# TP 2 - Initiation à qiskit

Informatique quantique pour la recherche opérationnelle, S5.

2024

Tout au long du TP, les questions à faire sur papier sont précédées du symbole †.

Compétences acquises à la fin du TP :

- Utilisation de qiskit
- Compréhension du fonctionnement d'un simulateur quantique

Ce TP a pour but d'initier les élèves à l'utilisation basique de qiskit.
Il faut bien noter que qiskit est en constante évolution, les versions de la bibliothèque changent très régulièrement et il n'est pas rare de voir des paquets devenir dépréciés voire disparaître. Dans le cas où il serait nécessaire d'installer un paquet qui n'a pas été prévu dans l'installation, vous pouvez utiliser la commande suivante. Attention à ne pas en abuser toutefois pour éviter de surcharger les serveurs de l'école avec de multiple copies de paquets.

```
pip -u install nom_du_paquet
```

**Exercice 1 — *Utilisation basique de qiskit***

1. Le circuit suivant crée le circuit qui fabrique l'état $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ de Bell.

```
from qiskit import QuantumCircuit

# Circuit avec 2 qbit et 2 bits pour la mesure.
qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0, 1)

print(qc)  # ou qc.draw(output='text')
```

2. On peut afficher le circuit en console.

3. On peut maintenant afficher le qbit obtenu en sortie.

```
from qiskit.quantum_info import Statevector
print(Statevector(qc))
```

4. On peut également afficher la matrice correspondant au circuit

```
from qiskit.quantum_info import Operator
print(Operator.from_circuit(qc))
```

5. On peut enfin afficher la distribution de mesure en sortie du circuit avec un `Sampler`. Un sampler exécute les mesures un certain nombre de fois puis affiche la distribution.

```
qc.measure([0,1], [0,1])

from qiskit.primitives import Sampler
sampler = Sampler()
```

```
job = sampler.run(circuit, shots=1000)

result = job.result()
print(result.quasi_dists[0].binary_probabilities())
```

**Exercice 2** — ***Mesure***

Effectuer un code qui, connaissant un qbit $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ calcule une estimation de l'angle $\theta$ sur la sphère de Bloch de ce qbit.

**Exercice 3** — ***Utilisation d'une vraie machine quantique***

**L'exercice suivant nécessite un compte IBM, avec tout ce que cela implique au niveau de la diffusion de vos données personnelles. Il est donc bien entendu facultatif. Vous pouvez à la place juste lire l'exercice et le tutoriel IBM sans exécuter le code.**

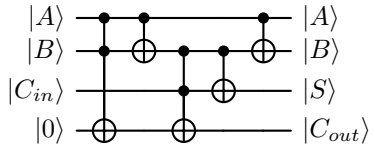
1. Connectez vous sur `https://quantum.ibm.com/`
2. Créez un `IBMid`, utilisez votre adresse email institutionnelle.
3. Vous arrivez ensuite sur la plateforme qui vous permet de gérer les calculs effectués sur les machines d'IBM. Vous trouverez sur la page `dashboard` votre **Clef d'API**. La page `Compute resources` vous indique les machines actuellement disponibles. Enfin la page `Workloads` vous permet de gérer vos *jobs* c'est à dire les calculs que vous avez demandé aux machines d'exécuter.
4. Vous pouvez maintenant programmer un circuit pour l'exécuter sur la machine. On va utiliser un Sampler.

```
...
Creation d'un circuit qc
...

from qiskit_ibm_runtime import QiskitRuntimeService
service = QiskitRuntimeService(channel="ibm_quantum", token= API")

# Trouver une machine libre (la moins utilisee ici)
backend = service.least_busy(simulator=False, operational=True)

# Transformer le circuit en un circuit equivalent qui sera accepte par la mach
pm = generate_preset_pass_manager(backend=backend, optimization_level=1)
isa_circuit = pm.run(qc)

# Afficher le circuit
isa_circuit.draw("mpl", idle_wires=False)

# Creer un Sampler
from qiskit_ibm_runtime import SamplerV2 as Sampler
sampler = Sampler(mode=backend)

# Lancer un job
job = sampler.run([(isa_circuit, param_values)])
print(f">>> Job ID: {job.job_id()}")
print(f">>> Job Status: {job.status()}")

# Attendre le resultat (cela peut prendre du temps)
result = job.result()
```

```
# Afficher les probabilites
print(result.quasi_dists[0].binary_probabilities())
```

**Exercice 4 — *Additionneur***

Dans un précédent TP, on a vu le circuit de l'additionneur suivant:




Produisez avec qiskit un programme qui connaissant un entier $n$, produit un circuit qui effectue l'addition de deux entiers sur $n$ bits. Testez le ensuite en produisant un circuit qui effectue en parallèle l'addition de tous les nombres compris entre 0 et $2^n - 1$. On effectuera un grand nombre de mesures pour afficher une estimation de la distribution des résultats.

