PHYC90045 Introduction to Quantum Computing

Lab Session 9

9.1 Introduction

Welcome to Lab 9 of PHYC90045 Introduction to Quantum Computing. The purpose of this lab session is to program and run circuits in the IBM Quantum Experience.

9.2 IBM Quantum Experience

The IBM Quantum Experience allows you program and run circuits on actual hardware, as described in the lectures. You may work in pairs and/or groups to pool run units and avoid long wait times. Before using your "experiment units" check your circuits by running the simulator and/or the QUI. Note the IBM simulator is not like the QUI – it runs a certain number of times to gather the statistics to give you the (approximate) probability distribution at the measurement (unlike the QUI you have to put in measurement gates to get results for both the IBM devices and the simulator). The simulator runs the code noise/error free (just as in the QUI, unless you put in error gates deliberately), but make sure to set the number of simulator runs to at least 1000 so the results aren't skewed by low statistics.

Hints: if you're programming a lot of gates switch to the QASM editor and then back to the composer to check. Save your circuits with a sensible name, and refresh the results section in the lower section to see your results after the experiment has run. If there is a cached version of your circuit, use it!

Exercise 9.2.1 When you access the system on the right hand side of the screen, you will see several devices which you can use. Clicking on any one of them brings up details about that device. For example, here is a summary of the IBM Vigo device:
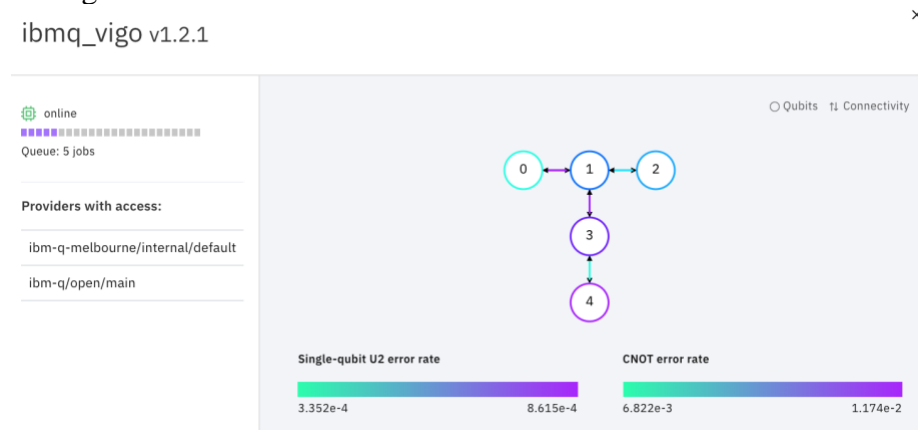

Choose a device which you will use during this session from the choice at the right, and examine the details of error rates on CNOT and single qubit gates. Note that it also indicates which CNOT gates are allowed on this device. Record this as a screenshot for reference.

Additional details can be found by clicking on the "Download Calibrations" button. Do this, and examine the file that you download. The T1 and T2 times quoted are an indication of how long qubits remain faithful to their state (i.e. stay coherent). T1 is related to how long it takes for say a $|1\rangle$ state to decay into a $|0\rangle$ state through natural relaxation processes (akin to an "X" error). T2 is related to how long it takes for the phase to change by 180° (akin to a "Z" error). We can program the computer and observe the effect of these processes (after repeating many times to build up the probability distributions).

