Compte-rendu du projet Qiskit

Exercice 1 et 2

L'objectif du circuit des deux premiers exercices est de réaliser la trasformation :

$$\begin{cases} |0\rangle & \rightarrow & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle & \rightarrow & \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases}$$

Pour cela nous allons appliquer au qubit d'entrée une porte d'Hadamard.

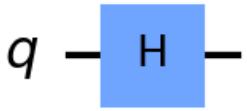
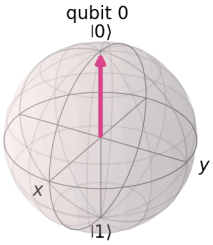
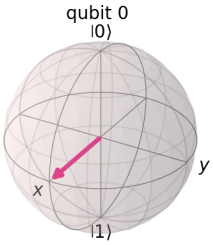


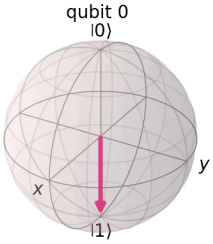
FIGURE 1 – Circuit de l'exercice 1 et 2



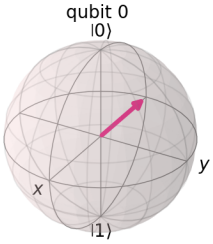
(a) Etat initial



(b) Etat final



(c) Etat initial



(d) Etat final

FIGURE 2 – Sphère de Bloch du circuit pour une entrée $|0\rangle$ (a,b) et $|1\rangle$ (c,d)

Exercice 3

Pour effectuer une rotation π on peut utiliser une porte X :

FIGURE 3 – Circuit de l'exercice 3

Pour un état d'entrée $|0\rangle$ on retrouve bien en sortie $|1\rangle$.

(a) Etat initial (b) Etat final

FIGURE 4 – Sphère de Bloch du circuit pour une entrée $|0\rangle$

Exercice 4

Pour créer la transformation $|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, nous pouvons appliquer successivement une porte X puis une porte d'Hadamard au qubit d'entrée :

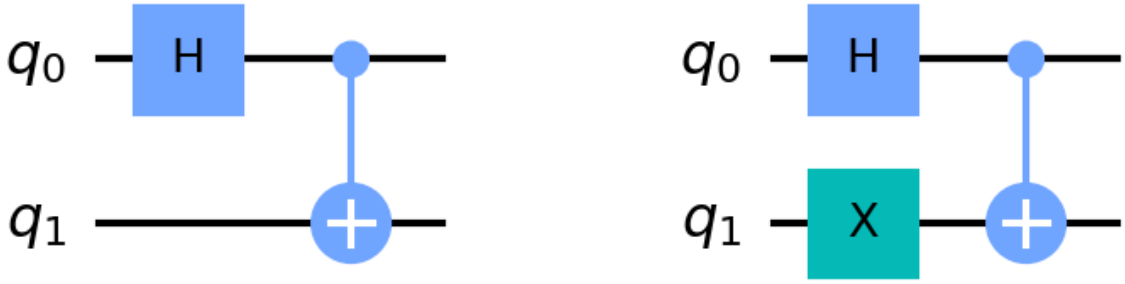
FIGURE 5 – Circuit de l'exercice 4

(a) Etat initial (b) Etat final

FIGURE 6 – Sphère de Bloch du circuit pour une entrée $|0\rangle$

Exercice 5 et 6

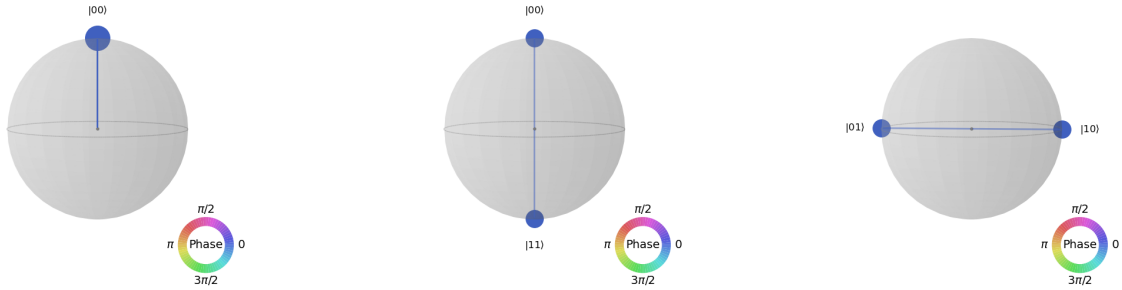
Pour créer les états de Bell β_{00} et β_{01}, on utilise les portes X et H vues précédemment, mais aussi une porte CNOT :



(a) Création de β_{00} (b) Création de β_{01}

FIGURE 7 – Circuits de l'exercice 5 (a) et 6 (b)

