# Rapport Hackaton Informatique Quantique

DAHER Sarah, DUVAL Mathilde
PEGUET Lise, SIMEON Milan

- **Exercice 1**

Le but de l'exercice est de creer un circuit qui produit 2 qubits $(x_0, x_1)$ tq $p(x_0, x_1) = \begin{cases} 0 \text{ si } (x_0, x_1) \in \{(0,0), (1,1)\} \\ 1/3 \text{ si } (x_0, x_1) = (0,1) \\ 2/3 \text{ si } (x_0, x_1) = (1,0) \end{cases}$

premièrement, on s'est dit, on doit commencer par mettre une porte X sur un qubit, en effet, la proba que $x_0$ et $x_1$ soient égaux est nulle. On l'a mis au qubit 1 ($q_1$)

pour avoir une proba de $\frac{1}{3}$ pour $(0,1)$, on ajoute une porte $R_y$ avec $2\arccos\left(\frac{1}{\sqrt{3}}\right)$ comme angle sur le qubit 0

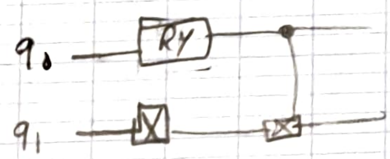
↳ ceci nous amène à $\cos\frac{\theta}{2}|01\rangle + \sin\frac{\theta}{2}|11\rangle$

$$= \frac{1}{\sqrt{3}}|01\rangle + \underbrace{\sqrt{\frac{2}{3}}|11\rangle}$$

↳ à transformer en $|10\rangle$
↳ CNOT

$$= \frac{1}{\sqrt{3}}|01\rangle + \sqrt{\frac{2}{3}}|10\rangle$$

d'où le circuit



## Exercice 2:

Intuition: On cherche $A, B, C$ tq $A \otimes B \otimes C = U$

En commencant par la fin : $\underbrace{\left(\begin{array}{cc|cc} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ \hline 0 & 0 & y & 0 \\ 0 & 0 & 0 & z \end{array}\right)}_{A \otimes B} \otimes \underbrace{\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}}_{C} = U$

On remarque les 4 matrices de la diagonale, $\begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix}, \underline{\begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}}, \underline{\begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}}, \begin{pmatrix} \beta & 0 \\ 0 & \gamma \end{pmatrix}$

On a le même coefficient car la 2ème et 3ème matrice de la diagonale de U sont égales

On obtient donc les égalités suivantes : $x\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix}$, $y\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}$

$z\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} = \begin{pmatrix} \beta & 0 \\ 0 & \gamma \end{pmatrix}$

En remarquant : $\alpha^2 = \beta$, $\alpha\beta = \gamma = \alpha^3$

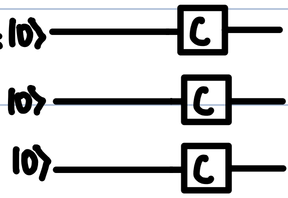
On peut résoudre ces équations pour trouver $x, y, z, a$ et $b$ et on obtient $\begin{cases} x = 1, y = \alpha, z = \beta \\ a = 1, b = \alpha \end{cases}$

Donc $A \otimes B = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ \hline 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & \beta \end{array}\right)$ $C = \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix}$

De la même manière on remarque finalement que $A \otimes B = \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix}$

et finalement $C \otimes C \otimes C = U$

On implémente donc le circuit :



<u>Exo 3:</u>

en appliquant la porte de Hadamard à ce qubit on obtient:

$$
\begin{aligned}
H|\psi\rangle &= \frac{(-1)^{f(0)}}{\sqrt{2}} H|0\rangle + \frac{(-1)^{f(1)\oplus b}}{\sqrt{2}} H|1\rangle \\
&= \frac{(-1)^{f(0)}}{\sqrt{2}} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{(-1)^{f(1)\oplus b}}{\sqrt{2}} \left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle\right) \\
&= \frac{(-1)^{f(0)}}{2}(|0\rangle + |1\rangle) + \frac{(-1)^{f(1)\oplus b}}{2}(|0\rangle - |1\rangle) \\
&= \frac{1}{2}\left(\left[(-1)^{f(0)} + (-1)^{f(1)\oplus b}\right]|0\rangle + \left[(-1)^{f(0)} + (-1)^{f(1)\oplus b+1}|1\rangle\right]\right)
\end{aligned}
$$

