MULT90063 Introduction to Quantum Computing

Lab Class 1

Welcome to Lab Class 1 of MULT90063 Introduction to Quantum Computing.

The purpose of this first tutorial class is to:

- Review your knowledge of complex numbers.
- Start using the Quantum User Interface (QUI).
- Understand complex amplitudes and their representation in the QUI system.
- Review matrices and vectors

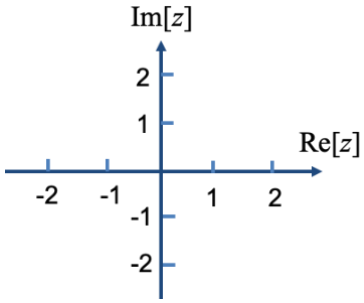
Refresher on complex numbers

Consider complex numbers $z = x + iy$ where x and y are ordinary real numbers, and $i^2 = -1$. The real and imaginary parts of z are Re[z] = x and Im[z] = y respectively.

Question 1.1 For the complex number $z = 3 + 2i$ determine the real and imaginary parts:

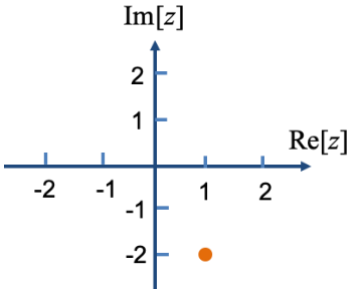
Re[z] = _____ and Im[z] = _____ .

Question 1.2 Plot the complex number $z = -1.5 + i$ in the complex plane.



Question 1.3 Write out $z = x + iy$ for the point shown (right):

z = _____ + i _____



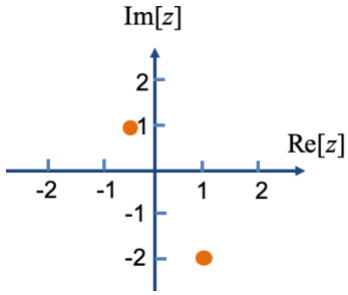
Consider two complex numbers: $z_1 = x_1 + iy_1$, and $z_2 = x_2 + iy_2$ where $\{x_1, y_1, x_2, y_2\}$ are real numbers. Addition and multiplication is as follows:

$$z_1 + z_2 = x_1 + iy_1 + x_1 + iy_1 = (x_1 + x_2) + i(y_1 + y_2)$$

$$z_1 z_2 = (x_1 + iy_1)(x_2 + iy_2) = x_1 x_2 + ix_1 y_2 + iy_1 x_2 - y_1 y_2$$

Question 1.4 Given $z_1 = -2 + i$ and $z_2 = 3 - 2i$, determine $z = z_1 + z_2$ and plot:

$z =$ _____ $+ i$ _____



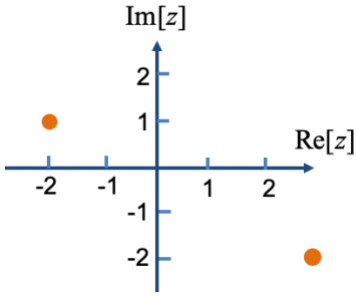
Question 1.5 Given $z_1 = -1/2 + i$ and $z_2 = 1 - 2i$, determine $z = z_1 z_2$ and plot:

$z =$ _____ $+ i$ _____

The "conjugate" z^* of a complex number z is formed by changing the sign of i, i.e. $z = x + iy \rightarrow z^* = x - iy$. The product $z\,z^* = (x + iy)(x - iy) = x^2 - ixy + iyx + y^2 = x^2 + y^2$. The magnitude of a complex number is denoted $|z|$ and is given by $|z| = \sqrt{x^2 + y^2}$. The magnitude squared of the complex number is given by: $z\,z^* = x^2 + y^2 = |z|^2$.

Question 1.6 For $z = -2 + i$ determine the magnitude $|z|$ and $|z|^2$.

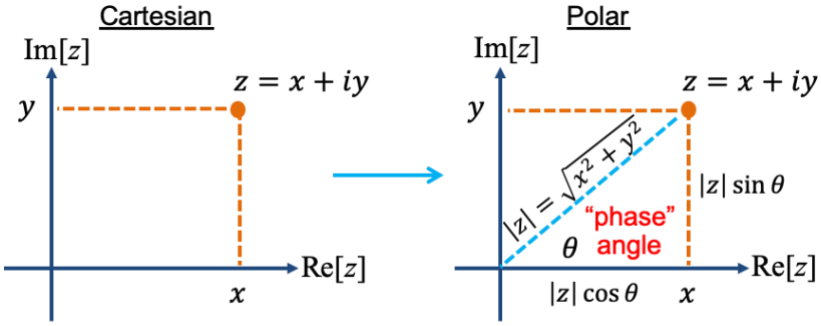
$|z| =$ ___ , $|z|^2 =$ ___ .



Question 1.7 Given $z = 3 - 2\,i$ determine the magnitude $|z|$ and $|z|^2$.

$|z| =$ _____ , $|z|^2 =$ _____ .

In the QUI, we use "polar notation" (angle) as a compact way to visualise complex numbers.

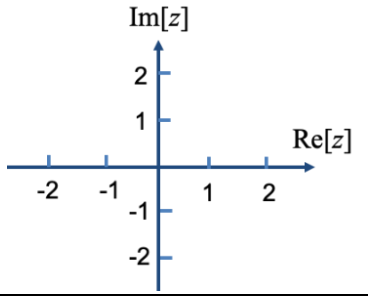


$z = x + iy$ $\qquad$ $z = |z|\cos\theta + i|z|\sin\theta$

$$z = |z|\cos\theta + i\,|z|\sin\theta = |z|(\cos\theta + i\sin\theta)$$

$$\rightarrow z = |z|e^{i\theta}$$

1.8 Plot $z = 1 + i$ on the complex plane, convert to polar notation, and label plot with magnitude and angle:



In QC we often measure angles in radians rather than degrees.

1.9 Use the conversions to fill in the table.

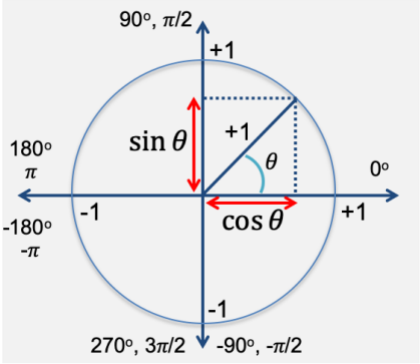
degrees	radians
18	
30	
	π/3
	-π/4

Conversion:

θ in rad = [π/180] x (θ in deg)

θ in deg = [180/π] x (θ in rad)

To understand the representation of quantum amplitudes, it's very useful to know how to compute sin and cos quickly around the unit circle (unit means radius 1):



Angle θ (radians)	cos θ	sin θ
0	1	0
π/4	$1/\sqrt{2}$	$1/\sqrt{2}$
π/2	0	1
π	-1	0
3π/2	0	-1
2π	1	0
-π/2	0	-1
-π	-1	0

The Quantum User Interface (QUI)

The QUI is a web-based graphical user interface (developed by the Hollenberg group at the University of Melbourne) to program, simulate and analyse quantum circuits. The QUI allows the users to specify qubit number, build quantum circuits, simulate and examine the quantum state at every time step in the circuit/program. The latter feature is critical to understanding QC, and distinguishes QUI from other on-line programming/simulation tools.

The QUI is accessed through a web-based interface (quispace.org – click on blue QUI logo). See the notes "QUI Intro" on LMS for quick guide. *If you haven't signed up already please let a demonstrator know and follow these steps.*

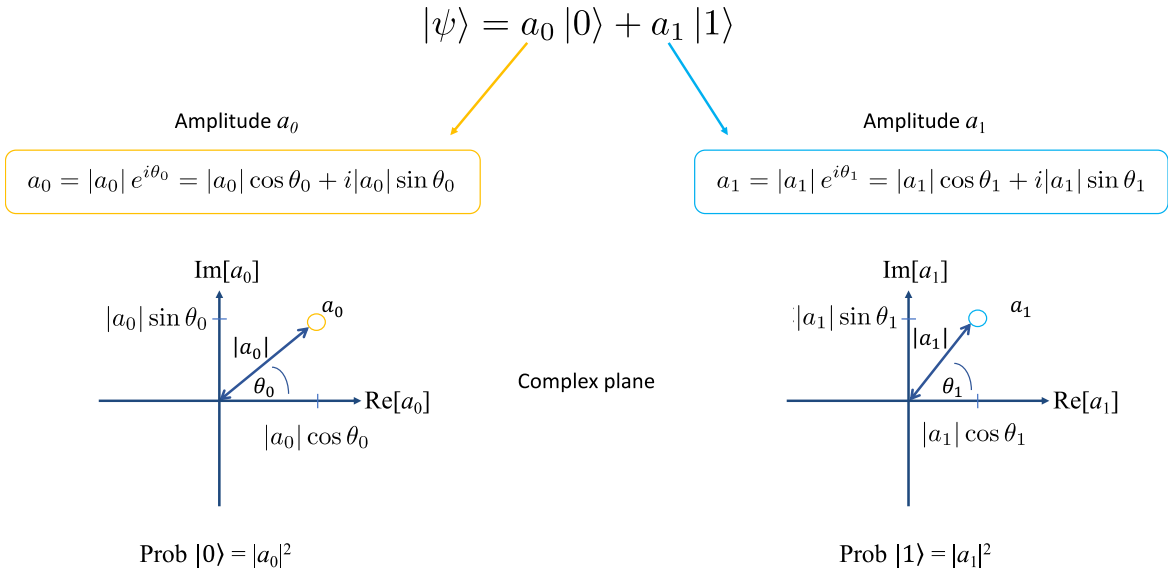
Step 1: Open a web browser (preferably Google Chrome or Firefox), go to quispace.org.

Step 2: You will need to create an account to access QUI for the first time. In order to access expanded capabilities of the QUI you must use your **University of Melbourne email address** as your login name. Follow the steps to create your account.

Step 3: Once you have signed-up, start the QUI.