En utilisant la fonction `Statevector.draw('latex')` de l'état de sortie de l'exercice 5, on obtient : $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$; et pour l'exercice 6 : $\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$. Comme les états de sortie sont intriqués, il n'est pas possible de les décomposer en deux qubits distincts et donc d'utiliser la représentation `'bloch'` de l'objet `Statevector`. On utilise alors la visualisation `'qsphere'` :



(a) Etat initial (b) Etat final Ex.5 (c) Etat final Ex.6

FIGURE 8 – Etats d'entrée et de sortie

Exercice 7

L'objectif de cet exercice est d'appliquer la transformation $|010\rangle \rightarrow |001\rangle$ et de mesurer le résultat obtenu (il faut se souvenir de la convention utilisé par *Qiskit* : $|q_2, q_1, q_0\rangle$).

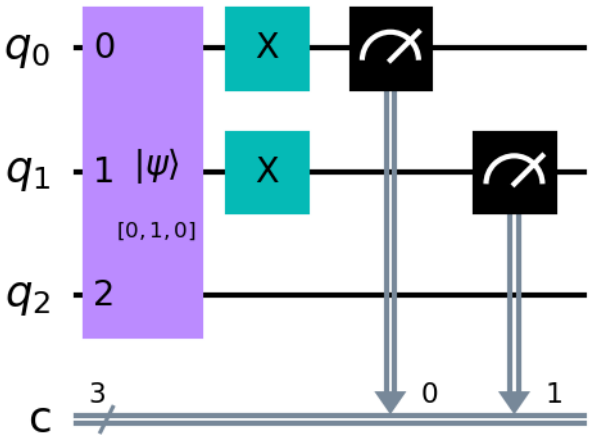


FIGURE 9 – Circuit de l'exercice 7

Pour cela on applique une porte X aux qubits q_0 et q_1. On mesure exclusivement l'état $|001\rangle$ en sortie car :

- il n'y a aucune superposition d'état au moment de la mesure, car l'état de sortie est pur.
- il a été utilisé le simulateur `Aer` sans chercher à modéliser du bruit.

(a) Etat initial (b) Etat final

FIGURE 10 – Représentation `'qsphere'` des états d'entrée et de sortie de l'exercice 7

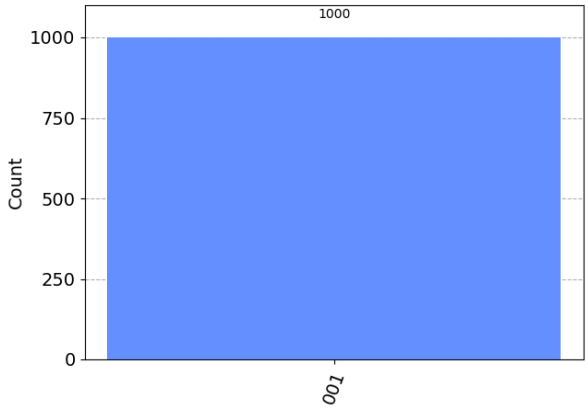


FIGURE 11 – Mesures effectuées sur le circuit de l'exercice 7

Exercice 8

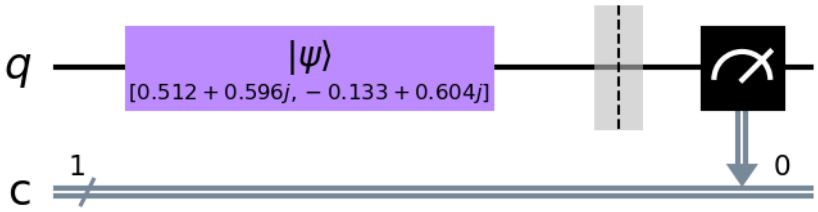


FIGURE 12 – Circuit de l'exercice 8

Le circuit de cette exercice (Fig. 12) crée l'état : (0,512 + 0,506 i) $|0\rangle$+(-0,133 + 0,604 i) $|1\rangle = a\,|0\rangle + b\,|1\rangle$. Pour connaître les probabilités de mesures des états de la base on calcule : $|a|^2 = 0{,}618$, $|b|^2 = 0{,}382$. En répétant les mesures 10000 fois (Fig. 13), on retrouve bien les probabilités de mesurer le qubit dans l'état $|0\rangle$ ou $|1\rangle$.
Il faut faire attention à une particularité de ce circuit : en refaisant tourner le script un nouvel état aléatoire sera imposé à l'initialisation du circuit, ce qui modifiera les probabilités associées et donc les mesures du qubit.

