
BSQ 101 - Projets intégrateurs en programmation quantique

Mesures et observables

Application à la tomographie d'états quantiques

À remettre 2 semaines après la présentation du projet

1 Objectifs du projet

Dans ce projet, vous aurez à déterminer l'état d'un système de quelques qubits en effectuant une tomographie d'état quantique. Pour ce faire, vous aurez à

1. Vous familiariser avec les concepts d'observable quantique et de chaines de Pauli
2. Vous familiariser avec la mesure d'observables sur un ordinateur quantique d'abord pour un seul qubit et pour plusieurs qubits
3. Implémenter un estimateur qui permet d'estimer la valeur moyenne d'observables quantiques exprimées comme des chaines de Pauli
4. Tester votre implémentation en estimant des valeurs moyennes de différentes chaines de Pauli pour différents états quantiques
5. Implémenter un tomographe quantique qui, étant donné un circuit quantique qui prépare un état quantique à n qubits, construit une série de circuits de mesure et utilise les résultats pour reconstruire une estimation du vecteur d'état
6. Tester votre implémentation en effectuant la tomographie de divers états quantiques préparer par différents circuits quantiques

Remarque

La tomographie complète d'un état quantique est une tâche qu'on évite généralement de faire, car elle est exponentiellement coûteuse. Il s'agit néanmoins d'un exercice académique très pertinent.

1.1 Premiers pas

Pour démarrer ce projet, il est fortement suggéré de lire attentivement les chapitres 4 et 5 des [notes de cours](#) qui portent sur les mesures, les observables et la matrice densité. Ensuite, les sections 4 et 5 du présent document offrent des notions complémentaires essentielles à ce projet. Ensuite, consultez la section 3 pour prendre connaissance des indications qui vous aideront à débuter votre code.

Nous utiliserons GitHub ainsi et GitHub Classroom pour ce projet. Suivez les étapes suivantes pour démarrer votre projet :

1. Créer un compte GitHub si vous n'en avez pas déjà un.
2. Suivez ce [lien](#) pour créer votre équipe. Ce projet doit se faire en équipe de 2 ou 3 personnes. Votre nom d'équipe devrait avoir le format suivant `BSQ110_2025_Equipe_Y`.
3. Accéder au *repo*.
4. Consultez la structure déjà présente et familiarisez-vous avec les fonctions à compléter.
5. Créez une branche pour le développement (ne travaillez pas directement dans la branche `main`).
6. Écrivez vos noms dans le fichier `README.md`, surtout si votre identifiant GitHub ne permet pas de vous identifier.
7. Commencez à coder !

Remarque

Étant donné les objectifs d'apprentissage du cours BSQ 110, l'utilisation d'outils comme ChatGPT ou Microsoft Copilot sont fortement déconseillés.

2 Évaluation

Votre travail sera évalué en lien avec les aspects suivants :

Fonctionnalités (20 pts) Votre travail sera d'abord évalué sur les fonctionnalités que vous avez implémentées. Les points seront attribués si votre code permet d'effectuer les tâches requises et s'il permet de solutionner le problème en utilisant l'approche proposée.

La section 3 donne quelques indications spécifiques à ce projet. Celles-ci sont fournies d'abord pour vous aider à structurer et démarrer votre travail, mais aussi pour servir de point d'ancrage afin que les solutions des différentes équipes aient quelques points communs.

Qualité du code (20 pts) Votre code sera également évalué sur sa lisibilité et sa structure en regard aux lignes directrices du *code propre*. Portez une attention particulière à définir des fonctions dont le rôle est simple et clair et dont le nom décrit bien leur action. L'ajout de commentaires explicatifs devrait alors être superflu. L'utilisation des *type hints* et des *doc strings* est fortement encouragé.

Structure du dépôt (5 pts) Vous devez utiliser GitHub pour gérer votre code. À la remise, votre dépôt GitHub doit être proprement structuré et ne doit pas contenir de fichiers inutiles.

Contribution au projet (10 pts) Des points seront attribués sur une base individuelle en fonction de votre contribution au dépôt basé sur l'historique du dépôt. Aucune contribution au dépôt vaut 0 pts, une contribution régulière et honnête vaut 10 pts. Portez une attention particulière pour démontrer votre contribution au travail d'équipe.