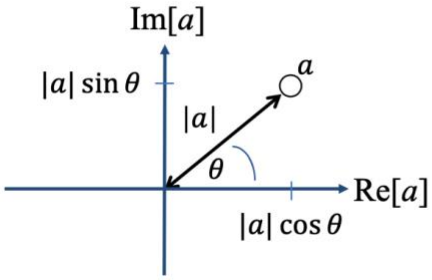
Quantum States, Amplitude, Probability and Phase

In general, a quantum superposition for a single qubit is written as $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$. The "state amplitudes" a_0 and a_1 are complex numbers. In the QUI we use polar notation, $|a|e^{i\theta}$, to describe the magnitude ($|a|$) and phase (θ) of the state amplitudes:

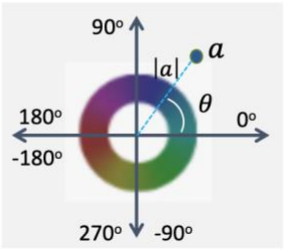


For a given amplitude $a = |a|e^{i\theta}$, QUI uses a colour wheel for the phase angle θ.

$$a = \mathrm{Re}[a] + i\,\mathrm{Im}[a] = |a|e^{i\theta}$$



QUI



degrees	radians
0	0
90	$\pi/2$
180	π
360	2π

As per lecture notes, the State Info Card (SIC) in the QUI gives the value of the complex amplitude a_i of a given state component $|i\rangle$ is given by $a_i = |a_i|e^{i\theta_i}$ where $|a_i|$ is the magnitude, and θ_i is the phase angle (colour wheel scale). The probability of measuring the state $|i\rangle$ is $|a_i|^2$.

Exercise 3.1 Consider the computational states $|0\rangle$ and $|1\rangle$ in the QUI.

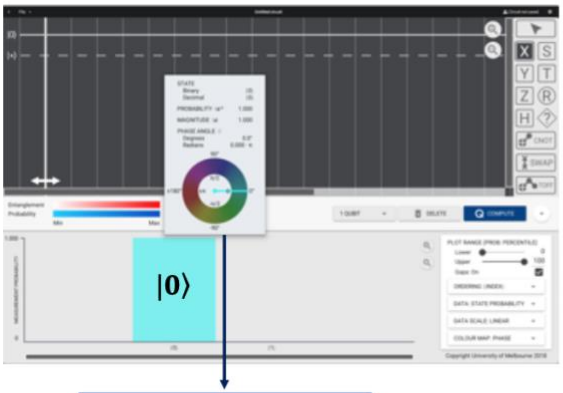
Select 1 qubit, and press compute (nothing in the circuit as per the schematic below). The system has been initialised in the $|0\rangle$ state. Mouse-over the histogram in the results panel (panel below the circuit) to bring up the State Info Cards (SICs) with the amplitude phase and magnitude values:

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle \rightarrow \text{initialized in } |0\rangle$$

$$\text{i.e. } a_0 = 1,\ a_1 = 0$$

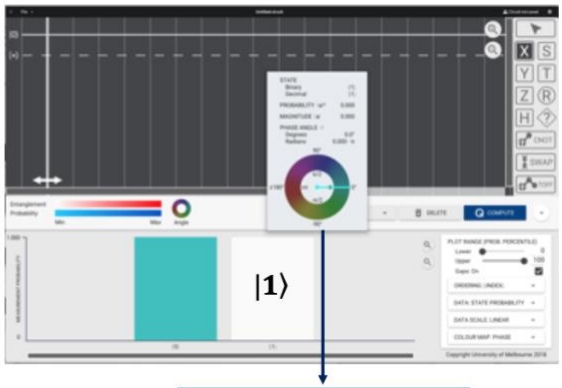
$$a_0 = |a_0|\cos\theta_0 + i\,|a_0|\sin\theta_0 = 1 \rightarrow \begin{cases} \theta_0 = 0 \\ |a_0| = 1 \end{cases}$$

$$a_1 = |a_1|\cos\theta_1 + i\,|a_1|\sin\theta_1 = 0 \rightarrow \begin{cases} \theta_1 = 0 \\ |a_1| = 0 \end{cases}$$



SIC info for a_0

$\theta_0 = 0$
$|a_0| = 1$
$|a_0|^2 = 1$



SIC info for a_1

$\theta_1 = 0$
$|a_1| = 0$
$|a_1|^2 = 0$

Question: What are the probabilities of measuring the result "0" and "1" respectively?

Prob["0"] = _____ : Prob["1"] = _____

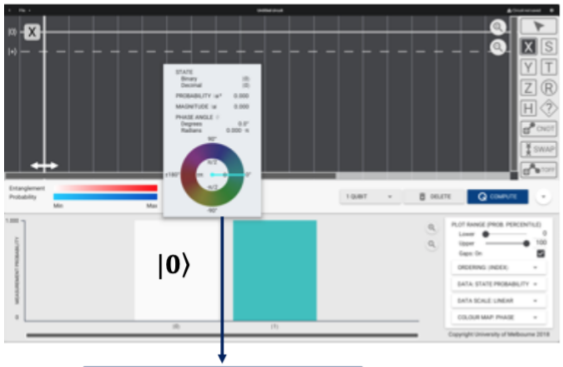
Now we will put the system into the $|1\rangle$ state. Click on the X-gate in the gate library on the right, and click it into the first time block as shown in the schematic below. Press compute. The X-gate flips the state of the qubit from $|0\rangle$ (as per the above – QUI qubits always initialised in $|0\rangle$) so qubit has been put into the $|1\rangle$ state. Mouse-over the histogram in the results panel (panel below the circuit) to bring up the State Info Cards (SICs) with the amplitude phase and magnitude values:

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle \rightarrow \text{initialized in } |1\rangle$$

$$\text{i.e. } a_0 = 0,\ a_1 = 1$$

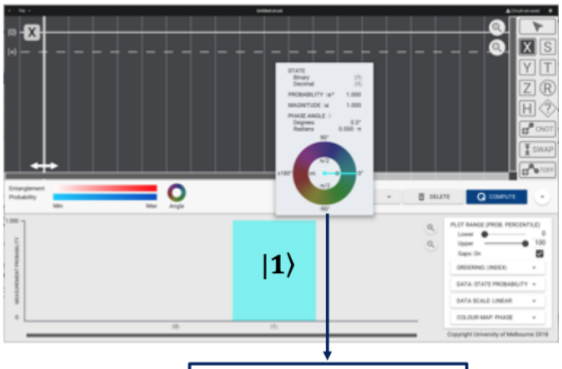
$$a_0 = |a_0|\cos\theta_0 + i\,|a_0|\sin\theta_0 = 0 \rightarrow \begin{cases} \theta_0 = 0 \\ |a_0| = 0 \end{cases}$$

$$a_1 = |a_1|\cos\theta_1 + i\,|a_1|\sin\theta_1 = 1 \rightarrow \begin{cases} \theta_1 = 0 \\ |a_1| = 1 \end{cases}$$



SIC info for a_0

$\theta_0 = 0$
$|a_0| = 0$
$|a_0|^2 = 0$



SIC info for a_1

$\theta_1 = 0$
$|a_1| = 1$
$|a_1|^2 = 1$

Question: What are the probabilities of measuring the result "0" and "1" respectively?

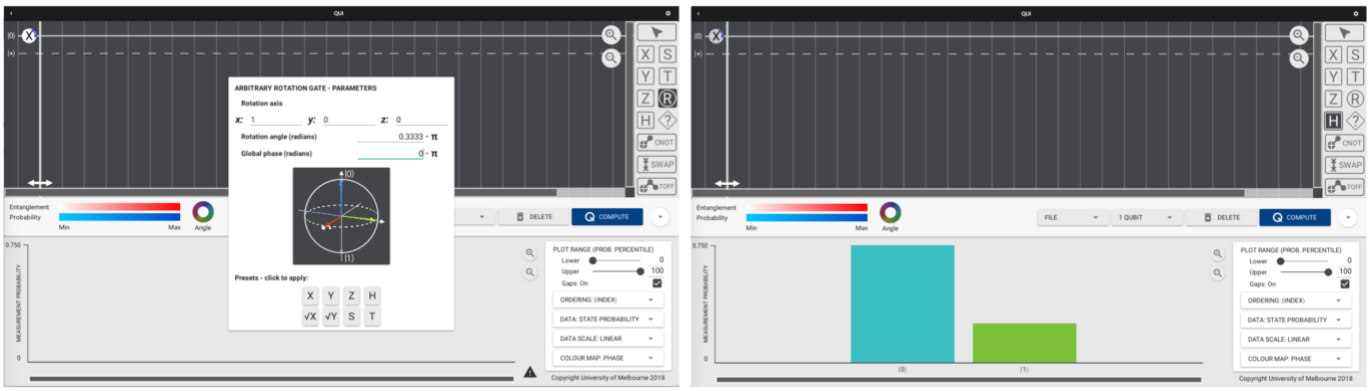
Prob["0"] = _____ : Prob["1"] = _____

Now we will set up a non-trivial quantum superposition state in the QUI. Set to 1-qubit (if not already) and clear any existing circuit.

Exercise 3.3 To construct our example state, start from a blank 1-qubit circuit and choose the R-gate from the gate library (for now, don't worry about what it is or how it works). Click the R-gate into the first time block. Once placed in the circuit **right click** on the R-gate in the circuit to bring up the editable gate menu:

Edit Parameters -> set axis to X, rotation angle to $\theta_R = \frac{\pi}{3}$, and global phase to zero.

Press compute and the QUI output will look like the following (bottom right):



Don't worry how this gate works for now, we will just look at the final output state to further illustrate the QUI notation for the individual complex amplitudes.

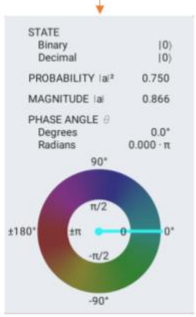
The overall quantum state at the end of this circuit is a superposition $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$. We will now inspect the individual state amplitudes in the SICs.

Question: In the following, convert from the QUI polar notation for the complex amplitudes to cartesian as indicated.