**Exercice 5 — *BB84***

Le protocole BB84 permet d'échanger une clef secrète entre deux personnes, Albert et Bonnie. Les deux disposent respectivement d'un canal quantique et d'un canal classique pour échanger des informations.

- Albert commence par générer 2 listes de $N$ bits aléatoires $k$ et $b$.
- Pour chaque paire de bits $(k_i, b_i)$, Albert génère un qbit avec le tableau suivant:

| $k_i$ \ $b_i$ | 0 | 1 |
|---|---|---|
| 0 | $\|0\rangle$ | $\|+\rangle$ |
| 1 | $\|1\rangle$ | $\|-\rangle$ |

  Chacun de ces qbit est envoyé à Bonnie via le canal quantique.

- Bonnie, indépendamment, génère une liste de $N$ bits aléatoires $b'$. Pour chaque qbit $|\varphi_i\rangle$ envoyé par Albert et chaque bit $b'_i$ de $b'$, Bonnie effectue une mesure de $|\varphi_i\rangle$.
  - Si $b'_i = 0$ alors Bonnie mesure $|\varphi_i\rangle$ dans la base $(|0\rangle, |1\rangle)$.
  - Si $b'_i = 1$ alors Bonnie mesure $|\varphi_i\rangle$ dans la base $(|+\rangle, |-\rangle)$.

  *Pour rappel, pour mesurer dans une base $(|e_0\rangle, |e_1\rangle)$, il faut trouver la porte quantique qui transforme $(|e_0\rangle, |e_1\rangle)$ en $(|0\rangle, |1\rangle)$ puis effectuer une mesure standard.*

  Pour chaque mesure, Bonnie obtient un bit $k'_i$ et peut ainsi fabriquer une liste de bits $k'$.

- Après avoir effectué les mesures (pas avant), Albert et Bonnie communiquent sur le canal classique les bits $b_i$ et $b'_i$. La clef secrète utilisée sera la liste $S = [k_i | b_i = b'_i]$.

1. † Montrer que $k_i = k'_i$ si $b_i = b'_i$.
2. Ecrire une fonction $a_1$ qui, connaissant $k$ et $b$ produit la liste des qbits que Albert envoie à Bonnie.
3. Ecrire une fonction $b_1$ qui, connaissant une liste de qbits et $b'$ produit la liste $k'$ que mesure Bonnie.
4. Ecrire une fonction $ab_2$ qui connaissant $k, b$ et $b'$ produit $S$.

5. Mettez vous en binôme avec une autre personne. L'un jouera le rôle d'Albert et l'autre de Bonnie. Albert utilise $a_1$ et envoie (par mail par exemple) la liste des qbits à Bonnie avec le format suivant : tous les qbit sont sur la première ligne du fichier, représentés par les caractères `0`, `1`, `+` et `-`. Pour jouer le jeu, Bonnie ne doit pas ouvrir le fichier avant la fin du protocole. Bonnie utilise ensuite la fonction $b_1$ pour produire $k'$. Albert envoie ensuite par email un fichier contennat $b$ et Bonnie envoie un fichier contenant $b'$. Utilisez la fonction $ab_2$ pour produire la clef secrète $S$. Vérifiez que vous avez bien la même clef.

6. † En temps normal, Albert et Bonnie ne vérifie pas la clef secrète (sinon le protocole n'a aucun intérêt, Albert pourrait envoyer la clef directement à Bonnie). Par contre, ils peuvent sacrifier 20 bits choisis aléatoirement dans $S$ pour détecter une attaque. Pour cela, ils vérifient par un canal classique que ces 20 bits sont identiques. S'ils sont différents, ils jettent la clef et sinon ils la conservent. Les bits ne utilisés ici sont ensuite supprimés de $S$.

   Que se passerait-il si Claude, un hacker terrifiant, lisait le canal quantique pour intercepter les qbits qu'Albert envoie à Bonnie? Montrer que si Albert et Bonnie ont alors 99.7% de chances de détecter l'attaque de Claude.