9.3 Superposition states

We will make use of the Circuit Composer functionality in the IBM experience website. Click on the "Circuit Composer" icon in the left sidebar.
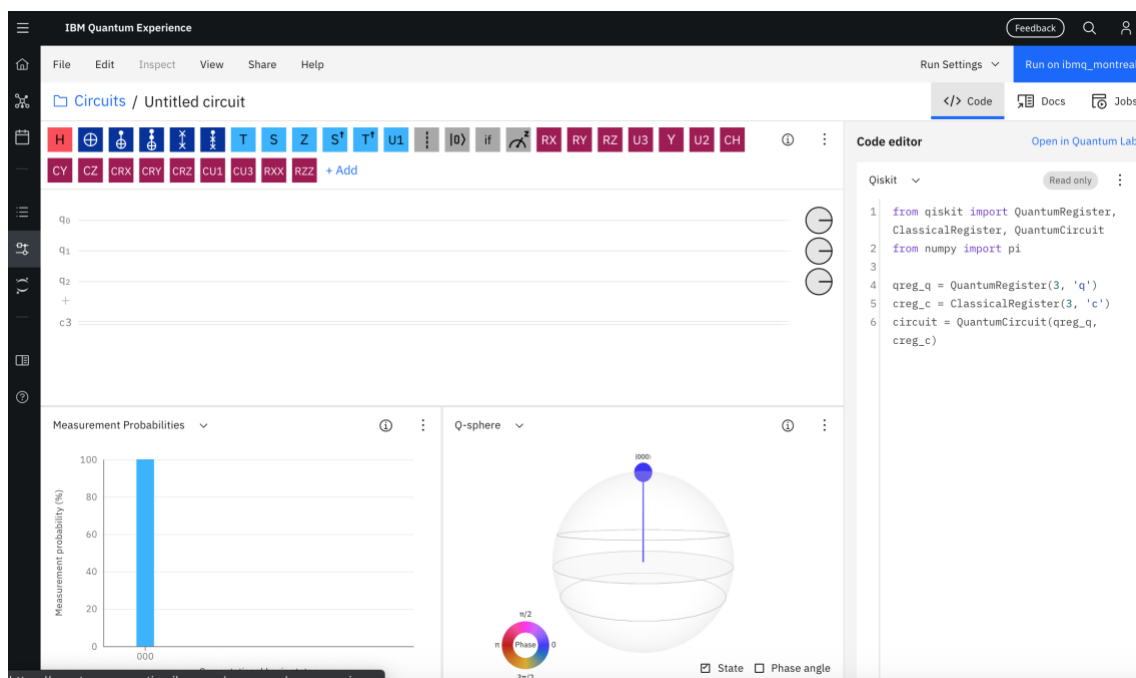
Exercise 9.3.1 Program an Hadamard gate to create an equal superposition state. Examine the different animations and information screens available to you, including the "Q-Sphere", measurement probabilities, and the code (both Qiskit and QASM) for the circuit. Be sure to also add a measurement gate to measure the outcome of the circuit. Your circuit should now look like this:
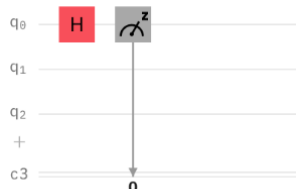

Note that you can add or remove qubits, by clicking on the qubit's name (q0, q1, q2...) or the plus sign.

Choose a suitable filename of "Hadamard" (or similar) by clicking on the filename in the top left hand corner.


Now let us execute the circuit. First, we will choose the device to run on. This can be chosen by selecting "Run Settings" in the top right. Choose an appropriate five-qubit device, and 1024 shots. Once you are happy with your settings, press the blue "Run on …" button.
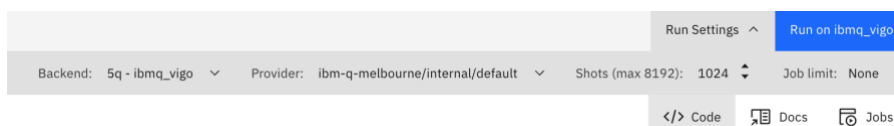
By clicking on the "Jobs" tab (below the blue "Run on…" button), you should see your job queued, and an estimate of the time required for it to run. This might be several minutes. You can queue more than one job at once, so it is possible to use this time to continue on in the lab either alone, or pooling resources with your classmates.

Once your job shows that is completed, you can examine the results by clicking on the job in the jobs pane. Note that you can see the probability of each measurement, the original circuit, and the "transpiled" circuit which actually ran on the machine. In this case you should get approximately 50/50%.
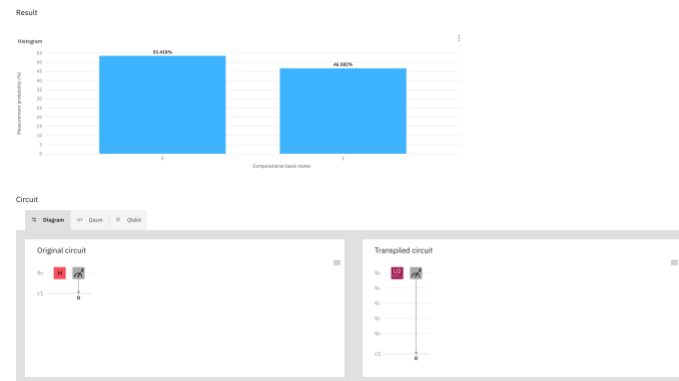
Now add a time delay by adding a lot of H gates, with barrier gates in between them to ensure they are not optimised away. You need dozens of H to see an effect, so don't waste too many units going small. Run this experiment, and see if you can see the effect of noise in your device.