Overall quantum state: $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$

QUI representation of complex amplitude a_0 in polar form:

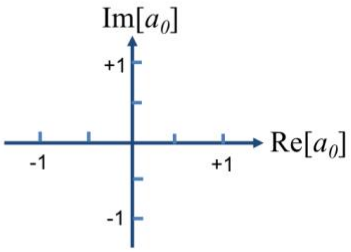
$|a_0|e^{i\theta_0}$



Convert amplitude a_0 to cartesian form and plot:

$a_0 = |a_0|e^{i\theta_0} = \mathrm{Re}[a_0] + i\mathrm{Im}[a_0]$

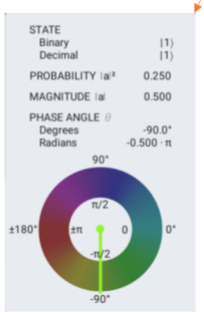
$a_0 =$ _____ $+ i$ _____



Overall quantum state: $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$

QUI representation of complex amplitude a_1 in polar form:

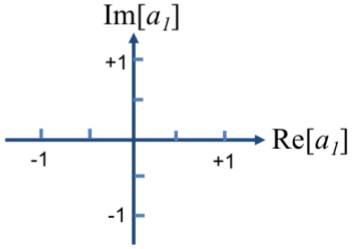
$|a_1|e^{i\theta_1}$



Convert amplitude a_1 to cartesian form and plot:

$a_1 = |a_1|e^{i\theta_1} = \mathrm{Re}[a_1] + i\mathrm{Im}[a_1]$

$a_1 =$ _____ $+ i$ _____



Question: Verify that the state created is:

$$|0\rangle \rightarrow \frac{\sqrt{3}}{2}|0\rangle + \frac{-i}{2}|1\rangle = \qquad |0\rangle + \qquad |1\rangle$$

In above space provided write the complex amplitudes in polar form, $|a_0|e^{i\theta_0}|0\rangle + |a_1|e^{i\theta_1}|1\rangle$.

Question: What are the respective probabilities of measuring "0" and "1" states?

Prob["0"] = _____ : Prob["1"] = _____

Exercise 3.4 Measurement on the QUI. On the same circuit in the above exercise 3.3, add a measurement from the gate library ("?" in the diamond symbol) in the time block after the R-gate. Press compute and you will see the measurement symbol spin and settle on a random outcome "0" or "1" according to the associated probabilities in the quantum state.

Every time you press compute you will get a new measurement outcome. You can move the vertical slider bar to inspect the quantum state before and after measurement.

Question: Do you see how the state effectively collapses after any given measurement?

Question: Press the compute button many times (say N = 100) and record the number of 0 and 1 outcomes and fill in the table below. Compare the estimated probabilities with those expected.

Quantum superposition component	Probability	Measurement record	# outcomes, n	Estimated Prob = n/N
$\|0\rangle$	0.75	𝍸 𝍸 ⋯		
$\|1\rangle$	0.25	𝍸 ⋯		

Exercise 3.5 To construct another example superposition state, right click on the R-gate in the circuit to bring up the rotation gate menu again. Program some random parameters for axis and rotation angle. Press compute repeat the above analyses of the amplitudes and measurement outcome probabilities. Generate other superpositions to make sure you understand the complex polar notation used in QUI, and the physical interpretation of the quantum states produced in terms of measurement outcomes.

Next week we will look at how the qubit operations work, and the Bloch Sphere representation (those little animations that keep popping up).

Matrices and Vectors

Here we will review some common concepts in linear algebra. If you have any trouble, be sure to ask the tutor to help, as getting these concepts clear now will be extremely useful in the coming weeks.

Exercise 4.1 Perform the following matrix multiplications.

$$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} =$$

In the lectures we saw that the qubit states could be expressed as:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Express your answers for Exercise 4.1 as a linear combination of these two vectors.

Exercise 4.2 Write

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

explicitly as a column vector.

Exercise 4.3 Write

$$\frac{\sqrt{3}i}{2}|0\rangle - \frac{1}{2}|1\rangle$$

explicitly as a column vector.

Exercise 4.4 To be correctly normalised, the probabilities of a state have to add to 1. The probabilities are equal to the (modulus of the) amplitude squared. Check that the state given in Exercise 4.3 is correctly normalised.

Also in lectures, we saw how to find the Hermitian adjoint of a matrix. To do this find the matrix transpose, and take the complex conjugate of all elements.

Exercise 4.5 Find the following

$$\begin{bmatrix} 3 & 1+2i \\ 0 & 1 \end{bmatrix}^{\dagger} =$$

$$\begin{bmatrix} 1 & -i \\ i & 2 \end{bmatrix}^{\dagger} =$$

Exercise 4.6 Which of the two matrices in Exercise 4.5 is Hermitian? Why?

Exercise 4.7 A matrix, U, is unitary if $U^{\dagger}U = UU^{\dagger} = I$. Is the matrix Y, given below, unitary?

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Exercise 4.8 Show that the following matrix is unitary,

$$U = e^{i\varphi/2}\begin{bmatrix} e^{i\varphi_1}\cos\theta & e^{i\varphi_2}\sin\theta \\ -e^{-i\varphi_2}\sin\theta & e^{-i\varphi_1}\cos\theta \end{bmatrix}$$

Show that (for appropriate choices of angles) that:

$$U = e^{i\varphi/2}\begin{bmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{bmatrix}\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} e^{i\Delta} & 0 \\ 0 & e^{-i\Delta} \end{bmatrix}$$

MULT90063 Introduction to Quantum Computing

Lab 2

Welcome to Lab 2 of MULT90063 Introduction to Quantum Computing, covering exercises relevant to the material presented in lectures in Week 2.

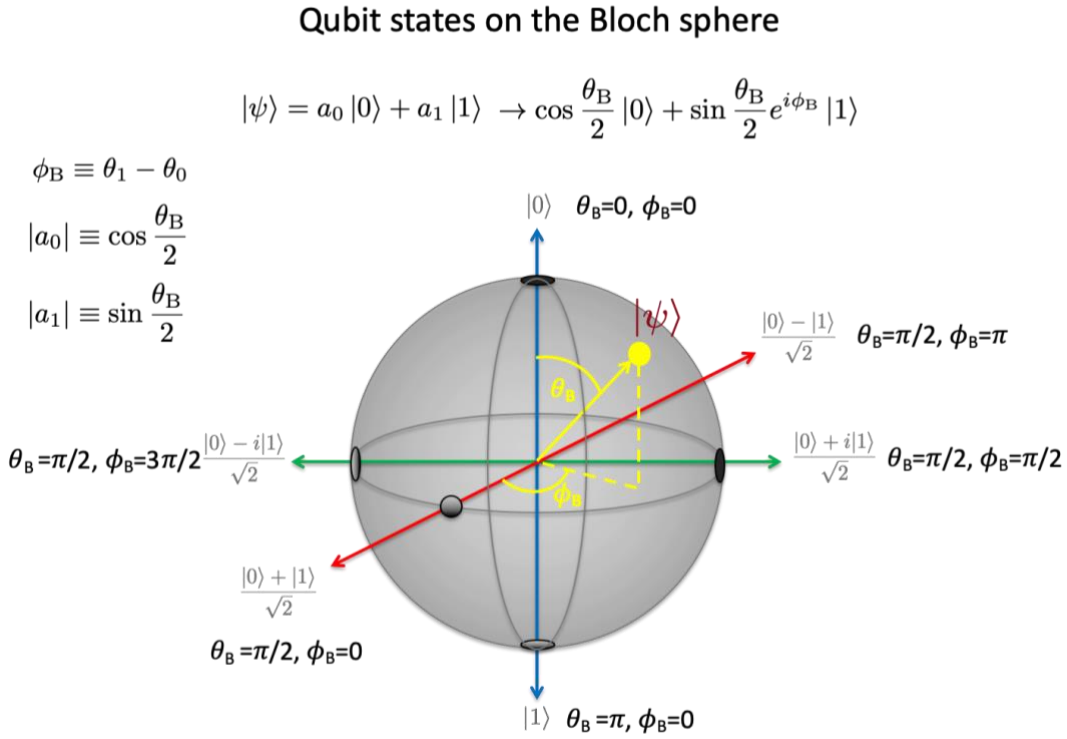
The purpose of this week's exercises is to:

- understand the representation of qubit states on the Bloch Sphere
- understand sequences of logic gates on single qubits
- program sequences of qubit logic gates in QUI
- understand the evolution of qubit states on the Bloch Sphere

The exercises in these notes are to assist your understanding of the subject and may require time outside of the lab to complete.

Qubit states on the Bloch sphere

In lectures we showed how a general qubit state $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$ can be represented as a point on the Bloch sphere. The conversion (ignoring the global phase) is given below:



Exercise 1.1 Consider the state $|\psi\rangle = \frac{1-i}{2}|0\rangle + \frac{1+i}{2}|1\rangle$. We'll go through the whole conversion from this form to the position on the Bloch sphere.

a) Show that the state is appropriately normalised, i.e. that the sum of the probabilities, $|a_0|^2 + |a_1|^2$ is unity.

b) Convert the state to polar form, i.e. $|\psi\rangle = |a_0|e^{i\theta_0}|0\rangle + |a_1|e^{i\theta_1}|1\rangle$.

$|a_0| =$ _____ $\theta_0 =$ _____ $|a_1| =$ _____ $\theta_1 =$ _____

c) Convert the state into "pre-Bloch sphere" form, by pulling out the phase $e^{i\theta_0}$, i.e.

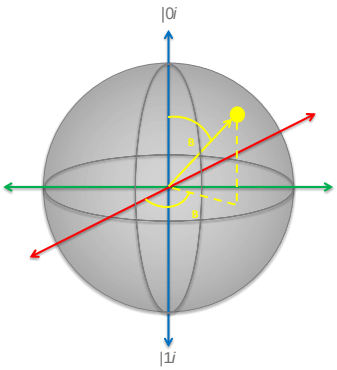
$|\psi\rangle = e^{i\theta_0}\left(|a_0||0\rangle + |a_1|e^{i(\theta_1-\theta_0)}|1\rangle\right)$.

$|\psi\rangle =$

d) From the definition of the Bloch sphere form, $|\psi\rangle = \cos\frac{\theta_B}{2}|0\rangle + \sin\frac{\theta_B}{2}e^{i\phi_B}|1\rangle$, determine the Bloch sphere angles $\phi_B = \theta_1 - \theta_0$ and θ_B, and write out the state in this form:

$\phi_B =$ _____ $\theta_B =$ _____ $|\psi\rangle =$ _____ $|0\rangle +$ _____ $|1\rangle$

e) Given the definitions of the Bloch angles in the schematic (right), plot a point corresponding where this state resides on the Bloch sphere. Compare with the figure on page 1.



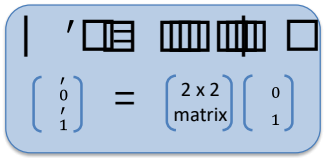
Exercise 1.2 Make up your own (normalised) state, and repeat all the steps a)-e) above.

