PHYC90045 Introduction to Quantum Computing

Lab Session 7

7.1 Introduction

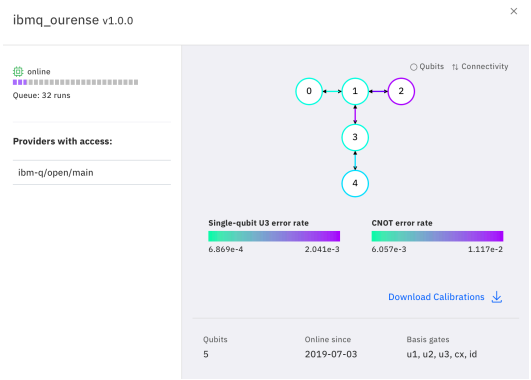
Welcome to Lab-7 of PHYC90045 Introduction to Quantum Computing. The purpose of this lab session is to program and run circuits in the IBM Quantum Experience.

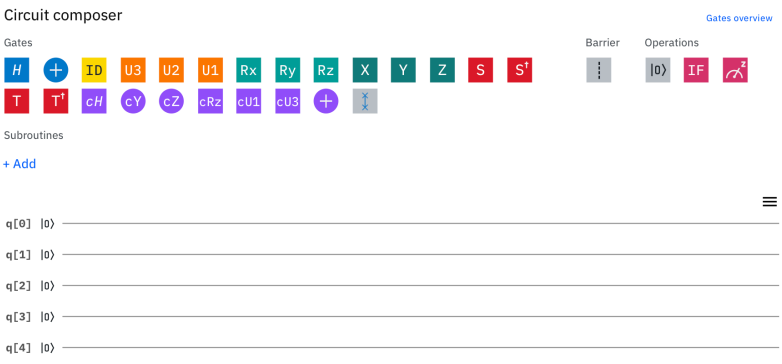
7.2 IBM Quantum Experience

The IBM Quantum Experience allows you program and run circuits on actual hardware, as described in the lectures. 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 (and unlike the QUI you have to put in measurement gates to get results for both the IBMQ machine and the simulator). Make sure to set the number of simulator runs to at least c.1000 so the results aren't skewed by low statistics.

Hints: if you're programming a lot of gates switch to the QASM editor (use copy/paste) 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! If "Ourense" has a busy queue, use the other publicly available machine "Melbourne". In this lab, working in groups will minimise wait times.

Exercise 7.2.1 When you access the system you will see a summary of the latest calibration of the available devices – e.g. see below for the IBMQ "Ourense".





Given the CNOT map, draw in the allowed CNOT gates above, and in the circuit composer itself verify that you understand where CNOTs can and can't be placed.

7.3 Single qubit gates and superposition states

If you download the calibration for Ourense you will get a table with all the relevant parameters for the single qubits and gates and readout errors (below is shown for the calibration 11/09/2019):

Qubit	T1 (¬µs)	T2 (¬µs)	Frecuency (GHz)	Readout error	Single-qubit U3 error rate	CNOT error rate
Q0	92.7666972	77.7127691	4.81946567	1.90E-02	7.17E-04	cx0_1: 6.057e-3
Q1	95.4213466	32.7649079	4.891104804	3.70E-02	6.87E-04	cx1_0: 6.057e-3, cx1_2: 1.017e-2, cx1_3: 1.117e-2
Q2	77.1507523	78.5837611	4.716892921	9.40E-02	2.04E-03	cx2_1: 1.017e-2
Q3	93.6711212	84.7347415	4.789036721	4.00E-02	7.82E-04	cx3_1: 1.117e-2, cx3_4: 7.104e-3
Q4	54.7021725	40.5174242	5.023696923	4.60E-02	9.48E-04	cx4_3: 7.104e-3

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 a "X" error we encountered in Lab 6). 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). A key operation for observing these effects is the identity (ID) gate. An identity gate does not actually perform any physical operation on the qubit, but forces it to sit idle for a certain period of time. These can be concatenated together in order to observe the decay of the qubit's state.