D'où par disjonction de cas :

- si m= 0 alors f(0)=f(1) ⊕ b
- si m = 1 alors f(0) = NOT(f(1) ⊕ b)

D'où le tableau:

| b | f(0) | f(1) | m |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

On remarque que b= f(0) ⊕ f(1) ⊕ m : on en déduit qu'avec cette formule, à partir de f(0), f(1) et m, on peut en déduire certainement la valeur de b.

## Exercice 4

on nous a donné la structure d'une map représentant les liens entre les qubits de la machine IBM. Le but est de créer un état de Bell entre les qubits 0 et 26. Cependant, nous n'avons pas de lien direct entre ces 2 qubits.

Notre première approche était d'utiliser des portes swap (après avoir appliqué une porte Hadamard à 0 suivi d'un CNOT contrôlé par 0 sur le qubit 1, nécessaires pour créer l'état de Bell) entre 0 et 1 puis entre 1 et 4. . . , entre 25 et 26. Par contre, le swap consiste en 3 CNOTs ce qui fait monter la profondeur du circuit. On a donc opté pour une soluo moins coûteuse en profondeur

Cette dernière consiste à utiliser 2 CNOTs au lieu de 3 pour propager l'intrication, en effet, on a pas besoin d'échanger $q_i$ et $q_{i+1}$ (suivant l'ordre qui nous ramène à $q_{26}$) mais il suffit juste de transférer $q_i$ à $q_{i+1}$ ce qui se fait en 2 CNOTS

↳ on a ainsi réduit la profondeur du circuit.

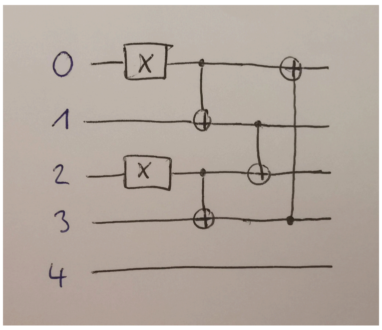
**Exercice 5** :

L’exercice 5 a été réalisé en plusieurs étapes, en partant d’une solution naïve et en l’améliorant au plus possible.

La solution naïve premièrement implémentée était presque celle déjà présente dans le fichier partagé, qui utilisait un appel à ***learn*** sur un circuit simple pour déterminer si une certaine liaison était présente ou manquante. Nous avons utilisé ce script (au maximum) 4 fois sur ces 4 liaisons. Dès que l’on avait déterminé laquelle était manquante, on créait simplement un état de Bell entre les 2 qubits à son extrémité, grâce au script qui était déjà présent dans le fichier partagé.
Cette solution naïve obtenait une fidélité de 1 (on était sûrs de créer l’état de Bell entre les bons qubits), mais faisait entre 1 et 4 appels à ***learn*** (en moyenne 2.5).

Nous avons ensuite proposé une première amélioration, qui consistait à réduire le nombre d’appels à ***learn*** : remarquant que le circuit sur lequel on appelait la fonction ***learn*** n’avait une influence que sur la liaison regardée, nous avons compris que l’on pouvait paralléliser le processus pour tester les 2 liaisons opposées en même temps, avec un seul appel à ***learn***.
Cette solution nous a permis de faire 1 ou 2 appels à ***learn*** (en moyenne 1.5).