À la fin du projet, vous devrez évaluer votre contribution et celle des membres de votre équipe au travail d'équipe. Cette évaluation servira de guide pour s'assurer que tous les membres d'une équipe ont contribué significativement. Ces évaluations pourraient avoir un impact sur votre note individuelle pour votre contribution au cours. Dans tous les cas, il est préférable d'aborder les problèmes de travail en équipe en cours de projet et non lorsque celui-ci est terminé.

2.1 Installation

Dans ce projet, vous allez construire un module python `state_tomography`. Lorsqu'on développe un module de la sorte, il est très utile de l'installer dans votre environnement python pour simplifier l'importation et l'exécution des tests. Nous allons utiliser le module `flit` pour la création du module. Il existe d'autres gestionnaires de module et vous êtes libre d'en utiliser un autre si vous le désirez. Si vous désirez utiliser `flit` vous devez l'installer. Assurez-vous d'être dans l'environnement dans lequel vous développer `state_tomography` et utiliser la commande suivante.

```
pip install flit
```

Pour initialiser votre module, naviguer jusqu'au dossier racine du module (là où se trouve le fichier `README.md`) et exécuter la commande suivante.

```
flit init
```

Répondez aux questions qui vous seront posées. Il n'est pas nécessaire de spécifier une licence si vous ne distribuez pas votre code. Cela créera un fichier `.toml` dans votre dossier racine.

Normalement, l'installation d'un module python dans un environnement implique de copier les fichiers sources dans cet environnement. Lorsqu'on développe, il est alors nécessaire de réinstaller le module après chaque modification. Pour éviter cela, on peut installer le module, toujours à partir du dossier racine du module, avec la commande suivante.

```
flit install --symlink
```

Plutôt que de copier les fichiers de votre module dans l'environnement, `flit` créer un lien symbolique vers le dossier racine de votre projet. Ainsi, chaque modification sera prise en compte sans devoir réinstaller le module.

Par contre, il sera quand même nécessaire de réimporter le module après chaque modification. Si vous travaillez en mode interactif, cela nécessite de redémarrer le Kernel.

3 Quelques indications

D'abord, l'implémentation doit se faire en Python. Ensuite, la création et l'exécution des circuits quantiques doivent se faire avec Qiskit. Référez-vous à l'introduction à Qiskit, ou aux tests déjà implémentés dans le projet, pour revoir comment simuler l'exécution de circuits quantiques.

3.1 Chaines de Pauli

Pour créer et manipuler des chaines de Pauli, nous utiliserons le sous-module `quantum_info` de Qiskit. En particulier, nous utiliserons les classes `Pauli` et `PauliList` qui définissent respectivement, une chaîne de Pauli et une liste de chaines de Pauli.

```
1 from qiskit.quantum_info import Pauli, PauliList
```

On peut initialiser une chaîne de Pauli de plusieurs manières. D'abord, en la spécifiant comme une chaîne de caractères.

```
1 pauli_1 = Pauli("ZIXY")
```

On peut également fournir les vecteurs de composantes binaires (voir section 4.1) qui défisent la forme zx de la chaîne de Pauli.

```
1 zbits = [0, 0, 1, 1]
2 xbits = [0, 1, 1, 0]
3 pauli_2 = Pauli((zbits, xbits))
```

Dans tous les cas, on peut accéder à ces vecteurs comme les attributs `z` et `x` de la chaîne de Pauli.

```
1 print(pauli_1.z)
2 print(pauli_1.x)
3 print(pauli_2.z)
4 print(pauli_2.x)
```

```
[ True False False True]
[ True True False False]
[ False False True True]
[ False True True False]
```

On peut également définir une liste de chaines de Pauli de plusieurs manières différentes. D'abord, on peut simplement fournir une liste de `Pauli`.

```
1 paulis = PauliList([pauli_1, pauli_2])
```

On peut aussi fournir celles-ci comme une liste de chaînes de caractères.

```
1 paulis = PauliList(["ZIXY", "ZYXI"])
```

Finalement, on peut utiliser la méthode `from_symplectic` et fournir directement les nombres binaires qui définissent les différentes chaines de Pauli.

```
1 zbits = [
2     [1, 0, 0, 1],
3     [0, 0, 1, 1]
4 ]
5 xbits = [
```

```

6     [1, 1, 0, 0],
7     [0, 1, 1, 0]
8 ]
9 paulis = PauliList.from_symplectic(zbits,xbits)

```

3.2 Fonctions à implémenter

Vous aurez à implémenter plusieurs fonctions dans le cadre de ce projet. Par souci d'uniformité, votre implémentation devrait au moins faire intervenir les fonctions décrites ici. Ne vous limitez cependant pas celles-ci. Ces fonctions peuvent faire intervenir d'autres fonctions que vous jugez pertinentes.