Important: After each gate you program in the decay experiments, add in a 'barrier' gate on the same qubit. This ensures that the compiler does not combine different gates together, meaning that each gate will correspond to its own time-step.

Exercise 7.3.1 Single qubit basis state decay. Pick the qubit with the shortest T1 and T2 (if obvious for the day's calibration). Program an X-gate to flip the qubit to the $|1\rangle$ state. Now add a time delay by adding a lot of ID gates – if there is no decoherence you should end up still in the $|1\rangle$ state. You need dozens of IDs to see an effect, so don't waste too many units going small. And of course the "experiment" will need to run the circuit many times (default = 1024 runs) to obtain the approximation to the probability distribution.

Exercise 7.3.2 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). Again add a time delay by adding a lot of IDs gates – if there is no decoherence you should end up still in the original superposition. You need dozens of H to see an effect, so don't waste too many units going small.

Exercise 7.3.3 Make a 5-qubit superposition (or higher if you are running on the "Melbourne" machine) over all binary numbers, run the simulator to check you understand what you will get out of the experiment and then run the experiment.

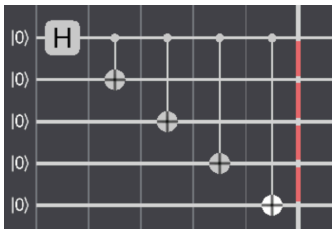
Pause for a second and appreciate what you're doing here – you are programming real qubits, getting actual results, and understanding (hopefully) what's happening.

7.4 Entangled states

Let's now start generating some entanglement.

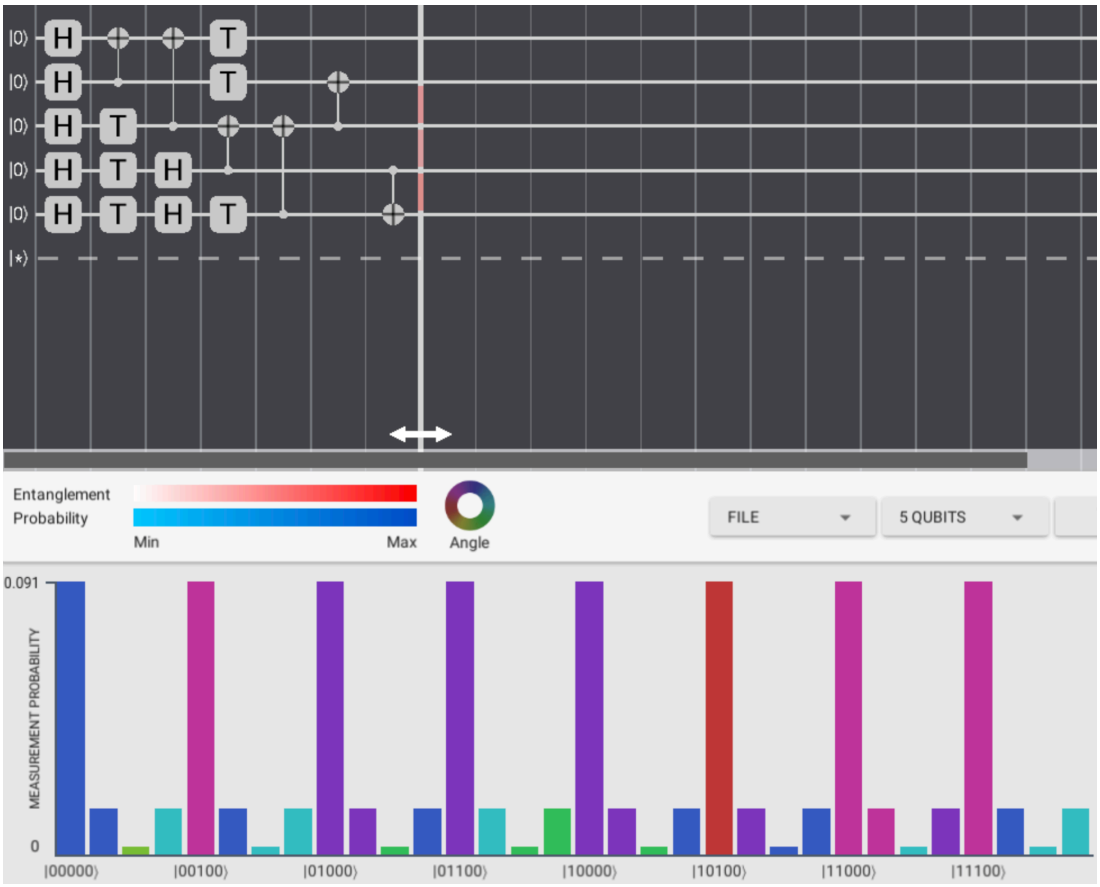
Exercise 7.4.1 Entanglement decay. Reset and program a sequence to put two qubits into a Bell state (see Lectures and/or check with the QUI). Again add a time delay by adding a lot of ID gates and see how the entangled state decays.