Notre réflexion a ensuite été la suivante : dans le circuit qui teste 2 liaisons en même temps avec un appel à ***learn***, l’information de la présence ou non de chacune des 2 liaisons était contenue dans 1 qubit. Nous utilisions donc seulement la mesure de 2 qubits dans la fonction ***missing_link_function***. Nous nous sommes dits que plus d’informations pouvaient être “retenues” dans le circuit lors d’un appel à ***learn***, puisque 5 qubits sont mesurés à chaque appel. Ainsi, nous avons essayé de trouver un circuit dont la mesure permettait de déterminer si l’une des 2 liaisons opposées est manquante, et dans le cas contraire, de déterminer laquelle des 2 autres l’est. A tâtons, nous avons trouvé le circuit ci dessous:

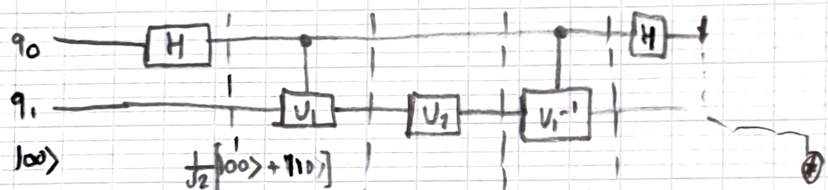



Dans l’hypothèse où les liaisons 0-1 et 2-3 étaient présentes (sinon on le détectait et créait l’état de Bell voulu), les qubits 1 et 3 étaient dans l’état |1}. Nous leur avons donc fait contrôler des CNOT respectivement sur les qubits 2 et 0, pour tester les liaisons 1-2 et 3-0. Grâce à la mesure de 2 et 0, nous pouvions désormais obtenir l’information de la liaison manquante entre les 2 dernières.
Cette dernière solution nous a permis une fidélité maximale, avec un seul appel à ***learn*** à chaque fois. Ne pensant pas qu’il était possible de réduire encore ce nombre d’appels en-dessous de 1, nous avons pensé avoir la solution optimale. Nous nous sommes donc contentés de cette solution.

## Exercice 6

Dans cet exercice, on nous donne 2 portes unitaires $U_1$ et $U_2$, le but étant de savoir si elles commutent ou anti-commutent.
Pour ce faire, nous partons de $|00\rangle$ et on applique une porte $H$ à $q_0$.
Notre but est de mesurer 0 sur ce $q_0$ si $U_1 U_2$ commutent et 1 sinon, il nous faut donc une relation sur $q_0$ sous forme de soustraction qui donne le bon résultat. Ainsi, sur $q_1$, on applique une porte $U_1$ contrôlée par $q_0$ (on l'annulera après), suivi d'une porte $U_2$ puis d'une $U_1$ inversée contrôlée. Enfin, $H$ sur $q_0$ permet d'avoir la soustraction demandée.

**Explication et détails du calcul:**




$\frac{1}{\sqrt{2}}\left[|00\rangle + |10\rangle\right]$

$\frac{1}{\sqrt{2}}\left[|00\rangle + |1U_1(0)\rangle\right]$

⚠ $U_2$ appliqué sur $q_1$ non contrôlé pour avoir le m̂ $q_1$ dans la soustrac°

$\frac{1}{\sqrt{2}}\left[|0U_2(0)\rangle + |1\, U_2 U_1(0)\rangle\right]$

$\frac{1}{\sqrt{2}}\left[|0U_2(0)\rangle + |1\, U_1^{-1} U_2 U_1(0)\rangle\right]$

- si $U_1$ et $U_2$ commutent
① ↳ $U_2 U_1 = U_1 U_2$
↳ $U_1^{-1} U_2 U_1 = U_1^{-1} U_1 U_2 = U_2$

② • sinon : $U_1^{-1} U_2 U_1 = -U_2$

(*) ①: après $H$ sur $q_0$ on a : $\frac{1}{2}|0U_2(0)\rangle + \frac{1}{2}|1U_2(0)\rangle + \frac{1}{2}|0U_2(0)\rangle - \frac{1}{2}|1U_2(0)\rangle$

