PHYC90045 Introduction to Quantum Computing

Lab Session 6

6.1 Introduction

Welcome to Lab-6 of PHYC90045 Introduction to Quantum Computing.

The purpose of this lab session is to:

- understand and implement various error types in quantum circuits
- program and run quantum supremacy circuits
- implement a simple case of randomised benchmarking

6.2 Rotation errors in the QUI

One of the key goals of this subject is to provide students with experience in programming an actual quantum computer. However, in the real-world the digital logic of qubits is prone to "errors" due to noise in the control systems and/or the immediate environment of the qubit itself – in order to understand and appreciate the results when we access quantum computer hardware we need to study these effects. While the quantum computer simulator powering the QUI is effectively a pristine qubit environment (we have set the errors to zero!), we can introduce such effects systematically and investigate how quantum gate errors affect the output of quantum circuits. Consideration of quantum errors is particularly important when we study quantum supremacy circuits as the real world is not pristine, and the effect of errors on quantum gates must be taken into account in the determination of the quantum supremacy point.

In what follows, we will represent rotation errors around the cartesian axes in the QUI using the R-gate. For example, a Z-rotation error (or just "Z-error") is a gate δZ defined as:

$$\delta Z \equiv \begin{pmatrix} e^{-i\epsilon/2} & 0 \\ 0 & e^{i\epsilon/2} \end{pmatrix}$$

where the level of error is governed by the angle ε (assumed to be small). We can implement this effective error gate in the QUI (c/f QFT) as follows:

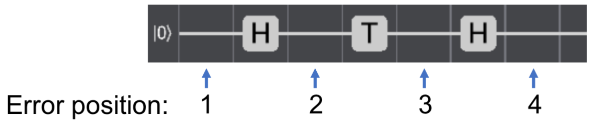
$$R_Z(\epsilon) = e^{i\theta_g}\big|_{\theta_g=0}\left(I\cos\frac{\epsilon}{2} - iZ\sin\frac{\epsilon}{2}\right) = \begin{pmatrix} e^{-i\epsilon/2} & 0 \\ 0 & e^{i\epsilon/2} \end{pmatrix}$$

Similarly for X-error and Y-error gates so we have for the "Pauli" rotation error gates:

$$\delta X = R_X(\epsilon)\,, \quad \delta Y = R_Y(\epsilon)\,, \quad \delta Z = R_Z(\epsilon)$$

with global phases zero. We could define more general errors, but these will suffice.

Exercise 6.2.1 Let's go back to our favourite 3-gate (interference) example, HTH, and examine the effect of rotation errors as a function of where they occur:



For definiteness, we will take the error angle to be 0.01π (i.e. ~1% of a X, Y or Z gate).

a) X-errors. Program the _H_T_H_ circuit in the QUI (with appropriate spaces for the X-errors). Create a R-gate with the rotation angle set to the error angle $\varepsilon = 0.01\pi$ (global phase set to zero per the above definitions). Put the error gate in the various locations and examine the effect on the "pristine" no-error output by filling in the table below:

Table 1: effect of X-errors (1%) on HTH

X Error position	Output state $\vert\psi\rangle = a_0\vert 0\rangle + a_1\vert 1\rangle$	Output state change	Output state relative change
None	$p_0\vert_{\text{exact}} = 0.854$ $p_1\vert_{\text{exact}} = 0.146$	$\Delta p_0 = 0$ $\Delta p_1 = 0$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} = 0$ $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} = 0$
1	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} = \quad \%$ $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} = \quad \%$
2	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} =$ $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} =$
3	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} =$ $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} =$
4	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} = \quad \%$ $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} = \quad \%$

Look at the data in the table – what do you conclude?

b) Repeat for the case of Y-errors.