Exercise 7.4.2 Big entanglement. Try to entangle all 5 qubits in a so called "GHZ state", $|\psi\rangle = (|00000\rangle + |11111\rangle)/2$. In the QUI this would be generated by the circuit on the right:



However, on the real device you can't directly connect all qubits, and not necessarily in the directions you want. Pick the obvious qubit to act as the control and refer to Lecture 14 to see how you invert direction of a CNOT when required. Simulate to check you've programmed it correctly and then run the experiment. How did it go?

Exercise 7.4.3 Program the IBMQ machine to run a non-trivial circuit (e.g. see below) based on the CNOT map of the actual IBMQ machine and compare with the QUI output.



Exercise 7.4.4 See if you can understand the results from the IBMQ machine 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 control errors represented by X, Y and Z error gates (see Lab 6) applied (randomly) with commensurate error levels (e.g. 0.01 – 0.10 in units of π).

7.5 Algorithms

Now we will start programming some quantum algorithms and see how far you can push the IBMQ machines. Again, you can use the QUI to investigate and understand how well the machine performs. Where you can, repeat the circuit using different qubits on the IBMQ5 machine taking note of which are more or less precise according to the calibration.

Exercise 7.5.1 Program and run dense coding circuits for the four classical bit-send scenarios.

Exercise 7.5.2 Program and run a (non-trivial) instance of Deutsch-Josza.

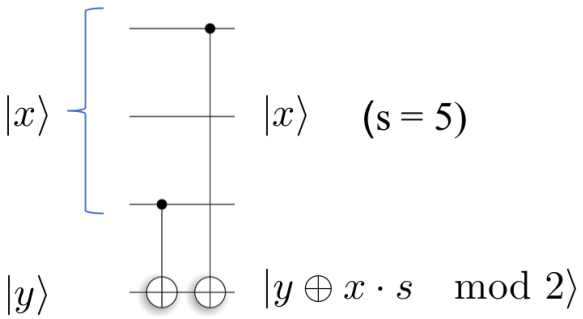
Exercise 7.5.3 Program and run a two-qubit QFT_4 circuit. You will need to use a U gate corresponding to the rotation R_k and refer to Lecture 14 on how to create the controlled-U operation. Note the measurement bit ordering if you are comparing with the QUI.

Exercise 7.5.4 Can you program and successfully run a three-qubit QFT_8 circuit? Investigate and understand how well the machine performs.

Exercise 7.5.5 Can you program and run a Grover search over two qubits? Investigate and understand how well the machine performs.

Exercise 7.5.6 What's involved in going to 3 bit Grover search?

Exercise 7.5.7 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

Now for the IBMQ5. Taking care to note the CNOT constraints in the physical system, can you implement this circuit and obtain a probability distribution with sufficient precision to verify the table above? You change the parameter s if that makes it easier, but you have re-compute (classically) the table for comparison (or let the QUI do it).

7.6 Depth and complexity

We will now try to repeat the analysis in 7.4.4, but for more complex circuits.

Exercise 7.6.1 Program random circuits (single and two qubit gates) of increasing depth and complexity. Explore the outputs by comparing with the QUI for the same circuit with and without simulated error gates.