Logical operations on qubits

In lectures, we introduced the notion of an operator U acting on a quantum state to transform it into a new state, i.e.

$$|\psi'\rangle = U\,|\psi\rangle$$

In quantum computing these operations correspond to logic operations, or gates, on qubits. The matrix representation gives us a useful representation of these logic operations:



Here is some practice with the maths involving the entire QUI logic gate library for single qubits, in both matrix and ket form.

Exercise 2.1 Compute by hand the single gate operations H, X, Y, Z, S, and T on the state $|0\rangle$, and complete the table below. Compare with the QUI in each case.

Gate	Operator (matrix rep)	Operation (matrix rep)	Operation (ket rep)	Final state $\|0\rangle$ amplitude	Final state $\|1\rangle$ amplitude	Final state probabilities p_0 and p_1
H	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix}$ $= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix}$	$H\|0\rangle$ $= \frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle$ $= \|a_0\|e^{i\theta_0}\|0\rangle + \|a_1\|e^{i\theta_1}\|1\rangle$	$\|a_0\| = \frac{1}{\sqrt{2}}$ $\theta_0 = 0$	$\|a_1\| = \frac{1}{\sqrt{2}}$ $\theta_1 = 0$	$p_0 = 0.5$ $p_1 = 0.5$
X	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix}$ $= \begin{pmatrix}0\\ 1\end{pmatrix}$	$X\|0\rangle = \|1\rangle$	$\|a_0\| = 0$ $\theta_0 = 0$	$\|a_1\| = 1$ $\theta_1 = 0$	$p_0 = 0$ $p_1 = 1$
Y	$\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$					
Z	$\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$					
S	$\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$					
T	$\begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}$					

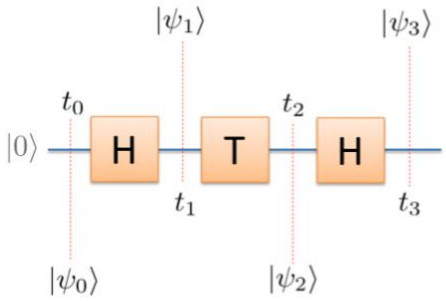
Exercise 2.2 Compute by hand the single gate operations H, X, Y, Z, S, and T on the state $|1\rangle$, and complete the table below. Compare with the QUI in each case.

Gate	Operator (matrix rep)	Operation (matrix rep)	Operation (ket rep)	Final state $\|0\rangle$ amplitude	Final state $\|1\rangle$ amplitude	Final state probabilities p_0 and p_1
H	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix}$ $= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix}$	$H\|1\rangle$ $= \frac{1}{\sqrt{2}}\|0\rangle - \frac{1}{\sqrt{2}}\|1\rangle$ $= \|a_0\|e^{i\theta_0}\|0\rangle + \|a_1\|e^{i\theta_1}\|1\rangle$	$\|a_0\| = \frac{1}{\sqrt{2}}$ $\theta_0 = 0$	$\|a_1\| = \frac{1}{\sqrt{2}}$ $\theta_1 = \pi$	$p_0 = \frac{1}{2}$ $p_1 = \frac{1}{2}$
X	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix}$ $= \begin{pmatrix}1\\ 0\end{pmatrix}$	$X\|1\rangle = \|0\rangle$	$\|a_0\| = 1$ $\theta_0 = 0$	$\|a_1\| = 0$ $\theta_1 = 0$	$p_0 = 1$ $p_1 = 0$
Y	$\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$					
Z	$\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$					
S	$\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$					
T	$\begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}$					

Sequences of logic gates

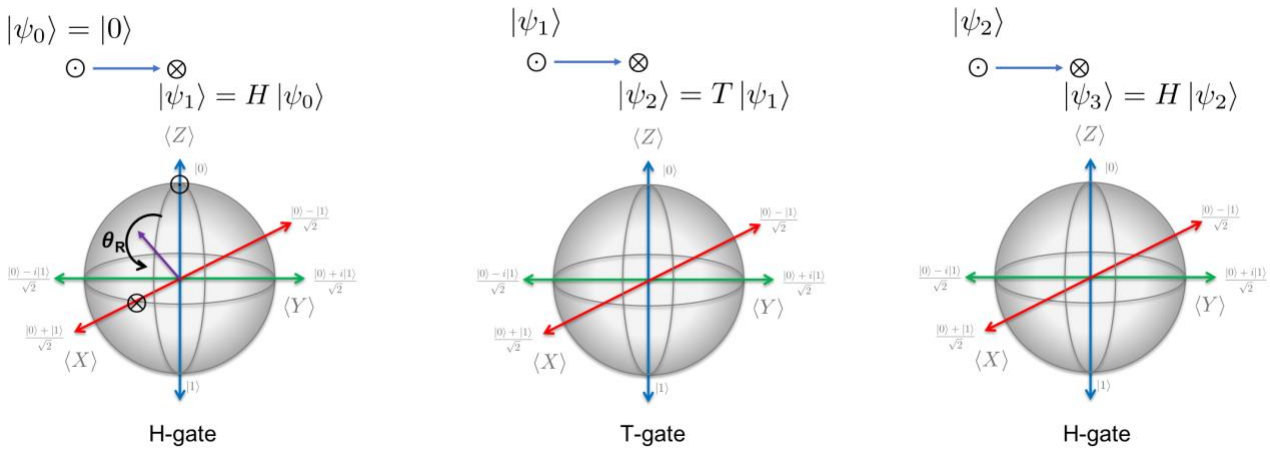
In the following exercises we will look at the mathematics of the quantum state evolution in more detail. In order to understand what is happening we will compute some examples by hand and compare with the QUI output.

Exercise 3.1 Program the following sequence of single qubit gates H-T-H (shown below). Compute by hand the states at each time step in the matrix representation, covert to ket representation and fill out the table below. Now compare the amplitudes you obtained with the QUI output at each time step and check they agree.



Matrix representation	Ket representation	Amplitudes
$\|\psi_0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\|\psi_0\rangle = \|0\rangle$	$\|a_0\| = 1 \quad \|a_1\| = 0$ $\theta_0 = 0 \quad \theta_1 = 0$
$\|\psi_1\rangle = H\,\|\psi_0\rangle$		
$\|\psi_2\rangle = T\,\|\psi_1\rangle$		
$\|\psi_3\rangle = H\,\|\psi_2\rangle$		

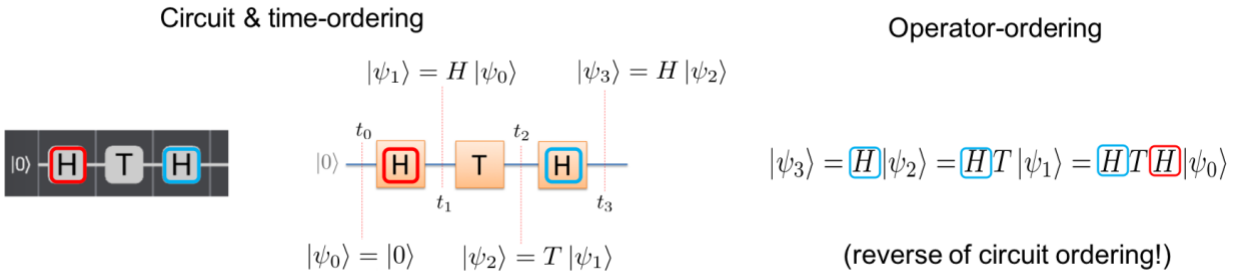
Exercise 3.2 Examine the QUI Bloch sphere animations by hovering the mouse over each gate in the program circuit and complete the following representations of these gates in the sequence H-T-H as per below (i.e. plot ⊙ and ⊗):



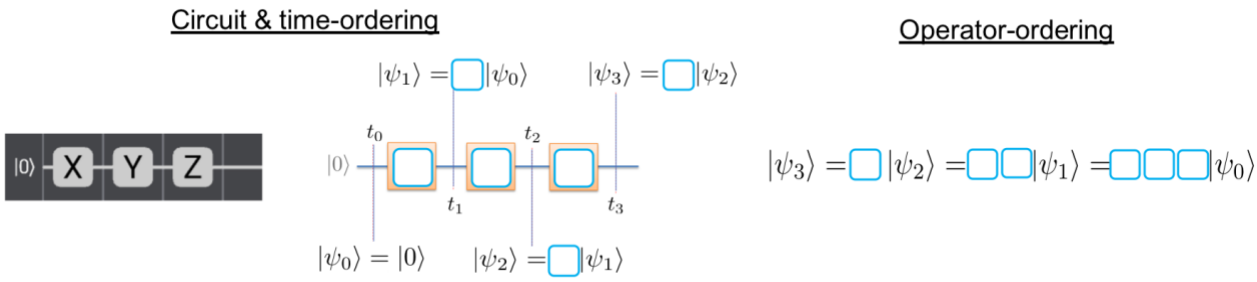
Exercise 3.3 Writing the above example as a string of operations on the initial state would look like the following:

$$|\psi_3\rangle = H\,|\psi_2\rangle = H\,T\,|\psi_1\rangle = H\,T\,H\,|\psi_0\rangle$$

This looks exactly like the circuit ordering, but that's because this example is palindromic (looks the same from either direction) – see below.

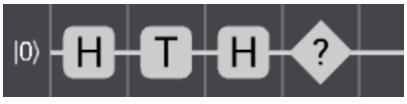


Complete the same analysis for the circuit comprising the combination X-Y-Z:



The reversal of operator and time orderings is something to keep in mind.

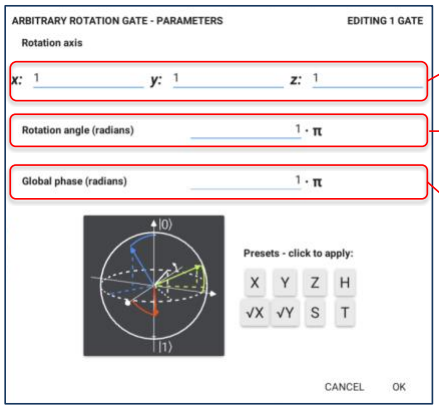
Exercise 3.4 Add a measurement gate at the end of the HTH sequence. Hit the compute button many times (say N = 100) and record the number of 0 and 1 outcomes and fill in the table below. Compare the estimated probabilities with those expected.