**Exercice 6 — *Observable et estimateur***

Qiskit propose d'utiliser des `Estimator` en plus des `Sampler` pour effecter des mesures. Un estimateur vous permet de mesurer une valeur particulière (et qui dépend de ce que vous souhaitez mesurer). Cette mesure particulière est donnée par ce qu'on appelle un *Observable*. Un observable est une matrice $O$, qui n'est pas nécessairement unitaire et qui renvoie un résultat qui a du sens physiquement.

Lorsque vous avez caractérisé un observable $O$ et que vous souhaitez observer un qbit $|\phi\rangle$ au travers de $O$, la valeur que vous souhaitez en sortie est $O_\phi = \langle\phi| O |\phi\rangle$. Cependant, premièrement, puisque $O$ n'est pas unitaire, vous ne pouvez pas créer de circuit correspondant à $O$ directement, deuxièmement un tel circuit ne vous donnerait pas $O_\phi$ mais la distribution de mesures de $O|\phi\rangle$. Le but de cet exercice est de produire deux circuits, un qui calcule $O_\phi$ à la main et et un autre en utilisant qiskit.

Dans cet exercice, on va supposer que $O$ est le produit $X \otimes Y \otimes Z$ (il y a donc 3 qbits dans le circuit et $O$ est unitaire).

1. Créer un circuit `qc` avec 3 qbits de votre choix. Identifiez le qbit $|\phi\rangle$ obtenu en sortie du circuit.

2. Utiliser un estimateur pour mesurer le qbit. qiskit permet facilement de créer l'observable à partir des portes qui consistuent $O$.

```
from qiskit.quantum_info import SparsePauliOp
# Attention l'ordre des portes est inverse ci-dessous.
observable = SparsePauliOp("ZYX")

estimator = Estimator()
estimator.options.default_shots = 5000

job = estimator.run([qc, [observable]])
result = job.result()[0]
print(f"Expectation values: {result.data.evs}")
```

3. Vérifier que le résultat obtenu à la question précédente est égal à $O_\phi$. *Pour cela, vous pouvez calculer la matrice $O$ à la main, ou construire dans qiskit le circuit correspondant et demander la matrice unitaire associée. Calculez ensuite le produit $\langle\phi| O |\phi\rangle$.*

4. † Soit $P = H \otimes (HS^\dagger) \otimes Id$ où $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$. Montrer que $D = POP^{-1}$ est une matrice diagonale dont les coefficients sont tous 1 et $-1$.

5. † Soit $x \in \{0, 1, ..., 7\}$ et $\lambda_x$ la valeur propre associée à $x$ pour la matrice $D$ (donc le $x$-ième élément de la diagonale de $D$). Montrer que $O_\phi = \sum_{i=0}^{7} \lambda_x p_x$ où $p_x$ est la probabilité de mesurer $x$ dans à partir de l'état $P|\phi\rangle$.

6. En réalisant de nombreuses mesures de $P|\phi\rangle$, estimez $O_\phi$, vérifiez que le résultat est cohérent avec ce qu'a proposé qiskit.

Vous pouvez recommencer l'expérience avec un autre observable $O = A_1 \otimes A_2 \otimes \cdots \otimes A_n$ où $A_i \in [X, Y, Z, Id]$.

- si $A_i = X$, il faut mettre une porte $H$ dans $P$.
- si $A_i = Y$, il faut mettre une porte $HS^\dagger$ dans $P$.
- si $A_i = Z$ ou $Id$, il faut mettre une porte $Id$ dans $P$.

Il faut ensuite calculer $D$ et les valeurs propres sur la diagonales de $D$. On peut ensuite estimer $O_\phi$ à l'aide de $P|\phi\rangle$ et des valeurs propres.

Notez enfin que si $O$ n'est pas unitaire, on peut toujours l'écrire comme une somme de produits $A_1 \otimes A_2 \otimes \cdots \otimes A_n$ où $A_i \in [X, Y, Z, Id]$, on peut donc appliquer la techique précédente (autant de fois qu'il y a de produits dans la somme).

**Exercice 7 — *Réécrire qiskit***

Produire un simulateur de machine quantique sans utiliser qiskit ou une autre bibliothèque d'informatique quantique.

Le simulateur devra être capable:

- de construire un circuit contenant des portes $H$ et $CNOT$.
- de donner en entrée du circuit un qbit quelconque
- de décrire à chaque étape du circuit la valeur du qbit à cet emplacement
- de donner une estimation de la distribution des mesures du qbit de sortie.