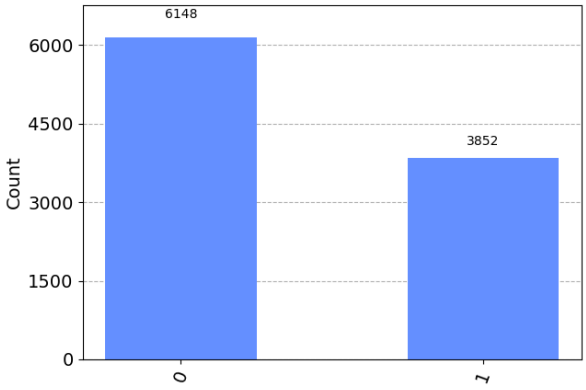


FIGURE 13 – Mesures du qubit du circuit

Exercice 9

Le circuit suivant permet de réaliser la téléportation quantique d'un qubit en utilisant 2 autres qubits intriquées.

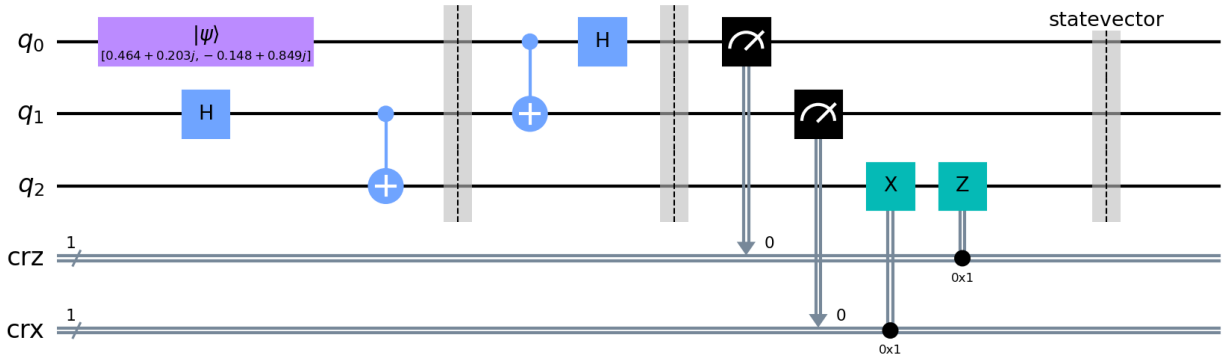
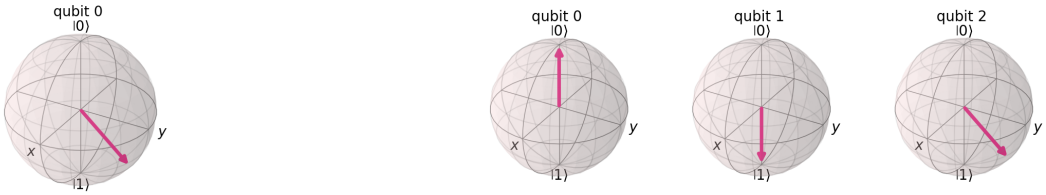


FIGURE 14 – Circuit de l'exercice 9

Initialement le qubit q_0 est dans un état aléatoire que l'on souhaite téléporter, et les 2 autres sont dans un état $|0\rangle$. A la fin du circuit on constate que l'état initial de q_0 a été transférer à q_2.



(a) Etat initial de q_0 (b) Etat final de $\{q_0, q_1, q_2\}$

FIGURE 15 – Sphère de Bloch des états d'entrée et de sortie de l'exercice 9

Conclusion

Ce projet m'a permis de découvrir le module *Qiskit*, de mettre en place quelques simulations sur des circuits simples. Le dernier exercice m'a permis de mieux comprendre la mise en place de la téléportation quantique qui est un phénomène assez subtil à appréhender.