$= |0U_2(0)\rangle$

↳ on mesure bien 0 sur $q_0$

② après $H$ sur $q_0$ on a : $\frac{1}{2}|0U_2(0)\rangle + \frac{1}{2}|1U_2(0)\rangle - \left(\frac{1}{2}|0U_2(0)\rangle - \frac{1}{2}|1U_2(0)\rangle\right)$

$= |1U_2(0)\rangle$

↳ on mesure bien 1 sur $q_0$

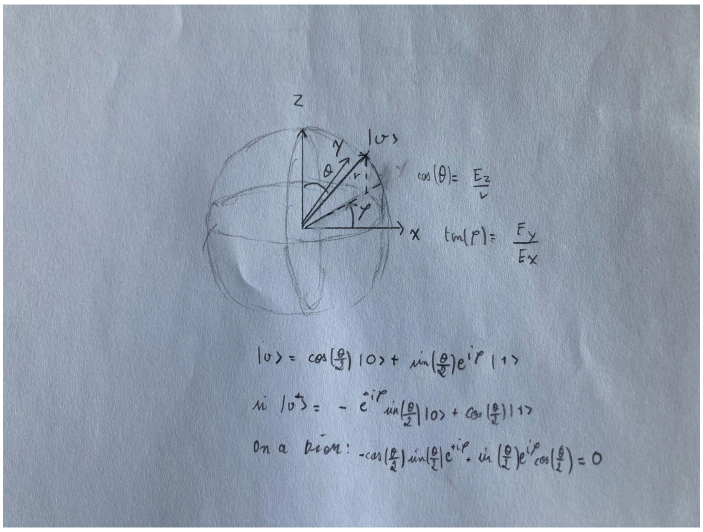
Exo 7:

On calcule d'abord dans le circuit 1 les composants dans la base Z. Dans le circuit 2 , on applique une porte de Hadamard sur chacun des qubits , en faisant des mesures cela revient à calculer les composants dans la base X. Dans le circuit 3, on applique 6 fois un déphasage de pi/4 à chacun des qubits, ce qui revient à multiplier par la matrice :

$$\begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$$

Puis on multiplie encore chacun des qubits par une porte de Hadamard : faire des mesures revient à calculer les composants dans la base Y.

D'après le schéma suivant:




On obtient alpha, bêta, gamma et delta à partir des formules ci-dessus, en calculant EX, EY et EZ comme espérance lorsque le premier bit (celui de droite) vaut 0.

**Exercice 8** :

Pour cet exercice, le premier objectif a été de maximiser q, c'est-à-dire de ne jamais se tromper. La première solution a pour cela été de mesurer le qubit dans la base computationnelle. Si l'on obtenait l'état |1}, on était sûr que le qubit était initialement dans l'état |+}, et on en déduisait le code. Dans le cas contraire, puisqu'on était pas sûr, on redemandait le message à notre ami.
Cette solution, bien qu'obtenant théoriquement q=0, pose problème dans le cas où le code est 0 : dans ce cas, on reçoit l'état 0, et on ne saura jamais le code, donc on ouvrira jamais le coffre. On avait donc une probabilité très élevée (d'ailleurs, dans la pratique et de manière inexplicable, on obtenait p = 1 en testant, et q était alors égal à 0/0, ce qui créait un Nan).

Nous avons donc essayé de garder une stratégie similaire pour garder q = 1, tout en diminuant p, et en ouvrant le coffre avec une espérance de temps finie, dans tous les cas.
Pour cela, nous voulions "inverser" la mesure du qubit, c'est-à-dire être soit sûr que le mot de passe est 0, soit ne rien pouvoir conclure.
Pour cela nous avons vu qu'une porte Y(pi/4) sur le qubit le faisait passer de |0} à |+} ou de |+} à [1}. Ainsi, cette fois, c'est en obtenant l'état |0} à la mesure que l'on était sûr que le mot de passe serait 0, et si l'on obtenait |1} on ne pouvait en tirer aucune conclusion.
Il nous fallait maintenant alterner ces 2 types de mesures pour tester alternativement si le code est 1 ou 0.
Ne pouvant pas garder de mémoire dans la fonction f (qui était appelée avec le seul argument outcome), nous avons gardé l'information de quelle mesure est choisie dans le circuit directement, à l'aide d'un deuxième qubit. Un booléen tiré au hasard nous permettait de choisir si l'on faisait une mesure ou l'autre dans la construction du circuit quantique, et de stocker ce booléen dans le deuxième qubit. Ainsi, dans la fonction f, on pouvait accéder au booléen et savoir quelle circuit avait été implémenté, et, grâce à la mesure du premier qubit, déterminer si l'on connaissait le mot de passe ou s'il fallait redemander un message.