$\|\psi\rangle = HTH\|0\rangle$ components	Exact probability	Measurement record	# outcomes, n	Estimated Prob = n/N
$\|0\rangle$	0.854	𝍸 𝍸 ⋯		
$\|1\rangle$	0.146	𝍸 ⋯		

Arbitrary rotation gate, R

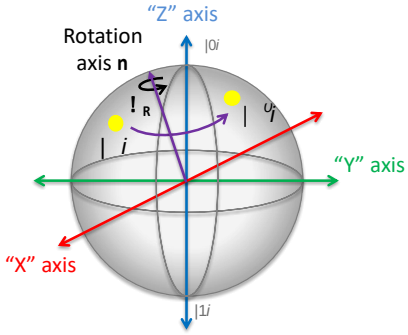
Consider the R-gate in the QUI, with edit menu given below (right clock on the circuit symbol to bring up this menu):



Cartesian cords for axis of rotation **n**

Angle of rotation $_R$ about **n**

Global phase $_g$: generally set to zero unless otherwise directed!



Exercise 4.1 Set QUI to 1-qubit, initialised in the default zero state. Add an R-gate in the first time block. Setting the global phase to zero, explore the action of the R-gate for a range of rotation axes and rotation angles. Hovering the mouse over the R-gate make sure you understand how the qubit state evolves on the Bloch sphere. You can start the system from states different from the default by adding gates prior to the R-gate. If you would like to see the Bloch animations go crazy, try some very high rotation angles.

Exercise 4.2 Going back to the HTH example, can you determine R-gate parameters to place the system, initially in the default zero state, into the same final state?

MULT90063 Introduction to Quantum Computing

Lab Session 3

Introduction

Welcome to Lab 3 of MULT90063 Introduction to Quantum Computing.

In this lab session, you will learn about multi-qubit systems in the QUI, implement several two-qubit quantum gates, and generate entanglement. In the latter half of the lab session, students will implement two simple protocols that exploit quantum entanglement that we covered in the Lectures – teleportation and dense coding.

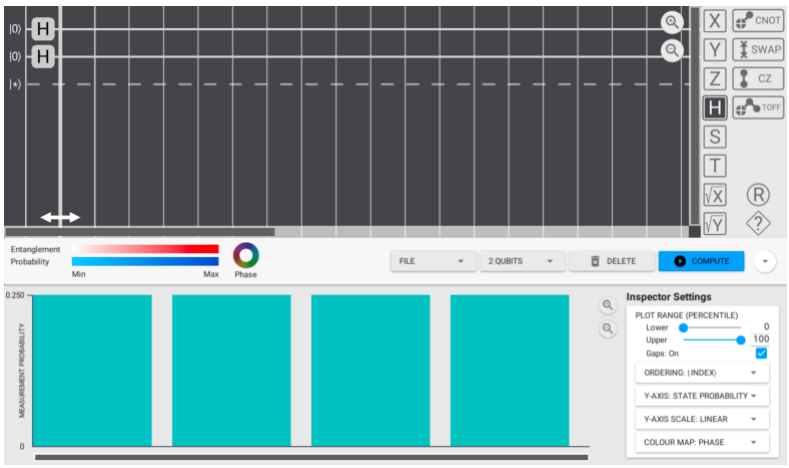
The purpose of this lab session is to:

- implement multi-qubit circuits, and understand the binary state representation, single qubit operations, and measurement on these systems
- implement two-qubit gates on multi-qubit systems
- investigate and understand entanglement generated in simple cases
- implement dense coding and teleportation

Single qubit gates and measurement on multi-qubit Systems

In quantum computing we need to become familiar with the binary representation of information encoded on qubit systems, and the effect of logic operations and measurement on these "binary" states.

Exercise 2.2.1 Set up a two-qubit system in the QUI, place a Hadamard in the first slot for each qubit, and hit compute, i.e:



Mathematically, the circuit has performed the operation:

$$|0\rangle \otimes |0\rangle \rightarrow (H \otimes H)\,|0\rangle \otimes |0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

We will use the more convenient shorthand notation :

$$|00\rangle \rightarrow H_1 H_2 |00\rangle = \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

Based on this two-qubit state answer the following (don't forget to renormalise as required):

What's the probability of measuring any of the four basis states in the final state? _____
What's the probability of measuring $|0\rangle$ in the first qubit? _____
What's the probability of measuring $|0\rangle$ in the second qubit? _____

If you measure $|0\rangle$ in qubit-1, what does the state collapse to? _____

If you measure $|1\rangle$ in qubit-1, what does the state collapse to? _____

If you measure $|0\rangle$ in qubit-2, what does the state collapse to? _____

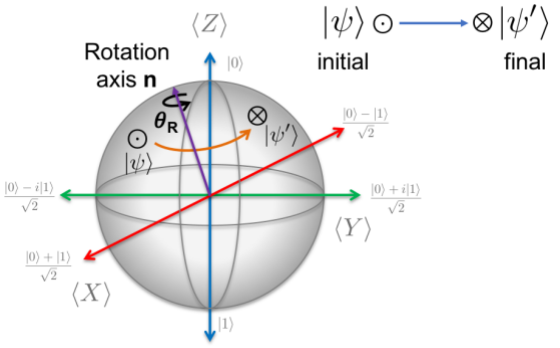
If you measure $|1\rangle$ in qubit-2, what does the state collapse to? _____

Add a measurement gate for qubit-1 after the Hadamard and investigate what happens (hit compute a number of times), consulting the State Info Cards (SICs) for each basis state. Delete the measurement gate on qubit-1 and add a measurement gate to qubit-2, repeat.

Exercise 2.2.2 Now we will repeat, but create a more general two-qubit state. Consider the following operation:

$$|00\rangle \rightarrow |\psi\rangle = R_X(\theta_R)_1 \, H_2 \, |00\rangle$$

Recall the definition of the R-gate in terms of the Pauli operators σ = (*X*, *Y*, *Z*):



$$|\psi'\rangle = R_{\hat{\mathbf{n}}}(\theta_{\mathrm{R}}) \, |\psi\rangle$$

$$R_{\hat{\mathbf{n}}}(\theta_R) = e^{i\theta_g} \left(I \cos\frac{\theta_R}{2} - i \, \hat{\mathbf{n}} \cdot \boldsymbol{\sigma} \sin\frac{\theta_R}{2} \right)$$

$$\boldsymbol{\sigma} = (X, Y, Z) \qquad \hat{\mathbf{n}} = \frac{\mathbf{n}}{|\mathbf{n}|}$$

Construct the $R_X(\theta_R)$ operator in the matrix representation (global phase zero) and convert to ket representation to show that it transforms qubit-1 as:

$$R_X(\theta_R)_1 \, |0\rangle = \cos\frac{\theta_R}{2} \, |0\rangle - i \sin\frac{\theta_R}{2} \, |1\rangle$$

Hence, express the final state of the following independent operations on qubit-1 and qubit-2 in the ket basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$:

$|00\rangle \rightarrow |\psi\rangle = R_X(\theta_R)_1 \, H_2 \, |00\rangle =$ _____

Based on this two-qubit state answer the following (don't forget to renormalise as required):

What's the probability of measuring any of the four basis states in the final state? _____
What's the probability of measuring $|0\rangle$ in the first qubit? _____
What's the probability of measuring $|0\rangle$ in the second qubit? _____

If you measure $|0\rangle$ in qubit-1, what does the state collapse to? _____

If you measure $|1\rangle$ in qubit-1, what does the state collapse to? _____

If you measure $|0\rangle$ in qubit-2, what does the state collapse to? _____

If you measure $|1\rangle$ in qubit-2, what does the state collapse to? _____

Verify the above by examining this state in separable form:

$$|\psi\rangle = \left(\cos\frac{\theta_R}{2}|0\rangle - i\sin\frac{\theta_R}{2}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)$$

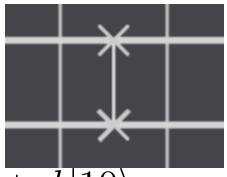
Now program the corresponding circuit in the QUI (R-gate: X-axis, global phase zero) and examine the output state probability distribution and SICs for various choices of θ_R. Perform measurements on qubit-1 and qubit-2 (as per questions above) and make sure you understand the outputs by moving the time scrubber through the circuit (ask a tutor if need be).

Exercise 2.2.3 Generalise Ex 2.2.1 and Ex 2.2.2 to more than 2 qubits, and measurements on subsets of qubits, and make sure you understand the QUI results.

Two-qubit and three-qubit gates

A two-qubit gate acts on two qubits and performs a certain operation either on both qubits, or on one of the qubits conditioned with the state of the other qubit.

CNOT Gate: Target qubit flips when control qubit is in "1" state.



On a general superposition (assuming qubit-1 is the control) we have:

$$\mathrm{CNOT}(a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle) \rightarrow a\,|00\rangle + b\,|01\rangle + c\,|11\rangle + d\,|10\rangle$$

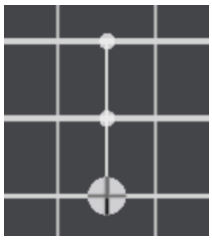
NB. Another way to view CNOT here is that the amplitudes (c, d) of the $|10\rangle$ and $|11\rangle$ states have been switched.

SWAP Gate: States of qubit-1 and qubit-2 are interchanged.

On a general superposition we have:

$$\mathrm{SWAP}(a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle) \rightarrow a\,|00\rangle + b\,|10\rangle + c\,|01\rangle + d\,|11\rangle$$

NB. Amplitudes (b, c) of the $|01\rangle$ and $|10\rangle$ states have been switched. Also, a SWAP gate can be constructed from 3 CNOTs as:



Toffoli Gate: Flip the target if <u>both</u> controls are in the "1" state.

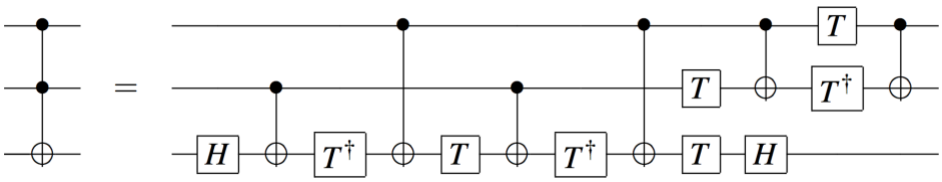
$$\begin{aligned} &a\,|000\rangle + b\,|001\rangle + c\,|010\rangle + d\,|011\rangle \\ &e\,|100\rangle + f\,|101\rangle + g\,|110\rangle + h\,|111\rangle \\ &\rightarrow a\,|000\rangle + b\,|001\rangle + c\,|010\rangle + d\,|011\rangle \\ &\quad e\,|100\rangle + f\,|101\rangle + h\,|110\rangle + g\,|111\rangle \end{aligned}$$

Controlled Rotations: Perform given rotation R on target qubit if control/s is/are in the "1" state.