L'utilisation du typage nécessite que les variables de types soient accessibles. Pour cela, on aura à importer certains éléments.

```

1 from typing import Tuple, List
2 from numpy.typing import NDArray
3 from qiskit.primitives import BaseSamplerV2

```

Nous verrons comment les utiliser dans les exemples.

Conversion d'état de base Lorsqu'on exécute un circuit quantique, le résultat obtenu est sous la forme de comptes (`counts`), c'est-à-dire un dictionnaire de clés (`keys`) et de valeurs (`value`). Les clés sont sous la forme de chaînes de caractères de "`0`" et de "`1`", comme par exemple "`0111`".

Pour ce projet, en particulier pour le calcul des valeurs propres des chaînes de Pauli diagonales sur les états de base, il sera utile de convertir ces chaînes de caractères en tableau de booléens. Vous devrez donc implémenter la fonction suivante.

```

1 def bitstring_to_bits(bit_string: str) -> NDArray[np.bool_]:
2     bits = ...
3     return bits

```

Le typage utilisé ici spécifie que la fonction doit avoir un argument `bit_string` de type `str` et qu'elle doit retourner un `numpy.ndarray` contenant des booléens (`NDArray[np.bool_]`). L'ordre des booléens devrait être tel que le premier qubit soit en première position.

```

1 bits = bitstring_to_bits("0111")
2 print(bits)

```

```
[ True True True False]
```

Estimation de valeur moyenne Le calcul de la valeur moyenne d'une chaîne de Pauli quelconque repose sur le calcul de la valeur moyenne d'une chaîne de Pauli diagonale. Vous devrez donc implémenter une fonction qui étant donné une chaîne de Pauli diagonale et les résultats d'une exécution de circuit (sous la forme de `counts`), calcule une estimation de la valeur moyenne.

```

1 def diag_pauli_expectation_value(pauli: Pauli, counts: dict) -> float:
2
3     assert(np.all(~pauli.x))
4     expectation_value = ...
5
6     return expectation_value

```

Diagonalisation de chaines de Pauli Pour les ordinateurs quantiques qui ne peuvent effectuer des mesures que dans la base computationnelle, il est nécessaire d'appliquer une transformation pour diagonaliser des chaines de Pauli non diagonales. Vous devrez donc implémenter une fonction qui diagonalise une chaîne de Pauli. Celle-ci devra retourner une chaîne de Pauli diagonale ainsi que le circuit quantique qui applique ladite transformation.

```

1 def bitwise_diagonalize_pauli_with_circuit(pauli : Pauli) -> Tuple[Pauli, QuantumCircuit]:
2
3     diagonal_pauli = Pauli(...)
4     circuit = QuantumCircuit(...)
5     assert(np.all(~diagonal_pauli.x))
6
7     return diagonal_pauli, circuit

```

Estimation de plusieurs valeurs moyennes Selon l'approche que nous utilisons, l'estimation des valeurs moyennes de plusieurs chaines de Pauli requiert autant de circuits quantiques. Idéalement, on veut assembler tous ces circuits dans une liste qu'on fournira à la fonction `sampler.run` pour les réunir dans une seule job.

Ainsi, vous devrez implémenter une fonction qui prend en entrée la liste des chaines de Pauli (\hat{P}_i) et un circuit quantique qui définit l'état quantique ($|\psi\rangle$) sur lequel calculer les valeurs moyennes. Cette fonction doit également prendre en entrée le `Sampler` sur lequel exécuter les circuits. Cette fonction doit retourner un `numpy.ndarray` contenant les valeurs moyennes ($\langle\psi|\hat{P}_i|\psi\rangle$).

```

1 def estimate_paulis_expectation_values(
2     paulis: PauliList,
3     state_circuit: QuantumCircuit,
4     sampler: BaseSamplerV2
5 ) -> NDArray[np.float64]:
6
7     expectation_values = ...
8
9     return expectation_values

```

Tomographie d'état quantique Une fois que vous serez capables d'estimer les valeurs moyennes de plusieurs chaines de Pauli, et en vous appuyant sur les notions de tomographie (section 5), vous devriez être en mesure d'effectuer la tomographie d'un état quantique.