Table 2: effect of Y-errors (1%) on HTH

Y Error position	Output state $\lvert\psi\rangle = a_0\lvert 0\rangle + a_1\lvert 1\rangle$	Output state change	Output state relative change
None	$p_0\rvert_{\text{exact}} = 0.854$ $p_1\rvert_{\text{exact}} = 0.146$	$\Delta p_0 = 0$ $\Delta p_1 = 0$	$\frac{\Delta p_0}{p_0\rvert_{\text{exact}}} = 0$ $\frac{\Delta p_1}{p_1\rvert_{\text{exact}}} = 0$
1	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\rvert_{\text{exact}}} =$ % $\frac{\Delta p_1}{p_1\rvert_{\text{exact}}} =$ %
2	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\rvert_{\text{exact}}} =$ % $\frac{\Delta p_1}{p_1\rvert_{\text{exact}}} =$ %
3	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\rvert_{\text{exact}}} =$ % $\frac{\Delta p_1}{p_1\rvert_{\text{exact}}} =$ %
4	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\rvert_{\text{exact}}} =$ % $\frac{\Delta p_1}{p_1\rvert_{\text{exact}}} =$ %

Look at the data in the table – what do you conclude?

c) Repeat for the case of Z-errors.

Table 3: effect of Z-errors on HTH

Z Error position	Output state $\lvert\psi\rangle = a_0\lvert 0\rangle + a_1\lvert 1\rangle$	Output state change	Output state relative change
None	$p_0\vert_{\text{exact}} = 0.854$ $p_1\vert_{\text{exact}} = 0.146$	$\Delta p_0 = 0$ $\Delta p_1 = 0$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} = 0$ $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} = 0$
1	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} =$ $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} =$
2	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} =$ % $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} =$ %
3	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\frac{\Delta p_0}{p_0\vert_{\text{exact}}} =$ % $\frac{\Delta p_1}{p_1\vert_{\text{exact}}} =$ %
4	$p_0 =$ $p_1 =$	$\Delta p_0 =$ $\Delta p_1 =$	$\Delta p_0/p_0\vert_{\text{exact}} =$ $\Delta p_1/p_1\vert_{\text{exact}} =$

Look at the data in the table – what do you conclude?

Exercise 6.2.2 Gate errors and CNOT gates. Program a CNOT gate and experiment with the placement of one error gate in the four possible locations, with various error angles $\varepsilon = 0.01 - 1.0\ \pi$. What do you notice?

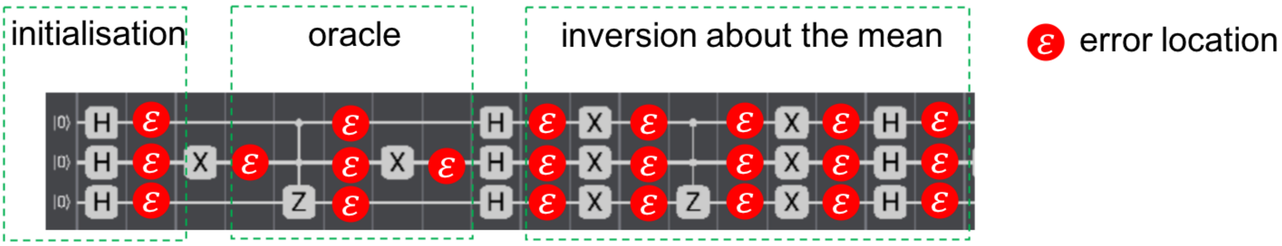
6.3 Effect of rotation errors in quantum circuits

Obviously, not all errors have the same effect – it will depend on the type of error (X, Y, Z, or other effective axes of rotation) and where it occurs in the circuit. The effect of the error also depends on the circuit itself. We will now investigate how the outputs of some of our previously coded quantum algorithms (hopefully saved!) are affected by such rotation errors.

Exercise 6.3.1 Error position experiment. Load your file "Lab4 Grover 3 oracle marks 5", run it and re-familiarise yourself with how it works and the output. Set the circuit to compute the first two iterations only – to check: the probability of finding the target state 5=101 is 94.5%. Use the "insert time block" to create spaces between all gates, save the file as "Lab6 Grover spaces for errors".

- Create a X-error at the 10% level ($\varepsilon = 0.1\pi$). We have chosen 10% so you can easily observe the effects. By experimenting with the placement, find the locations which have minimum and maximum effect on the circuit outcome.
- Repeat for Z and Y errors independently (also at the $\varepsilon = 0.1\pi$ level).

