Quantum Information and Quantum Computing, Problem set 4

Assistants : sara.alvesdossantos@epfl.ch, david.linteau@epfl.ch, shao.chiew@epfl.ch

This exercise session represents the first hands-on of the course. We will see that is possible simulate quantum circuit and even have access to real quantum hardware directly from your laptop. To this purpose, we will use Qiskit, an open-source SDK (Software Developement Kit) developed by IBM for working with quantum computers at the level of pulses, circuits, and application modules.

Qiskit has been written in Python, therefore we will use this programming language in our hands-on.

After having installed Python and the latest release of the Qiskit library, in this lesson we will see some basic usage of it.

Another possibility for those who can't install python on their devices is to use the IBM Quantum Lab.

Problem 1 : A first quantum circuit in Qiskit

Use Python and the Qiskit library to build a single or multi qubit circuit, then measure the outcome and plot the results. In particular:

- Initialise a single qubit circuit, apply an X gate and measure it. Do the same with other single qubit gates such as H,S and T.
- Create a Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and measure the two qubits. You can also create the other Bell states seen in the last exercise session. What do you notice? And if you increment the number of shots?

Problem 2 : Use different simulators in Qiskit

Through Qiskit Aer, you have the possibility to simulate an ideal quantum computer, but also to reproduce at a certain level of accuracy the different sources of error that can affect results on real devices.

To use the Noise Model from real quantum devices, create an account on the IBM Quantum Experience and then run the following code to link your account to the laptop you are using

```
!pip install qiskit_ibm_runtime
from qiskit_ibm_runtime import QiskitRuntimeService

# Save your credentials on disk.
QiskitRuntimeService.save_account(channel='ibm_quantum', token=<IBM Quantum API
    key>)

service = QiskitRuntimeService(
    channel='ibm_quantum',
    instance='ibm-q/open/main',
)
```

where <IBM Quantum API key> is the alpha-numeric code that you can find in the homepage of your IBM Quantum Experience.

Then,create a four qubit GHZ state

$$\frac{1}{\sqrt{2}}\left(|0000\rangle + |1111\rangle\right)$$

and measure it. Simulate you sistem with the `statevector_simulator`, the `qasm_simulator` and apply a `NoiseModel` from a device of you choice (at least of 4 qubits), then plot the results. What do you notice?

Problem 3 : Transpile a Quantum Circuit

Every quantum device has a set of gates that are implemented natively. This set varies through the different platforms (superconducting, quantum dots, ion traps, dots) and sometimes even between different hardware of the same type.

This means that once the circuit is created it has to be translated into a set of instructions that the hardware can understand. This procedure is called transpilation and it can increase significantly the depth of the circuit.

Qiskit offer the possibility to transpile the circuit on your laptop before sending the instructions to the quantum hardware, in order to control the final result. Transpilation is not an easy task and, given the same circuit as input, the output results may very from (quantum) device to (quantum) device.

Here you have to use the knowledge of the previous steps to create a circuit and then transpile it on a real hardware. You can create whatever circuit you desire, but add also controlled gates between different qubits, to see how the configuration of the device affects the results. Then, you can use the options of the `transpile` function to try to reduce the depth of the final circuit.

Problem 4 : Quantum Fourier Transform on Qiskit

Using Qiskit, implement the Quantum Fourier Transform circuit seen in the last exercise session. Run it on `StatevectorSimulator`, `QasmSimulator` and `NoiseModel`. Try with different initialisations: $|000\rangle, |111\rangle, \ldots$ and see what happens. Is it what you expect? What happens when you consider noise in your device?