Exercise 9.3.2 *Single qubit superposition decay.* Reset and program a sequence HTH from the $|0\rangle$ state to put it into a known superposition (see Lab 1, or check with the QUI). Add a measurement gate (and as before run the circuit and examine the output).

Exercise 9.3.3 Make a 5-qubit superposition over all binary numbers, run the simulator (by selecting the simulator as the backend) to check you understand what you will get out of the experiment and then run the experiment on a real quantum computing device.

Pause for a second and appreciate what you're doing here – you are programming real qubits, getting actual results, and understanding (hopefully) what's happening.

9.4 Entangled states

Let's now start generating some entanglement.

Exercise 9.4.1 *Bell state.* Reset and program a sequence to put two qubits into a Bell state (see Lectures and/or check with the QUI).

Exercise 9.4.2 *Cat state entanglement.* See if you can entangle all 5 qubits in a GHZ state, $|\psi\rangle = (|00000\rangle + |11111\rangle)/2$. In the QUI this would be generated by a circuit given by the following:
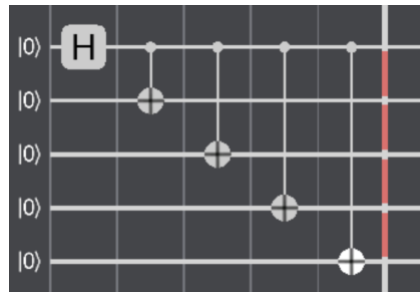
But on the real device you can't directly connect all qubits, and not necessarily in the directions you want. Pick the obvious qubits to act as the controls. Simulate to check you've programmed it correctly and then run the experiment. How did it go?

Exercise 9.4.3 See if you can understand the results from your experiments in terms of the errors quoted for the single and two-qubit gates, by programming a 5-qubit GHZ state circuit in the QUI with X, Y and Z control error gates applied (randomly) with commensurate error levels.

9.5 Algorithms

Now start programming some quantum algorithms and see how far you can push the machine.

Exercise 9.5.1 Program and run a (non-trivial) instance of Deutsch-Josza. Again, you can use the QUI to investigate and understand how well the machine performs.

Exercise 9.5.2 Program and run a two-qubit QFT_4 circuit. You will need to use a U gate corresponding to the rotation R_k and refer to Wednesday's lecture on how to create the controlled-U operation. Investigate how well the machine performs (and note the measurement bit ordering if you are comparing with the QUI).

Exercise 9.5.3 Can you program and successfully run a three-qubit QFT_8 circuit? Investigate and understand how well the machine performs.

Exercise 9.5.4 Can you program and run a Grover search over two qubits? Investigate and understand how well the machine performs, attempting to demonstrate an instance of Grover's algorithm achieving the highest fidelity possible.

Exercise 9.5.5 In the Bernstein-Vazirani problem we have a function $f = x \cdot s$ mod 2 which maps the product of two n-bit numbers to $\{0,1\}$. Program the following (oracle) function in the QUI for the case $s = 5$ and verify the table of results (e.g. setting the initial state into a superposition via a column of H on the x-register).
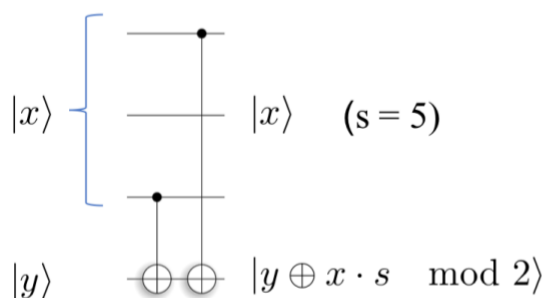
x	f(x)
000	0
001	1
010	0
011	1
100	1
101	0
110	1
111	0

Taking care to note the CNOT constraints in the physical system, can you implement BV algorithm for this function and obtain a probability distribution with sufficient precision to determine the value of *s* above? Implement this version of the BV algorithm and examine the result.