The QUI provides a default controlled-Z gate in the library, and you can program a general controlled-rotation by adding control(s) to a single qubit R-gate once it is in the circuit (right click).

Exercise 2.3.1 Program each of the above gates in the QUI, using a non-trivial input state (e.g. constructed as per Ex 2.2.2). Examine the outputs carefully and make sure you understand how the states have been transformed.

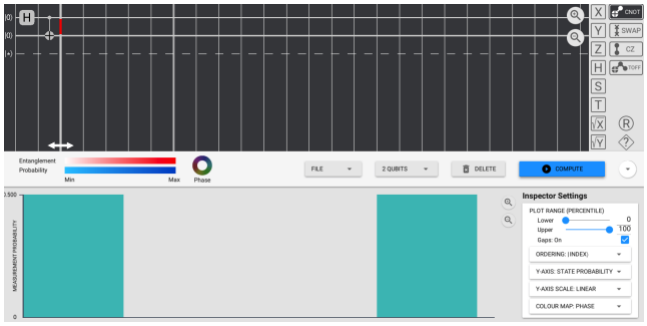
Exercise 2.3.2 As we saw in lectures, the Toffoli gate can be decomposed into CNOTs and single qubit gates (see below).



Note: from T gate definition, show from the expression for $R_{\hat{n}}(\theta_R)$ that to implement T† in terms of R gate you need to reverse angle and global phase settings. Program T† in the QUI and verify it acts as the Toffoli gate (including for superposition states as input). Program the circuit above for the Toffoli gate in the QUI and verify using various inputs that it works correctly.

Entanglement generation

A key aspect of quantum behaviour exploited in a quantum computer is entanglement. In the QUI we indicate the level of entanglement in the overall state at any given time point by plotting the "bi-partite" entanglement (i.e. between qubits above and below) vertically on the time scrubber bar itself (white = no entanglement, red = max entanglement).



Exercise 2.4.1 Program a circuit to create a maximally entangled Bell-State:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

Note that final state has maximal entanglement, as indicated by the scale shown in the Control panel. If you move the time-scrubber back one time step to just after the qubit-1 Hadamard you will see the time-scrubber indicates zero entanglement (since the state is separable at this point).

In general, (for more than 2 qubits) hover over the time-scrubber to bring up the quantitative value at a given qubit bi-partition point.

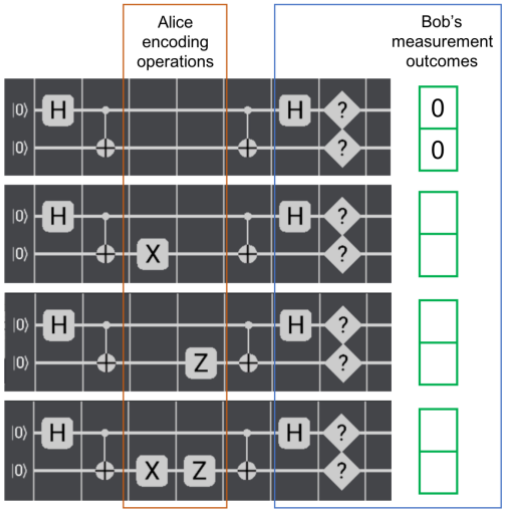
Exercise 2.4.2 Replace the Hadamard on qubit-1 in the above circuit with a $R_X(\theta_R)$ gate and examine the change in entanglement as a function of θ_R and fill in the table.

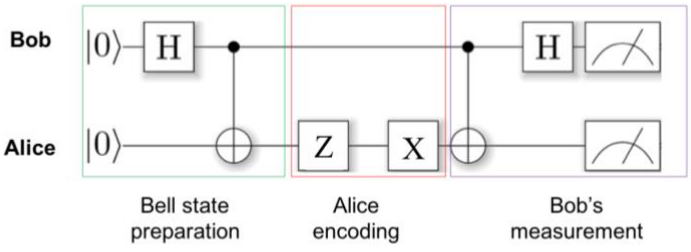
Rotation angle, θ_R (multiples of π)	Prob[|00>]	Prob[|11>]	Entanglement level
0			
1/4			
1/2			
3/4			
1			

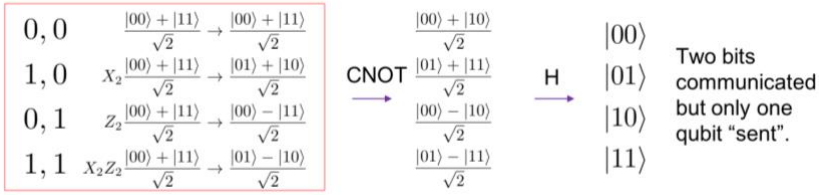
Optional: From the definition of entanglement entropy in the lectures ($S = -\sum_i p_i \log p_i$) can you quantitatively understand the results?

Dense Coding

As we saw in lectures, in classical communication if Alice wants to send two bits of information to Bob she will have to transmit the two bits over the channel. However, by using quantum entanglement, Alice can send one qubit over a quantum channel and transmit two bits of classical information. This is called dense coding – see below.

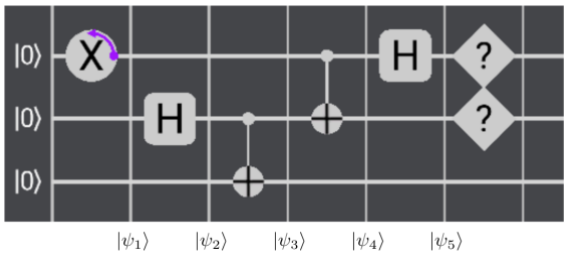






Exercise 2.5.1 Program the basic dense coding circuit in the QUI. Based on Alice's encoding table of operations, code each of the four possible two-bit strings and run the circuit for each case. Move the time-scrubber through the circuit to understand/verify the evolution of the state, and the final state prior to Bob's measurement of both qubits. Fill in the table (above right) verifying the data received by Bob. For this circuit, why do we not really need to have the measurement gates?

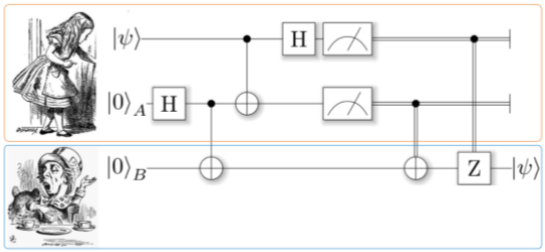
Quantum Teleportation



Alice has a state $|\psi\rangle$ that she wishes to transfer to Bob's qubit. They can't copy because of the no-cloning theorem, so they use the spooky properties of quantum entanglement. Initially they share two qubits (A = Alice, B = Bob) that are entangled, and then Alice makes measurements on her two qubits. Those measurements cause the original state $|\psi\rangle$ to be instantaneously imprinted on Bob's qubit. Incredible, but true (verified in labs).

You might wonder: now that we have a SWAP gate why doesn't Alice just use that? Yes, she could, and that is how we would normally do it in a quantum computer when everything is proximal, but imagine Alice and Bob physically separate after setting up the initial shared entanglement…the teleport procedure will still work no matter where they are!

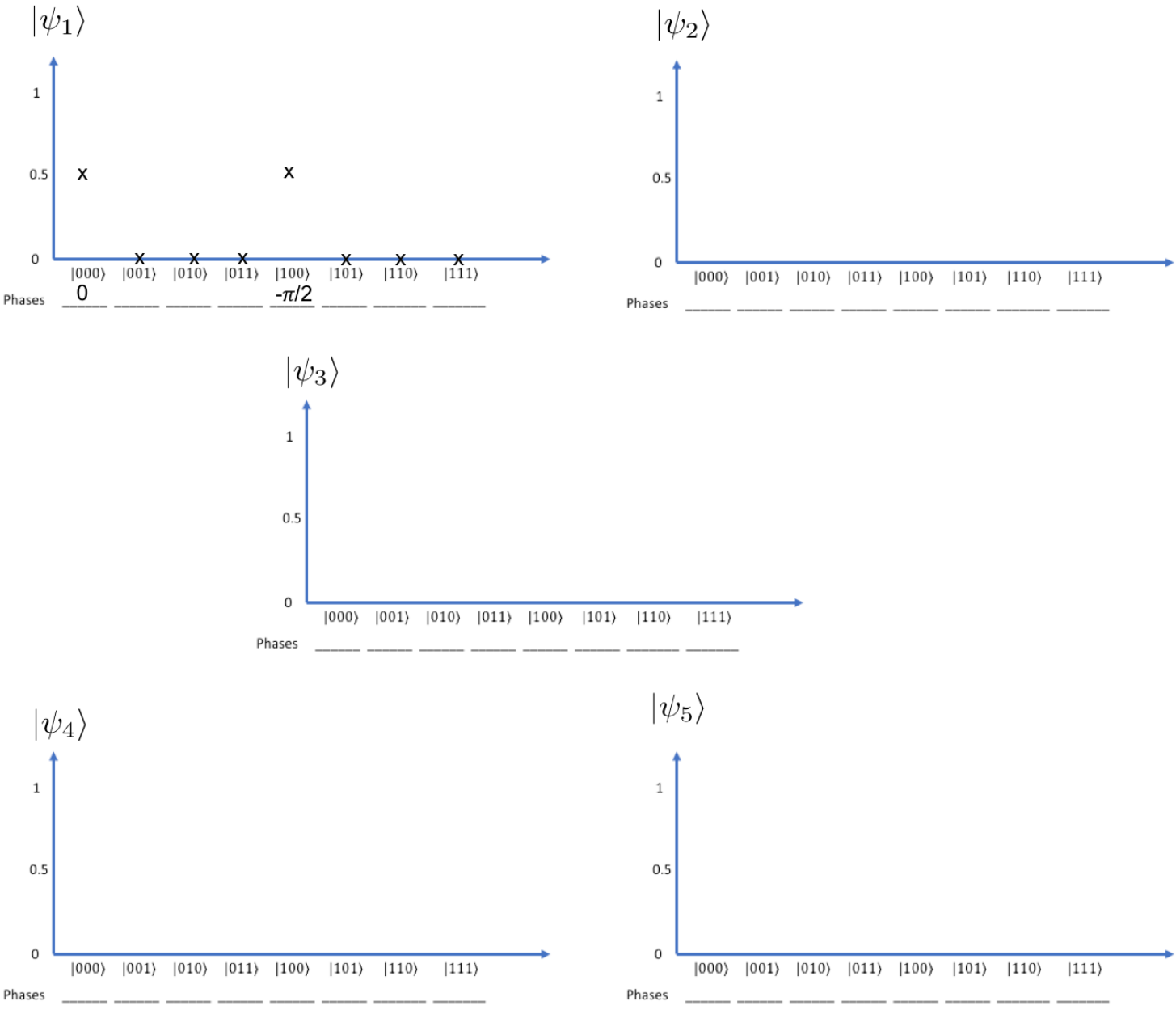
In lectures, we saw how teleportation is implemented through the following circuit:



Alice measures	Bob receives	Alice directs Bob to apply to his qubit	Bob corrects to successfully reconstruct Alice's original state.
$0,0$	$\alpha\|0\rangle + \beta\|1\rangle$	No operation	$\alpha\|0\rangle + \beta\|1\rangle \rightarrow \alpha\|0\rangle + \beta\|1\rangle$
$0,1$	$\alpha\|1\rangle + \beta\|0\rangle$	X	$X(\alpha\|1\rangle + \beta\|0\rangle) \rightarrow \alpha\|0\rangle + \beta\|1\rangle$
$1,0$	$\alpha\|0\rangle - \beta\|1\rangle$	Z	$Z(\alpha\|0\rangle - \beta\|1\rangle) \rightarrow \alpha\|0\rangle + \beta\|1\rangle$
$1,1$	$\alpha\|1\rangle - \beta\|0\rangle$	ZX	$ZX(\alpha\|1\rangle - \beta\|0\rangle) \rightarrow \alpha\|0\rangle + \beta\|1\rangle$

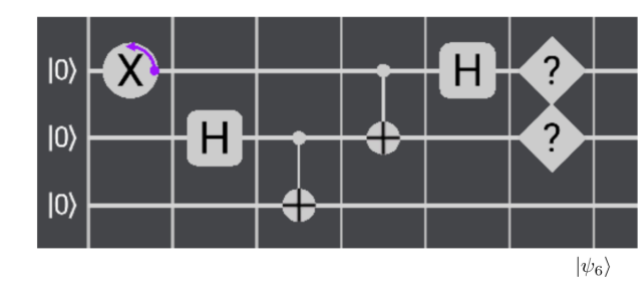
Exercise 2.6.1 In the QUI, set the teleportation circuit (up to and including the measurement gates) using a $R_X(\theta_R)$ gate to create the state $|\psi\rangle$ that Alice will teleport to Bob. Save circuit as "Lab-2 Teleport" (example below has $\theta_R = \pi/2$), and global phase zero)

Before you run the QUI simulation, go through the circuit by hand with $\theta_R = \pi/2$ and fill in the following (vertical axes = basis state probability):



Exercise 2.6.2 Now run the circuit "Lab-2 Teleport" and for each time step (1-5) compare the results to those above.

Exercise 2.6.3 Run the circuit "Lab-2 Teleport" enough times to result in all four of the measurement outcomes on Alice's qubits and note the state of the system at time step 6 (as per below):



From the QUI output fill in the table below for the state of the 3rd qubit (Bob's qubit) in each scenario.

Alice measures	System state $\|\psi_6\rangle$	Bob's state (t = 6)	Correction to obtain Alice's state
0 , 0	$\frac{1}{\sqrt{2}}\|000\rangle + \frac{-i}{\sqrt{2}}\|001\rangle$	$\frac{1}{\sqrt{2}}\|0\rangle + \frac{-i}{\sqrt{2}}\|1\rangle$	No correction
0 , 1			
1 , 0			
1 , 1			

Exercise 2.6.4 Exploration. Run the teleportation circuit a number of times for different values of θ_R. For each instance examine the output SICs and see if you can confirm by inspection the teleport of Alice's state as a result of the required correction.

MULT90063 Introduction to Quantum Computing

Lab Session 4

Welcome to Lab 4 of MULT90063 Introduction to Quantum Computing.

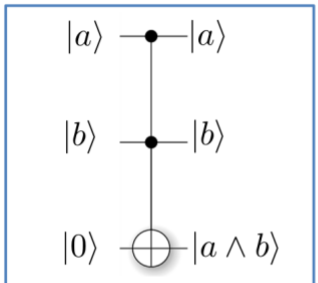
The purpose of this first lab session is to:

- implement classical logic using Toffoli gates
- program arithmetical operations (addition)
- program and analyse simple quantum algorithms

4.1 Classical logic using the Toffoli gate

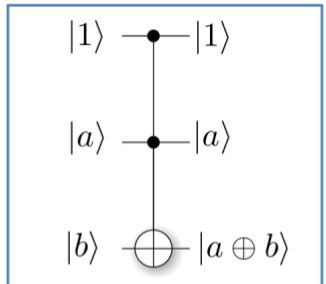
Here we will verify, by direct calculation, how the 3-qubit Toffoli gate allows us to perform classical logic. In the figure below examples are given together with the truth tables.

a AND b



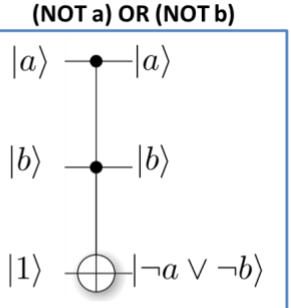
a	b	a ∧ b
0	0	0
0	1	0
1	0	0
1	1	1

a XOR b



a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

(NOT a) OR (NOT b)

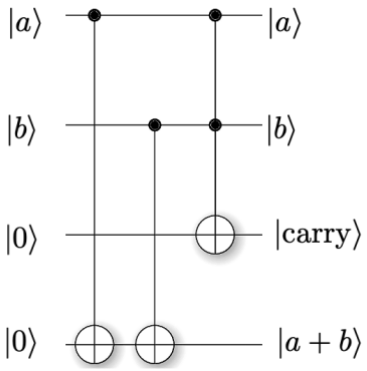


a	b	¬a ∨ ¬b
0	0	1
0	1	1
1	0	1
1	1	0

Exercise 4.1.1 In the QUI set up a 3 qubit instance and program a Toffoli gate, as above with qubit-1 and qubit-3 as the controls and qubit-3 as the target. By setting the input states appropriately verify each entry in the truth tables.

4.2 One-bit adder circuit

Exercise 4.2.1 In the lectures we went through the one-bit adder circuit in detail.



a) With reference to the one-bit adder circuit, review and understand the table below.

Input		Output			
a	b	a+b	carry	Output string	Output string: decimal
0	0	0	0	[0 0]	0
0	1	1	0	[0 1]	1
1	0	1	0	[0 1]	1
1	1	0	1	[1 0]	2

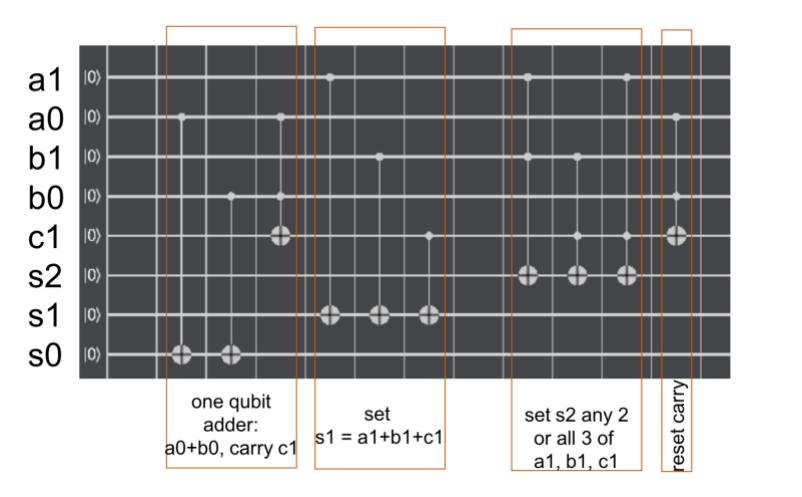
b) Program the circuit in the QUI, leaving space at the start of the circuit to add gates to specify the input states. For classical bit inputs, run the circuit and complete the following table:

Input		Output		
|a>	|b>	|a>|b>|carry>|a+b>	Answer (binary)	Answer (decimal)
|0>	|0>	|0>|0>|0>|0>	[0 0]	0

c) Explore the adder circuit with various superpositions as inputs, and understand what you are seeing.

4.3 Two-bit adder circuit

Consider the two-bit adder circuit below for S = A + B, where A = a0 + 2a1, B = b0 + + 2b1 and S = s0 + 2s1.

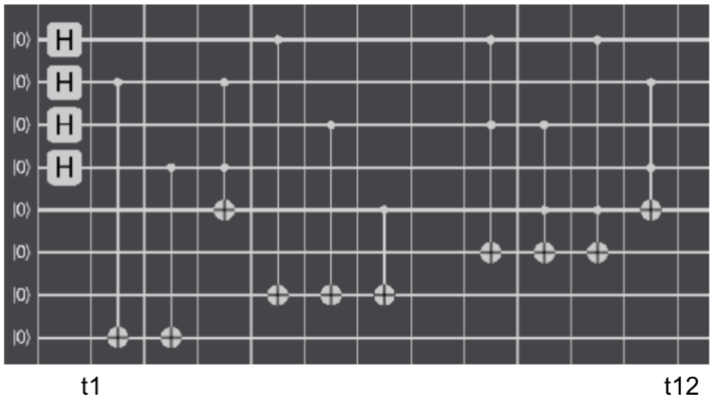


Exercise 4.3.1

a) Program the two-bit adder in the QUI. Run the circuit (with measurements at the end) and verify the arithmetic by filling in the following table (least significant bit last).