Celle solution est celle que nous avons proposée. Elle a permis d'obtenir q=0, mais on avait à nouveau p=1, sans pouvoir se l'expliquer (la fonction f était censée, dans un cas sur 4 et avec une probabilité non nulle, renvoyer 0 ou 1, et non 2 dans tous les cas comme p=1 le laisse entendre).

__**Exercice 9**__ :

Notre première stratégie a été d’utiliser les 6 000 points pour déterminer à quel point le point de départ de notre sauterelle était “décalé” sur les axes x, y et z.
Nous avons donc premièrement créé 3 circuits qui faisaient une simple mesure sur les bases des axes x, y et z respectivement.
A l’analyse, nous faisions la moyenne (à l’aide des nombres complexes) de tous les états obtenus (qui étaient uniquement |0}, |1}, |+}, |-}, |+i}, |-i}, puisqu’on mesurait dans ces bases), on normalisait le nombre complexe obtenu et on renvoyait le qubit correspondant.

Cependant, cette solution ne nous permettait d’obtenir qu’environ 0.72 de score.

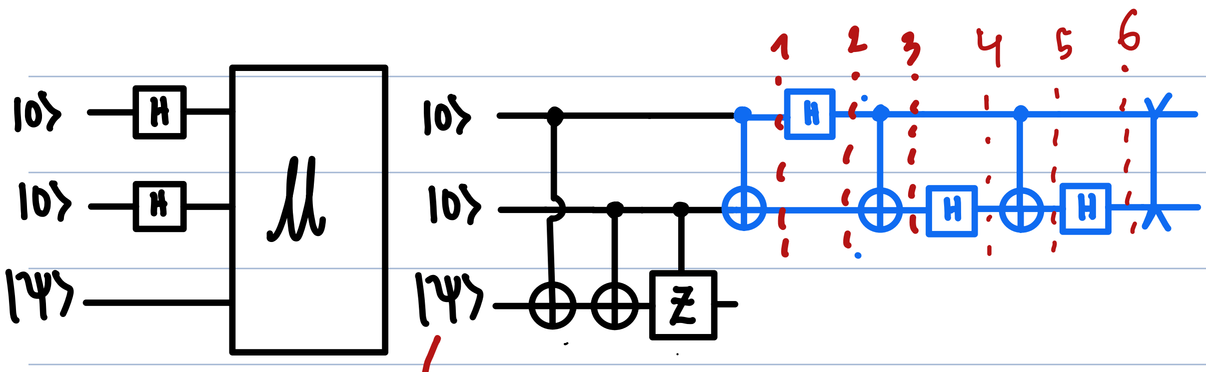
Pensant que mesurer dans 3 bases n’était pas suffisant, nous avons alors eu l’idée de mesurer dans beaucoup de bases. Pour cela, nous créions une distribution de qubits, et les circuits qui mesuraient dans ces bases associées. Nous aurions préféré généré des qubits aléatoires pour faire des mesures dans des bases aléatoires, mais nous ne pouvions pas donner l’information de ces qubits aléatoires à la fonction ***guess_grassehopper***.
L’analyse était la même : nous faisions la moyenne de tous les qubits obtenus comme mesures, puis normalisions et renvoyions le résultat.
Nous avons alors testé différentes valeurs de N et M, qui étaient la manière dont était faite notre répartition (N était le nombre d’angles ***theta*** différents, M le nombre d’angles ***phi*** différents).