Annexe : code python des exercices

```python
from qiskit import *
from qiskit.quantum_info import Statevector, random_statevector
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram,plot_bloch_multivector
import matplotlib.pyplot as plt
import numpy as np

## EXERCICE 1
titre = "Exercice 1"
circuit = QuantumCircuit(1)
circuit.h(0)
circuit.draw('mpl',filename=titre+" - Circuit")
#Visualiser les états
state = Statevector.from_int(0, 2)
state.draw('bloch',filename=titre+" - Etat initial")
state = state.evolve(circuit)
state.draw('bloch',filename=titre+" - Etat final")

## EXERCICE 2
titre = "Exercice 2"
circuit = QuantumCircuit(1)
circuit.h(0)
#Visualiser les états
circuit.draw('mpl',filename=titre+" - Circuit")
state = Statevector.from_int(1, 2)
state.draw('bloch',filename=titre+" - Etat initial")
state = state.evolve(circuit)
state.draw('bloch',filename=titre+" - Etat final")

## EXERCICE 3
titre = "Exercice 3"
circuit = QuantumCircuit(1)
circuit.x(0)
circuit.draw('mpl',filename=titre+" - Circuit")
#Visualiser les états
state = Statevector.from_int(0, 2)
state.draw('bloch',filename=titre+" - Etat initial")
state = state.evolve(circuit)
state.draw('bloch',filename=titre+" - Etat final")

## EXERCICE 4
titre = "Exercice 4"
circuit = QuantumCircuit(1)
circuit.x(0)
circuit.h(0)
circuit.draw('mpl',filename=titre+" - Circuit")
#Visualiser les états
state = Statevector.from_int(0, 2)
state.draw('bloch',filename=titre+" - Etat initial")
state = state.evolve(circuit)
state.draw('bloch',filename=titre+" - Etat final")

## EXERCICE 5
titre = "Exercice 5"
circuit = QuantumCircuit(2)
circuit.h(0)
circuit.cnot(0,1)
circuit.draw('mpl',filename=titre+" - Circuit")
#Visualiser les états
state = Statevector.from_int(0, 2**2)
state.draw('qsphere',filename=titre+" - Etat initial")
```

```
state = state.evolve(circuit)
state.draw('latex_source')
state.draw('qsphere',filename=titre+" - Etat final")

## EXERCICE 6
titre = "Exercice 6"
circuit = QuantumCircuit(2)
circuit.h(0)
circuit.x(1)
circuit.cnot(0,1)
circuit.draw('mpl',filename=titre+" - Circuit")
#Visualiser les états
state = Statevector.from_int(0, 2**2)
state.draw('qsphere',filename=titre+" - Etat initial")
state = state.evolve(circuit)
state.draw('latex_source')
state.draw('qsphere',filename=titre+" - Etat final")

## EXERCICE 7
titre = "Exercice 7"
circuit = QuantumCircuit(3,3)
circuit.initialize('010')
init = Statevector(circuit)
circuit.x(0)
circuit.x(1)
final = Statevector(circuit)
circuit.measure([0, 1], [0, 1])
circuit.draw('mpl',filename=titre+" - Circuit")
init.draw('qsphere',filename=titre+" - Etat initial")
final.draw('qsphere',filename=titre+" - Etat final")
#Mesures des bits classiques
#Choix du simulateur
simulator = AerSimulator()
circuit.save_statevector()
#Compilation du circuit
compiled_circuit = transpile(circuit, simulator)
# Execution du circuit sur le simulateur
job = simulator.run(compiled_circuit, shots=1000)
# Agrégation des résultats
result = job.result()
counts = result.get_counts(compiled_circuit)
plot_histogram(counts,filename=titre+" - Mesures")

## EXERCICE 8
titre = "Exercice 8"
circuit = QuantumCircuit(1,1)
psi = random_statevector(2)
circuit.initialize(psi,[0])
circuit.barrier()
final = Statevector(circuit)
circuit.measure(0,0)
circuit.draw('mpl',filename=titre+" - Circuit")
psi.draw('bloch',filename=titre+" - Etat initial")
print(psi.draw('latex_source'))
print('|a|² = ',np.abs(psi[0])**2,', |b|² = ',np.abs(psi[1])**2)
#Mesures des bits classiques
compiled_circuit = transpile(circuit, simulator)
job = simulator.run(compiled_circuit, shots=10000)
result = job.result()
counts = result.get_counts(compiled_circuit)
plot_histogram(counts,filename=titre+" - Mesures")

## EXERCICE 9
```

```
titre = "Exercice 9"
init = random_statevector(2)
qr= QuantumRegister(3, name="q")
crz, crx = ClassicalRegister(1, name="crz"), ClassicalRegister(1,name="crx")
teleportation_circuit = QuantumCircuit(qr, crz, crx)
teleportation_circuit.initialize(init,0)
teleportation_circuit.h(1)
teleportation_circuit.cnot(1,2)
teleportation_circuit.barrier()
teleportation_circuit.cnot(0,1)
teleportation_circuit.h(0)
teleportation_circuit.barrier()
teleportation_circuit.measure([0,1],[0,1])
teleportation_circuit.x(2).c_if(crx,1)
teleportation_circuit.z(2).c_if(crz,1)
teleportation_circuit.save_statevector()
teleportation_circuit.draw('mpl',filename=titre+" - Circuit")
#Visualiser les états par simulation du cicuit
compiled_circ = transpile(teleportation_circuit, simulator)
result = simulator.run(compiled_circ).result()
counts = result.get_counts(compiled_circ)
final = result.get_statevector(teleportation_circuit)
init.draw('bloch',filename=titre+" - Etat initial")
final.draw('bloch',filename=titre+" - Etat final")
```