En effet, en mesurant toutes les chaines de Pauli à n qubits, vous devriez être capable de reconstruire la matrice densité de l'état quantique.

La fonction à implémenter doit donc prendre en entrée le circuit quantique qui prépare un état quantique donnée et le `Sampler` sur lequel exécuter les circuits quantiques.

```

1 def state_tomography(
2     state_circuit: QuantumCircuit,
3     sampler: BaseSamplerV2
4 ) -> NDArray[np.complex128]:
5
6     density_matrix = ...
7
8     return density_matrix

```

Il ne vous restera qu'à extraire le vecteur d'état de cette matrice densité.

```
1 def density_matrix_to_statevector(density_matrix: NDArray) -> NDArray[np.complex128]:  
2     statevector = ...  
3  
4     return statevector
```

z	x	$\hat{Z}^z \hat{X}^x$
0	0	\hat{I}
1	0	\hat{Z}
0	1	\hat{X}
1	1	$i\hat{Y}$

TABLE 1 – Opérateur appliqué sur le qubit q pour chaque combinaison des composantes binaires.

4 Chaines de Pauli

Nous avons déjà vu qu'on définit une chaîne de Pauli de longueur n comme un produit tensoriel de n opérateurs de Pauli (+ l'identité)

$$\hat{P} = \bigotimes_{q=0}^{n-1} \hat{\sigma}_q \quad \text{avec} \quad \hat{\sigma}_q \in \{\hat{I}_q, \hat{X}_q, \hat{Y}_q, \hat{Z}_q\}.$$

4.1 Représentation zx

Pour traiter les chaines de Pauli à l'aide d'un ordinateur on pourrait utiliser des chaînes de caractères (ex : "ZIXY"), mais cela ne serait pas très pratique pour effectuer des opérations. La représentation zx , aussi appelée symplectique, est beaucoup plus adaptée. Celle-ci s'appuie sur le fait que l'exponentiation d'un opérateur par les nombres binaires 0 ou 1 résulte respectivement en l'identité ou en lui-même. Par exemple, pour l'opérateur \hat{Z} on voit que

$$\hat{Z}^z = \begin{cases} \hat{I} & \text{si } z = 0 \\ \hat{Z} & \text{si } z = 1. \end{cases}$$

Comme il existe quatre opérateurs à un qubit possibles, seulement deux bits d'information sont nécessaires pour spécifier duquel il s'agit. En effet, on peut spécifier cet opérateur grâce à

$$\hat{\sigma} = (-i)^{zx} \hat{Z}^z \hat{X}^x$$

En effet, le tableau 1 dresse la liste des opérateurs $\hat{Z}^z \hat{X}^x$ résultant pour toutes les combinaisons des z et x possibles. On remarque que pour $z = x = 1$, le résultat est $\hat{Z}\hat{X} = i\hat{Y}$ et pas seulement \hat{Y} . D'où l'importance du facteur $(-i)^{zx}$ qui s'applique uniquement dans cette situation et compense le facteur i .

Pour un système de n qubits on peut représenter une chaîne de Pauli agissant avec des \hat{Z} sur différents qubits comme

$$\hat{\mathcal{Z}} = \hat{Z}^{\mathbf{z}}$$

où

$$\mathbf{z} = (z_0 \ z_1 \ \cdots \ z_{n-1})$$

est un vecteur avec n composantes binaires. Ainsi, la chaîne de Pauli \hat{P} agit avec un \hat{Z} sur le qubit q si la composante binaire $z_q = 1$. Par exemple, pour 3 qubits

$$\hat{Z}^{(1,1,0)} = \hat{Z}_0 \hat{Z}_1 \hat{I}_2 = \hat{I}_2 \hat{Z}_1 \hat{Z}_0 = \hat{I} \hat{Z} \hat{Z}.$$

En faisant la même chose pour l'opérateur \hat{X} avec un second vecteur de composantes binaires, de même longueur que \mathbf{z} ,

$$\mathbf{x} = (x_0 \ x_1 \ \cdots \ x_{n-1})$$

on peut donc représenter n'importe quelle chaîne de Pauli grâce à

$$\hat{P} = (-i)^{\mathbf{z} \cdot \mathbf{x}} \hat{Z}^{\mathbf{z}} \hat{X}^{\mathbf{x}}. \quad (1)$$

où le produit scalaire

$$\mathbf{z} \cdot \mathbf{x} = \sum_q z_q x_q \pmod{4}$$

compte le nombre de matrices \hat{Y} dans la chaîne. Celui-ci est calculé comme un entier modulo 4, car $(-i)^4 = 1$. Ainsi on peut décrire une chaîne de Pauli en spécifiant ses deux vecteurs \mathbf{z} et \mathbf{x} .