Exercise 6.3.2 Random errors experiment. For a given quantum circuit we will assume a rotation error can occur during each operation, i.e. for the first iteration (including the initialisation) of the Grover circuit we have:



For each location place an error R-gate, $R_E(\varepsilon)$, chosen at random from the set {error type E = (X, Y, or Z), $\varepsilon = \pm 0.03\pi$} (note we have included the possibility of over or under rotations through the sign of ε). You can seed the editing by putting in a column of X, Y and Z errors in the initialisation next to the Hadamards and use the QUI copy-paste feature to quickly populate all other locations, from there mix the gates and the rotation angles. Run the circuit and investigate how the output probability is affected by these errors. Change around the mix of errors and the values of ε.

Exercise 6.3.3 Repeat for the 3-qubit QFT adder circuit. What is the maximum rotation error level you would tolerate in the adder?

6.4 Types of errors: control precision and decoherence

The ways in which qubits and quantum gates can be disrupted is still an area of active research. Broadly, a quantum computer can be affected by imprecise control, or by stray interactions with the environment (i.e. decoherence), or both. In the previous sections the inclusion of rotation errors were closely related to the sorts of effects produced by control errors. Here we will look at the other type of error – those produced by decoherence. Typically, these are modelled as a complete "Pauli" gate ($\varepsilon = \pi$) occurring at random in the circuit with some probability p per time step. If we have a circuit with M possible error locations the total number of errors N_e that can occur for each run is given by the product $N_e = p\,M$. We interpret this as each time we run the quantum circuit we effectively have N_e extra X, Y or Z gates occurring at random locations, and which are different every time we run the circuit.

Exercise 6.4.1 Mimicking decoherence experiment. Load the Grover circuit "Lab6 Grover spaces for errors", which has all error locations free over two iterations. The number of error locations is 43. If we set the probability of an error per time step to be 3% (typically where the hardware is at), then the total number of errors in our circuit will be roughly 0.03x43~1. So, each time we run the Grover circuit an error in the form of an extra X, Y or Z gate will occur somewhere in the circuit. In the QUI, since we will move the error gates around define these as R-gates (to distinguish from the circuit gates needed by the Grover search) with a full π rotation and global phase set appropriately for X, Y and Z. Put measurements at the end of the circuit. Now you're ready to experiment.

1. Create a full X error R-gate and move it to an error location at random
2. Run the circuit and record the measurement outcome in the table below
3. Move the error to some other location at random
4. Goto step 2 and repeat for a total of 33 times

5. Create a full Y error R-gate and move it to an error location at random
6. Goto step 2 and repeat for a total of 33 times

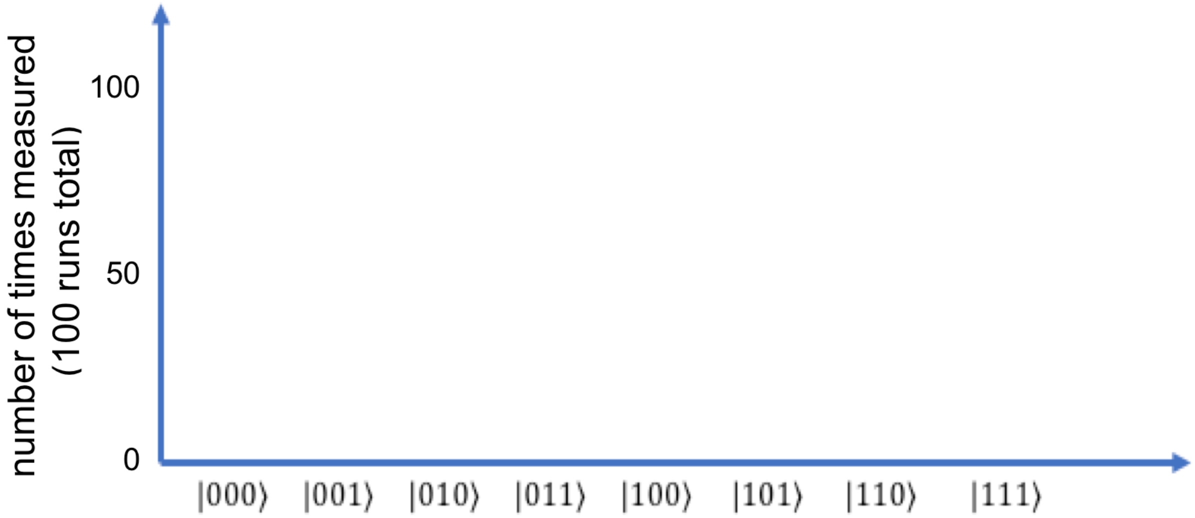
7. Create a full Z error R-gate and move it to an error location at random
8. Goto step 2 and repeat for a total of 34 times

Table of results:

Measurement outcome	Record ~~||||~~...etc	Total
000		
001		
010		
011		
100		
101		
110		
111		

Now plot the results in the histogram below.