Les performances variaient peu selon ces valeurs : nous obtenions environ 0.82 à chaque fois.

Nous avons dernièrement essayé de donner plus de “poids” à des bases dans lesquelles beaucoup plus de mesures avaient donné un état que son état complémentaire, en ajoutant une puissance 100 au coefficient de chaque qubit dans la moyenne.

Cela n’a pas drastiquement augmenté les performances. Cette solution était celle soumise.

# Exercice 10 :




$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

## Analyse de la situation :

Avant la porte $U$ on a l'état : $|+\rangle_0 \otimes |+\rangle_1 \otimes (\alpha|0\rangle_2 + \beta|1\rangle_2)$

Après la porte $U$ : $\frac{\alpha}{2}\left[|000\rangle + |101\rangle - |011\rangle + |110\rangle\right)$

$+ \frac{\beta}{2}\left[|001\rangle + |100\rangle + |010\rangle - |111\rangle\right)$

En particulier si on s'intéresse aux qubits 0 et 1 : $\frac{1}{2}\left[(\alpha+\beta)|00\rangle + (\alpha+\beta)|11\rangle + (\beta-\alpha)|01\rangle + (\alpha-\beta)|10\rangle\right]$

Le but est de faire apparaître le produit tensoriel entre $|\psi\rangle$ et un autre vecteur

On procède d'abord à tâton pour voir ce que chaque gate nous permet de changer d'intéressant sur notre état.

1. $\frac{1}{2}\left[(\alpha+\beta)|00\rangle + (\alpha+\beta)|10\rangle + (\beta-\alpha)|01\rangle + (\alpha-\beta)|11\rangle\right]$

2. $\frac{1}{\sqrt{2}}\left[\alpha|00\rangle + \beta|10\rangle + \beta|01\rangle - \alpha|11\rangle\right] = \frac{1}{\sqrt{2}}\left(|0\rangle \otimes \underbrace{(\alpha|0\rangle + \beta|1\rangle)}_{✓} + |1\rangle \otimes \underbrace{(\beta|0\rangle - \alpha|1\rangle)}_{✗}\right]$

↳ On y est presque

Pour essayer d'inverser les 2 coordonnées ici on ressaie le processus en placant H sur le $2^{ème}$ qubit.

3. $\frac{1}{\sqrt{2}}\left[\alpha|00\rangle+\beta|11\rangle+\beta|01\rangle-\alpha|10\rangle\right]$

4. $\frac{1}{2}\left[(\alpha+\beta)|00\rangle-(\beta-\alpha)|01\rangle-(\alpha-\beta)|10\rangle-(\beta+\alpha)|11\rangle\right]$

On se retrouve maintenant dans l'état initial à quelques signes près, c'est pas gagné mais si on réitère l'operation on peut avoir le resultat voulu grâce à ces nouveaux signes.

5. $\frac{1}{2}\left[(\alpha+\beta)|00\rangle+(\alpha-\beta)|01\rangle+(\beta-\alpha)|11\rangle-(\beta+\alpha)|10\rangle\right]$

6. $\frac{1}{\sqrt{2}}\left[\alpha|00\rangle+\beta|01\rangle-\beta|11\rangle-\alpha|10\rangle\right]=\frac{1}{\sqrt{2}}\left[|0\rangle\otimes(\alpha|0\rangle+\beta|1\rangle)-|1\rangle\otimes(\alpha|0\rangle+\beta|1\rangle)\right]$

$$=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)\otimes(\alpha|0\rangle+\beta|1\rangle)$$

$$=|-\rangle\otimes|\Psi\rangle$$

Enfin on obtient le resultat voulu. Seulement, on retrouve $|\Psi\rangle$ sur le deuxième bit et non pas sur le $1^{er}$. Pour y remedier, on implémente une SWAP gate à l'aide de 3 CNOTs.