemble ou un sous-ensemble de qubits.

Quantum Routines (QRoutine)

Définition

Une ***QRoutine*** peut être vue comme une fonction (méthode) en informatique conventionnelle.
Une ***QRoutine*** est un ensemble d'instructions que nous pouvons appeler.

Dans un circuit, nous pouvons avoir un ensemble de schémas et de portes qui se répètent. Si nous regroupons ces schémas dans une *QRoutine*, nous serons capables d'appelé cette *QRoutine* autant de fois que nous le souhaitons pour constituer le circuit complet.

Exemple de définition d'une QRoutine

```
# générer  l'objet QRoutine

my_routine= QRoutine()

#définir l'ensemble des opérations que nous souhaitons appliquer

my_routine.apply(H, 0)

my_routine.apply(CNOT, 0, 1)
```

Cette *QRoutine* applique deux portes :

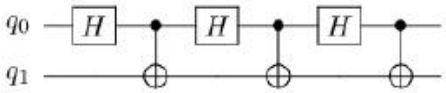
- Une porte d' ***Hadamard*** sur l'indice 0 (il faut spécifier l'indice sur lequel la porte sera appliquée et non le qubit).
- Une porte ***CNOT*** avec comme contrôleur l'indice 0 et comme cible l'indice 1.

Pour appeler la *QRoutine*, il suffit d'écrire :

my_program.apply(my_routine, qubits_reg[0], qubits_reg[1]).

Cela va appliquer la porte ***H*** sur le **qubits_reg[0]** et la porte ***CNOT*** avec comme contrôleur **qubits_reg[0]**, et comme cible **qubits_reg[1].**

Exemple




```
From qat.lang.AQASM import H, CNOT, QRoutine

#Define your routine

my_routine= QRoutine()

my_routine.apply(H, 0)

my_routine.apply(CNOT, 0, 1)

#Use this routine

my_program.apply(my_routine, qubits_reg[0], qubits_reg[1])

my_program.apply(my_routine, qubits_reg[0:2])

my_program.apply(my_routine, qubits_reg)
```

Remarque

Dans l'exemple ci-dessus, nous avons trois appels avec des syntaxes différentes :

1- Dans le premier : nous spécifions individuellement les qubits,
2- Dans le deuxième : nous spécifions une liste de qubits.
3- Dans le troisième : si la taille du registre est de même taille que l'arité de la QRoutine, nous spécifions directement le registre de qubits sans les indices.

QRoutine Walsh Hadamard (Transformée de Hadamard)

Nous allons voir maintenant la possibilité de mettre des *QRoutines* dans des fonctions Python de sorte à pouvoir les paramétrer.

Walsh Hadamard a pour objectif d'appliquer une porte Hadamard par qubit de registre. Ci-dessous un exemple de la **Walsh Hadamard** sur un registre de quatre qubits.

Nous avons une fonction ***Def WALSH_HADAMAR(n)*** avec un paramètre **n,** qui est le nombre de qubits sur lequel nous allons appliquer la *QRoutine* qui va effectuer la Walsh Hadamard.

Pour cela, il faut dans un premier temps générer une QRoutine :

walsh_Hadamard_routine= QRoutine()

et appliquer les portes souhaitées.

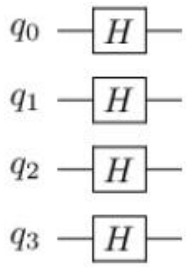
Dans cet exemple, au travers d'une boucle **for,** nous allons parcourir le registre de qubits et nous allons appliquer dans la QRoutine une porte Hadamard par indice :

walsh_Hadamard_routine.apply(H, i).

A la fin de la fonction nous retournons la QRoutine **return walsh_Hadamard_routine**

Une fois la fonction créer nous pouvons l'appliquer avec la fonction **apply(WALSH_HADAMARD(4), qubits_reg)**

```
from qat.lang.AQASM import QRoutine, H

#Define your walsh Hadamard

Def WALSH_HADAMARD(n):

walsh_Hadamard_routine=  QRoutine()

for i in range(n):

  walsh_Hadamard_routine.apply(H, i)

return walsh_Hadamard_routine


#Use this routine

my_program.apply(WALSH_HADAMARD(4), qubits_reg)
```



Quantum Fourier Transform : QFT

En informatique quantique, ***la transformée de Fourier quantique (QFT)*** est une transformation linéaire sur des qubits, et est l'analogie quantique de ***la transformée de Fourier discrète***.

La transformée de Fourier quantique est utilisée dans de nombreux algorithmes quantiques :

- l'**algorithme de Shor** qui permet de factoriser et de calculer le logarithme discret,
- l' **algorithme d'estimation de phase** quantique qui estime les valeurs propres d'un opérateur unitaire
- les algorithmes de problèmes de sous-groupes cachés , etc

La QFT agit sur un vecteur ***d'état quantique***, alors que la transformée de Fourier classique agit sur un ***vecteur (classique).*** Dans les deux cas, ces vecteurs peuvent être écrits sous la forme de listes de nombres complexes.