5 Notions de tomographie quantique

La tomographie d'un état quantique consiste à effectuer plusieurs mesures sur plusieurs copies d'un même état quantique avec l'objectif de pouvoir le décrire avec précision. Dans notre cas cela veut dire de reconstruire son vecteur d'état. Nous allons utiliser une approche systématique qui repose sur la reconstruction de ce qu'on appelle la *matrice densité*.

5.1 Matrice densité

La matrice densité est comme le vecteur d'état un outil qui permet de décrire l'état d'un système quantique.

Remarque

Notez que l'utilité de la matrice densité va bien au-delà de l'utilisation qu'on va en faire ici. En particulier, on se limitera à décrire des états quantiques dits *purs*. Tous les états quantiques qu'on a vus jusqu'à maintenant sont des états purs. La matrice densité permet également de décrire des états quantiques *mixtes* qui traduisent notre ignorance sur l'état d'un système quantique. Nous n'aborderons pas les états mixtes dans ce projet.

La matrice densité, notée $\hat{\rho}$, d'un état quantique pur $|\psi\rangle$ est simplement construite comme le produit dyadique

$$\hat{\rho} = |\psi\rangle\langle\psi|.$$

Par exemple, pour un état à deux qubits dont le vecteur d'état, dans la base computationnelle, est

$$|\psi\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}. \quad (2)$$

la matrice densité est le produit dyadique suivant (à ne pas confondre avec le produit scalaire)

$$|\psi\rangle\langle\psi| = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix} \begin{pmatrix} \alpha_{00}^* & \alpha_{01}^* & \alpha_{10}^* & \alpha_{11}^* \end{pmatrix} = \begin{pmatrix} |\alpha_{00}|^2 & \alpha_{00}\alpha_{01}^* & \alpha_{00}\alpha_{10}^* & \alpha_{00}\alpha_{11}^* \\ \alpha_{01}\alpha_{00}^* & |\alpha_{01}|^2 & \alpha_{01}\alpha_{10}^* & \alpha_{01}\alpha_{11}^* \\ \alpha_{10}\alpha_{00}^* & \alpha_{10}\alpha_{01}^* & |\alpha_{10}|^2 & \alpha_{10}\alpha_{11}^* \\ \alpha_{11}\alpha_{00}^* & \alpha_{11}\alpha_{01}^* & \alpha_{11}\alpha_{10}^* & |\alpha_{11}|^2 \end{pmatrix}. \quad (3)$$

La matrice densité contient toute l'information nécessaire pour reconstruire le vecteur d'état $|\psi\rangle$.

Remarque sur la phase globale

Fait intéressant, la matrice densité élimine le concept de phase globale. En effet, supposons deux états quantiques liés par une phase globale

$$|\psi'\rangle = e^{i\theta}|\psi\rangle.$$

On sait que ces deux états sont physiquement indistinguables, car toutes les opérations et mesures qu'on pourrait effectuer sur l'un et sur l'autre produiraient exactement les mêmes résultats.

La matrice densité, pour l'état $|\psi'\rangle$ s'obtient comme

$$\hat{\rho}' = |\psi'\rangle\langle\psi'| = e^{i\theta}|\psi\rangle e^{-i\theta}\langle\psi| = |\psi\rangle\langle\psi| = \hat{\rho}$$

où on voit que la phase globale s'annule trivialement et donc que les états physiquement indistinguables sont décrits par la même matrice densité.

Remarque sur la base

Réexprimons l'état $|\psi\rangle$ (équation 2) dans une base orthonormée $\{|\phi_i\rangle\}$ quelconque

$$|\psi\rangle = \sum_{i=0}^{4-1} a_i |\phi_i\rangle.$$

On pourrait écrire la matrice densité de cet état dans cette base et obtenir quelque chose de similaire à l'équation 3. Plutôt que de faire cela, choisissons cette base de sorte que $|\phi_0\rangle = |\psi\rangle$. La matrice densité, exprimée dans cette base est alors simplement

$$\hat{\rho} = |\psi\rangle\langle\psi| = |\phi_0\rangle\langle\phi_0| = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Pour un état pur, on peut toujours exprimer la matrice densité de cette manière. Il suffit de l'exprimer dans une base où l'un des vecteurs de base est ce même état.

