Project #1

Instructor: Dr. Fedrici Due date: 5^{th} April, 2020

Problem 1

In this project you will use Qiskit to benchmark a QPU. Rather than using a real QPU for the assignment you will use a simulated five-qubit one to practice your code with.

More specifically, you will consider amplitude- and phase- damping effects through T1 and T2* coherence time measurements - the star on T2 comes from the distinction usually made in between two different ways of measuring this quantity, here we will only consider the way introduced in class that usually refers as a T2* measurement. Remember, to access T1 and T2* you will have to perform the following experiments:

- **T1 experiment**: Apply a X gate, wait for time t, measure in the computational basis. Repeat for increasing values of t to access $P_1(t)$ - the probability of getting outcome 1 at time t. The decay time of this experiment is T1;
- **T2* experiment**: Apply a H gate, wait for time t, apply a second H gate, measure in the computational basis. Repeat for increasing values of t to access $P_1(t)$ - the probability of getting outcome 1 at time t. The decay time of this experiment is T2*.

An artificial delay can be applied through the implementation of successive identity gates. You are told that the duration of a single-qubit gate is 0.1 microseconds.

To access T1 and T2*, Qiskit comes with the **qiskit.ignis.characterization.coherence** submodule which provides the following functions - see Qiskit documentation for implementation details:

- **t1_circuits(num_of_gates, gate_time, qubits)**: Generates circuit for T1 measurement;
- **t2star_circuits(num_of_gates, gate_time, qubits)**: Generates circuit for T2* measurement;
- **T1Fitter(backend_result, xdata, qubits,...)**: T1 fitter;
- **T2starFitter(backend_result, xdata, qubits,...)**: T2* fitter.

Accompanying this assignment sheet is a **noise_model.py** file that provides two functions returning the noise models to study. In the following sections, your task is to load the noise models from the file by calling the aforementioned functions and then write benchmarking programs that characterize the performance of the simulated QPU with that model. If this were a real QPU you would just run your code against that processor backend directly.

In order to be graded correctly, your submitted code must include a **benchmark.py** file with the appropriate functions as described below.

a. Write a function in benchmark.py called **T1Benchmark** whose single argument is the string name of the noise model file and whose output is a dictionary whose keys are qubit ids and whose values are the T1 coherence time of those qubits in seconds, e.g.: {Q0: 1.3e-6, Q1: 6e-7, ...}.

b. Write a function in benchmark.py called **T2starBenchmark** whose single argument is the string name of the noise model file and whose output is a dictionary whose keys are qubit ids and whose values are the T2* coherence time of those qubits in seconds, e.g.: {Q0: 0.45e-6, Q1: 5e-7, ...}.