Dans le cas quantique, Les nombres complexes représentent les **amplitudes de probabilités** des différents résultats obtenus par la mesure.

La QFT peut être calculée efficacement à l'aide d'un ordinateur quantique, en utilisant une décomposition en un produit de matrices unitaires plus simples. A l'aide de cette décomposition, la Transformée de Fourier Discrète sur $\mathbf{2^n}$ amplitudes peut être mises en œuvre sous la forme d'un circuit quantique avec un nombre de Portes d' Hadamard et de portes à déphasage commandé évoluant en $\mathbf{O(n^2)}$, où **n** est le nombre de qubits (le nombre de porte évolue selon une fonction en $\mathbf{n^2}$).

En comparaison, la Transformée de Fourier Discrète classique requiert un nombre de porte évoluant en $\mathbf{O(n2^n)}$, soit exponentiellement supérieur à $\mathbf{O(n^2)}$.

Définition

La QFT est la Transformée de Fourier Discrète classique appliquée au vecteur d'amplitudes d'un état quantique, où l'on considère habituellement des vecteurs de longueur $\mathbf{N = 2^n}$

La Transformée de Fourier classique agit sur un vecteur $(\mathbf{x_0}, \mathbf{x_1}, \ldots . \mathbf{x_n}) \in \mathbb{C}^{\mathbf{N}}$ et le mappe au vecteur $(\mathbf{y_0}, \mathbf{y_1}, \ldots . \mathbf{y_n}) \in \mathbb{C}^{\mathbf{N}}$ selon la formule :

$$y_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \omega_N^{-kn}, \quad k = 0, 1, 2, \ldots, N-1,$$

Où: $\omega_N = e^{\frac{2\pi i}{N}}$ et ω_N^n est une racine N-ième de l'unité [1].

De même,

la QFT agit sur un état quantique : $|x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$

et renvoie un état quantique $\sum_{i=0}^{N-1} y_i |i\rangle$

selon la formule : $$y_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \omega_N^{nk}, \quad k = 0, 1, 2, \ldots, N-1,$$

Étant donné ω_N^n est une rotation de ω_N la QTF **inverse** agit de manière similaire mais avec :

$$x_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k \omega_N^{-nk}, \quad n = 0, 1, 2, \ldots, N-1,$$

Si |x> est un état de base, la QTF peut également être exprimée ainsi

$$\mathrm{QFT} : |x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{xk} |k\rangle.$$

De manière équivalente, la QTF peut être considérée comme une matrice unitaire (ou porte quantique) agissant sur des vecteurs d'état quantiques, où la matrice unitaire F_N est donné par

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

où $\omega = \omega_N$. Par exemple, dans le cas de $N = 4 = 2^2$ et la phases $\omega = i$ la matrice de transformation devient alors

$$F_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

[1] https://fr.wikipedia.org/wiki/Racine_de_l%27unit%C3%A9

Exemple

Considérons la QTF à 3 qubits. Il s'agit de la transformation suivante :

$$\mathrm{QFT} : |x\rangle \mapsto \frac{1}{\sqrt{2^3}} \sum_{k=0}^{2^3-1} \omega_3'^{xk} |k\rangle,$$

où $\omega_3' = \omega_{2^3}$ est une matrice huitième de l'unité satisfaisant

$$\omega_3'^8 = \left(e^{\frac{2\pi i}{2^3}}\right)^8 = 1 \text{ (puisque } N = 2^3 = 8\text{)}.$$

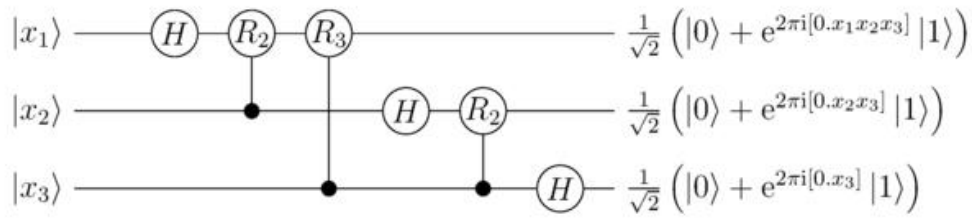
En posant $\omega = \omega_3' = \omega_8$ la représentation matricielle de cette transformation sur 3 qubits est :

$$F_{2^3} = \frac{1}{\sqrt{2^3}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix}.$$

La transformée de Fourier quantique à **n=3** qubits peut être réécrite comme suit :

$$\mathrm{TFQ}(|x_1, x_2, x_3\rangle) = \frac{1}{\sqrt{2^3}} \left(|0\rangle + e^{2\pi i\,[0.x_3]}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\,[0.x_2 x_3]}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\,[0.x_1 x_2 x_3]}|1\rangle\right).$$

Dans le schémas suivant, nous avons le circuit respectif pour **n=3** (l'ordre des qubits de sortie étant inversé par rapport à la QTF à proprement parler):



Comme calculé ci-dessus, le nombre de portes utilisées est $n(n+1)/2$ qui est égal à 6, pour $n = 3$.