5.2 Valeur moyenne et matrice densité

Montrons maintenant comment exprimer la valeur moyenne d'une observable \hat{A} pour un état $|\psi\rangle$ à l'aide de sa matrice densité. Pour ce faire, on utilise la relation de fermeture

$$\hat{I} = \sum_j |\phi_j\rangle\langle\phi_j|$$

qui est valide pour n'importe quel ensemble d'états $|\phi_j\rangle\langle\phi_j|$ qui forme une base orthonormée. En insérant une telle relation de fermeture dans l'expression d'une valeur moyenne, on peut écrire celle-ci comme

$$\langle\psi|\hat{A}|\psi\rangle = \langle\psi|\hat{A}\hat{I}|\psi\rangle = \langle\psi|\hat{A}\left(\sum_j |\phi_j\rangle\langle\phi_j|\right)|\psi\rangle = \sum_j \langle\psi|\hat{A}|\phi_j\rangle\langle\phi_j|\psi\rangle$$

où $\langle\psi|\hat{A}|\phi_j\rangle$ et $\langle\phi_j|\psi\rangle$ sont des nombres complexes. Comme la multiplication de nombres est commutative, on peut réexprimer la valeur moyenne ainsi

$$\langle\psi|\hat{A}|\psi\rangle = \sum_j \langle\phi_j|\psi\rangle\langle\psi|\hat{A}|\phi_j\rangle.$$

Cela permet en fait de faire apparaître la matrice densité $\hat{\rho} = |\psi\rangle\langle\psi|$. La valeur moyenne d'une observable \hat{A} pour un état quantique $|\psi\rangle$ peut donc être calculée grâce à

$$\langle\psi|\hat{A}|\psi\rangle = \sum_j \langle\phi_j|\hat{\rho}\hat{A}|\phi_j\rangle = \text{Tr}(\hat{\rho}\hat{A}) \quad (4)$$

où l'opération Tr correspond au calcul de la trace d'une matrice.

5.3 Trace

La trace d'une matrice carrée est la somme de ses éléments diagonaux. En notation indicelle, la trace d'une matrice M s'obtient en effectuant le calcul suivant

$$\text{Tr}(M) = \sum_i M_{ii} = M_{00} + M_{11} + \dots$$

La trace est une opération mathématique très utile qui comporte un bon nombre de propriétés intéressantes. Donnons quelques-unes de ces propriétés, sans nécessairement les démontrer, par souci de concision. D'abord la trace est linéaire c'est-à-dire que

$$\text{Tr}(A + B) = \text{Tr}(A) + \text{Tr}(B) \quad \text{et} \quad \text{Tr}(cA) = c \text{Tr}(A).$$

Ensuite, la trace est cyclique

$$\text{Tr}(ABC) = \text{Tr}(BCA) = \text{Tr}(CAB).$$

Cela a pour conséquence que la trace est indépendante de la base. En effet,

$$\text{Tr}(UAU^{-1}) = \text{Tr}(AU^{-1}U) = \text{Tr}(A).$$

En choisissant d'exprimer la trace d'une matrice dans la base dans laquelle elle est diagonale, on voit que la trace est égale à la somme de ses valeurs propres

$$\text{Tr}(M) = \sum_i \lambda_i. \tag{5}$$

Finalement, la trace d'un produit tensoriel est égale au produit des traces

$$\text{Tr}(A \otimes B) = \text{Tr}(A) \text{Tr}(B). \tag{6}$$

5.4 Trace et chaines de Pauli

Comme les chaines de Pauli peuvent être représentées par des matrices, on peut également calculer la trace d'une chaîne de Pauli.