Grover's algorithm (3 qubits, search on 5=101), prob error = 3%

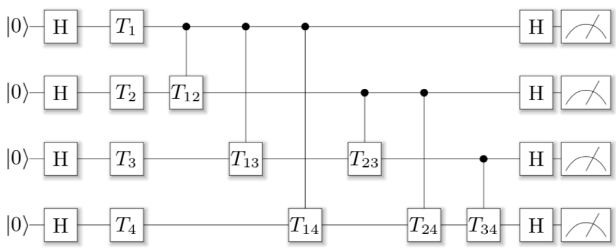


From these results estimate the probability of the algorithm producing a correct result in the presence of the 3% error rate.

Exercise 6.4.2 Repeat the procedure and analysis for circuits you have programmed previously (e.g. QFT adder, Baby Shor etc) and examine the effect on the outputs (e.g. different effective error probabilities).

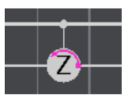
6.5 Quantum supremacy – IQP circuits

The basic IQP circuit scheme, and the relation to QUI gates, is shown below:



$$T_m = \cos\left(\frac{k_m \pi}{8}\right) I + i \sin\left(\frac{k_m \pi}{8}\right) Z_m \quad \to R_Z\left(\frac{-k_m \pi}{4}\right)\Big|_{\theta_g=0}$$

$$T_{mn} = \cos\left(\frac{k_{mn} \pi}{8}\right) I + i \sin\left(\frac{k_{mn} \pi}{8}\right) Z_m Z_n \;\to \boxed{R_Z\left(\frac{-k_{nm} \pi}{4}\right)\Big|_{\theta_g=0}}$$



$$\frac{-k_n \pi}{4}, \frac{-k_{nm} \pi}{4} \in \{0, -0.25, -0.50, -0.75, -1.00, -1.25, -1.50, -1.75\}\pi$$

Exercise 6.5.1 Code up a small-scale IQP circuit in the QUI (generalised from the schematic above) using random selections from the set of arguments of the R-gates. The QUI isn't set up (yet) to do a full analysis of the effect of errors on sampling from the IQP distribution. However, you can run different instances (i.e. changing the values of the k's) and introduce control and/or decoherence errors and watch the probability distribution change. It's reasonably straightforward to program and investigate 10-qubit instances (1024 states).

6.5 Randomised benchmarking

(NB. completely optional – it's a bit tedious in the present form of the QUI)

We will now consider some randomised benchmarking examples using the QUI. Below is an example of a 5-gate sequence X-H-SQRT(Y)-S-Z (SQRT(Y) = RY(π/2) etc) followed by its inverse. Notice the use of the R-gate to define some of the gates (why?). The schematic also indicates the error positions interleaved in the sequence.

Exercise 6.5.1 We don't know if this will work in the QUI (i.e. gather enough statistics). Program a random 5-gate sequence (with spaces as shown) using gates chosen from the set {X, Y, Z, H, RX(π/2), RY(π/2) RZ(π/2)} and run to check it produces the |0⟩ state at the end (i.e. that the inverse sequence is correct). For each error location place an error R-gate, $R_E(\varepsilon)$, chosen at random from the set {error type E = (X, Y, or Z), ε = ±0.1π}. Run the sequence and record the probability of measuring the |0⟩ state. Edit the gate sequence to produce another random 5-gate sequence (m=5), leaving the errors alone for simplicity (a static control error assumption). Run and record the probability of measuring the |0⟩ state. Keep repeating until you have enough statistics to calculate an average over the probabilities (i.e. $F_{m=5}$). Repeat all of that for an 8-gate sequence to obtain $F_{m=8}$. You've now got two data points (m=5 and m=8) to fit to the form given in the lectures:

$$F_m = A + Bf^m$$

i.e. determine *A* and *B* and *f* (assume m=0 implies A + B = 1). From this value of *f* , calculate average fidelity of the gates from:

$$F_{av} = \frac{f+1}{2}$$

and compare with what you expect from the values of ε used.