Commençons par calculer la trace de la chaîne de Pauli de longueur n entièrement composé d'opérateur identité. La matrice qui représente cette chaîne de Pauli est une matrice identité de taille $2^n \times 2^n$. La somme de ces éléments diagonaux est donc simplement 2^n . Pour toutes les autres chaines de Pauli, le résultat est le même : 0 ! En effet, du moment que la chaîne de Pauli comporte un \hat{X} ou un \hat{Y} , les éléments sur la diagonale sont tous 0. Finalement, si la chaîne de Pauli ne comporte que des opérateurs \hat{I} et \hat{Z} (avec au moins un \hat{Z}), il y aura autant d'éléments diagonaux positifs que négatifs. Ainsi, on peut résumer la trace d'une chaîne de Pauli par

$$\text{Tr}(\hat{P}) = \begin{cases} 2^n & \text{si } \hat{P} = \hat{I}^{\otimes n} \\ 0 & \text{sinon.} \end{cases} \tag{7}$$

Une autre manière d'interpréter cela repose sur le fait que les valeurs propres des opérateurs de Pauli sont toutes égales ± 1 . Cela, combiné à l'équation 6, permet de voir que les valeurs propres des chaines de Pauli sont également toutes égales à ± 1 et que le nombre de valeurs propres positives est égal au nombre de valeurs propres négatives, sauf pour la chaîne de Pauli identité. En utilisant l'équation 5 on arrive alors au même résultat qu'à l'équation 7.

Calculons maintenant la trace d'un produit de chaines de Pauli. Remarquons d'abord que, comme les opérateurs de Pauli, les chaines de Pauli sont égales à leurs propres inverses. En effet,

$$\hat{P}^2 = \hat{I}$$

pour toutes les chaines de Pauli. Une conséquence de cela est que le produit de deux chaines de Pauli produit la matrice identité si et seulement si il s'agit de la même chaîne de Pauli

$$\hat{P}_1 \hat{P}_2 = \hat{I} \quad \text{ssi} \quad \hat{P}_1 = \hat{P}_2.$$

Dans tous les autres cas, un tel produit retourne toujours une troisième chaîne de Pauli (à une phase près) différente de l'identité

$$\hat{P}_1 \hat{P}_2 \rightarrow \hat{P}_3 \neq \hat{I} \quad \text{ssi} \quad \hat{P}_1 \neq \hat{P}_2$$

Ainsi, en utilisant l'équation 7 on peut facilement démontrer que la trace du produit de deux chaines de Pauli est

$$\text{Tr}(\hat{P}_1 \hat{P}_2) = \begin{cases} 2^n & \text{si } \hat{P}_1 = \hat{P}_2 \\ 0 & \text{sinon.} \end{cases} \quad (8)$$

5.5 Tomographie via la mesure des chaines de Pauli

On cherche donc à déduire l'état quantique d'un système $|\psi\rangle$ en effectuant des mesures sur celui-ci. La matrice densité de ce système de n qubits est une matrice de taille $2^n \times 2^n$ pouvant donc être exprimée comme une combinaison linéaire de chaines de Pauli

$$\hat{\rho} = |\psi\rangle\langle\psi| = \sum_i a_i \hat{P}_i. \quad (9)$$

Si on arrive à déterminer les coefficients a_i nous serons capable de reconstruire la matrice densité et donc son vecteur d'état.

Supposons qu'on effectue une mesure pour estimer la valeur moyenne d'une chaîne de Pauli \hat{P}_m . En utilisant l'équation 4 on peut écrire cette valeur moyenne

$$\langle \psi | \hat{P}_m | \psi \rangle = \text{Tr}\left(\hat{\rho} \hat{P}_m\right) = \text{Tr}\left(\sum_i a_i \hat{P}_i \hat{P}_m\right) = \sum_i a_i \text{Tr}(\hat{P}_i \hat{P}_m).$$

Or, en utilisant l'équation 8 on peut facilement montrer que cette valeur moyenne permet de déterminer le coefficient associé à \hat{P}_m . En effet, on obtient

$$\langle \psi | \hat{P}_m | \psi \rangle = 2^n a_m \quad \text{ou} \quad a_m = \frac{1}{2^n} \langle \psi | \hat{P}_m | \psi \rangle.$$

En effectuant des mesures pour les 4^n chaines de Pauli on peut donc extraire les coefficients a_i et reconstruire complètement la matrice densité grâce à l'équation 9. Une fois cette matrice densité obtenue, on a toute l'information nécessaire pour déduire le vecteur d'état $|\psi\rangle$.

Remarque

Cette méthode de tomographie n'est pas très optimale. Il existe des astuces pour rendre ce processus plus efficace. Néanmoins, sauf si on fait des hypothèses sur l'état quantique ou si on fait des compromis sur la précision de l'estimation, le nombre d'opérations nécessaires pour reconstruire son vecteur d'état sera exponentiel avec le nombre de qubits.