[a1 a0]	[b1 b0]	[s2 s1 s0]	S = A + B (decimal)
[0 0]	[0 0]	[0 0 0]	0 = 0 + 0
[0 1]	[0 0]		
[1 0]	[0 0]		
[1 1]	[0 0]		
[0 0]	[0 1]		
[0 1]	[0 1]		
[1 0]	[0 1]		
[1 1]	[0 1]		
[0 0]	[1 0]		
[0 1]	[1 0]		
[1 0]	[1 0]		
[1 1]	[1 0]		
[0 0]	[1 1]		
[0 1]	[1 1]		
[1 0]	[1 1]		
[1 1]	[1 1]		

b) Now edit the circuit so that all possible binaries A and B are presented to the adder in a superposition:

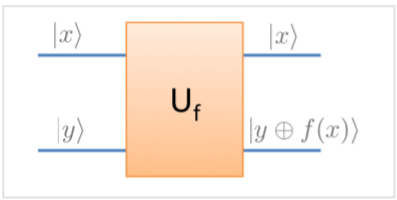


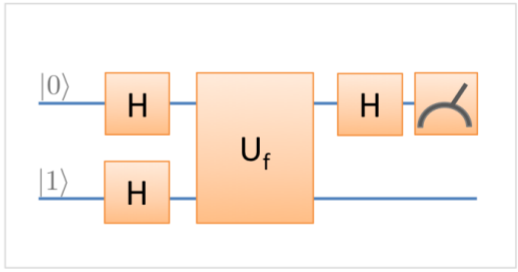
Explain the difference between the input state at t1 and the output state at t12 (also, note the level of entanglement generated in this circuit).

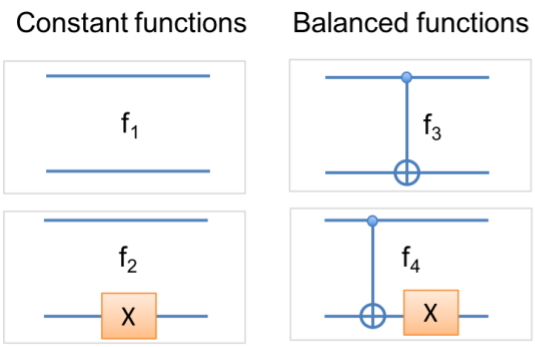
c) Add SWAP gates to the input and output parts of the circuit so that the bit ordering of A, B and S are reversed (i.e. most significant bit last).

4.4 Deutsch-Josza Circuits (1-qubit)

As we saw in lectures, the Deutsch-Josza (DJ) algorithm tests whether a binary function is constant or balanced – as per the summary below (for the one-bit functions).





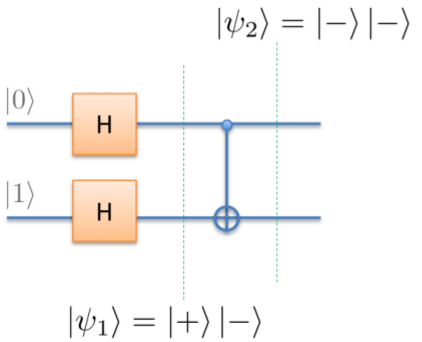


		Measured
CONSTANT	$f_1(0) = 0$ $f_1(1) = 0$	0
	$f_2(0) = 1$ $f_2(1) = 1$	0
BALANCED	$f_3(0) = 0$ $f_3(1) = 1$	1
	$f_4(0) = 1$ $f_4(1) = 0$	1

Exercise 4.4.1

a) Program the Deutsch-Josza circuit in the QUI as per above, and verify the algorithm outputs for the four function types.

b) Examine phase kickback in the QUI – program the following circuit (make sure you understand the X-basis states shown) and fill in the QUI states as indicated (ket/binary representation). Can you see the phase kickback in the QUI representation of the states?



QUI states

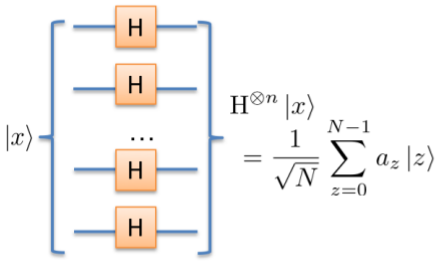
$|\psi_1\rangle$ _____

$|\psi_2\rangle$ _____

4.5 Deutsch-Josza Circuits (n-qubit)

Exercise 4.5.1 Verify the expression for the state after application of multiple Hadamards to a general state in the x-register:

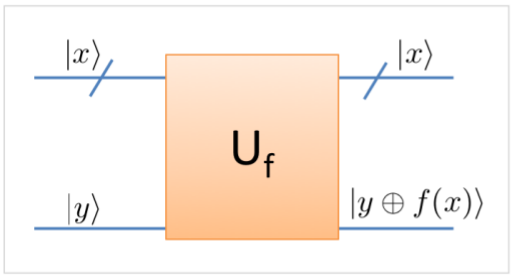
Hadamards applied to a general state (n qubits, N = 2^n): $\mathrm{H}^{\otimes n}|x\rangle = \frac{1}{\sqrt{N}}\sum_{z=0}^{N-1}(-1)^{x\cdot z}|z\rangle$

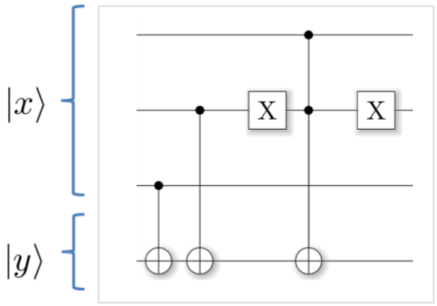


When 1's in the same location, we get a sign change: $(-1)^{x\cdot z}$

$$x \cdot z = \sum_{j=0}^{n} x_j z_j$$

Exercise 4.5.2 Consider the balanced function example given in lectures (see below).

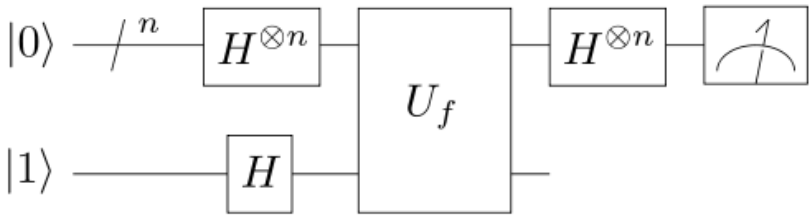




x	f(x)
000	0
001	1
010	1
011	0
100	1
101	0
110	1
111	0

Program this in the QUI and verify the table given using $|y\rangle = |0\rangle$. Now set $|y\rangle = |1\rangle$. Do you understand what is happening? Put the x-register into an equal superposition over all 8 binaries and run the circuit again. What is the circuit now producing as output? Place a measurement on the y-register after the function and explain what happens.

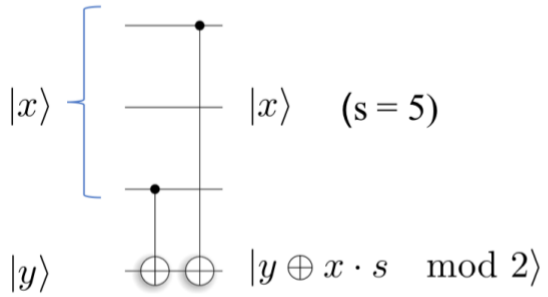
Exercise 4.5.3 Program this function into a circuit performing the Deutsch-Josza algorithm (as shown), run it and verify that the correct conclusion about the function is obtained (i.e. measurements on x-register imply a balanced function). Can you verify in the QUI that the conclusion is deterministic, i.e. that the circuit will never produce an output corresponding to a wrong conclusion?



4.6 Bernstein-Vazirani Algorithm

In the Bernstein-Vazirani problem we have a function $f = x \cdot s \bmod 2$ which maps the product of two n-bit numbers to {0,1}.

Exercise 4.6.1 Program the following (oracle) function in the QUI for the case s = 5 and verify the table of results.

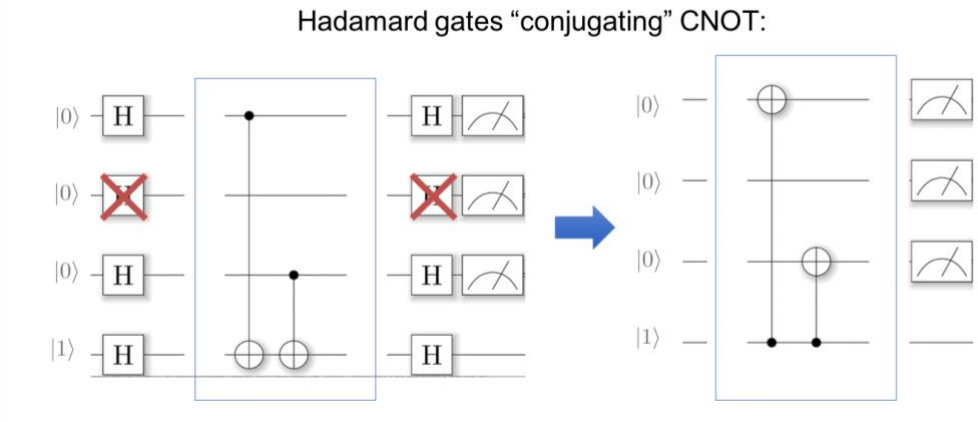


x	f(x)
000	0
001	1
010	0
011	1
100	1
101	0
110	1
111	0

Exercise 4.5.2 Modify the circuit to compute $f = x \cdot s \bmod 2$ for s = 3 and complete the table of results:

x	f(x)	x	f(x)
000		100	
001		101	
101		110	
011		111	

Exercise 4.6.3 Program the following circuits and verify that the equivalence holds. Superposition?

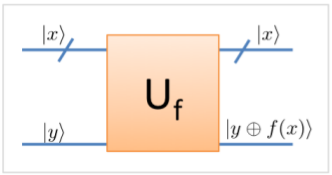


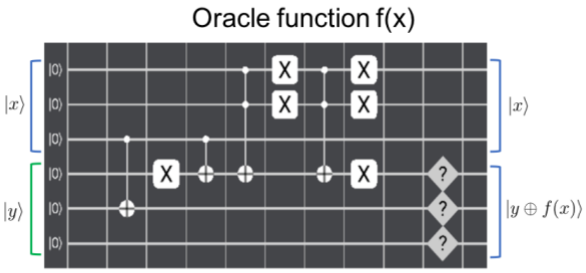
Exercise 4.6.4 Program the oracle function $f = x \cdot s \bmod 2$ which into the BV circuit to solve for the parameter s (i.e. using the DJ circuit) and verify the solution for a range of values of s (changing the oracle function as required).

4.7 Simon's Algorithm

Here we will implement a 3-bit example of finding the period of a function, i.e. a: $f(x) = f(x \oplus a)$, using Simon's algorithm.

Exercise 4.7.1 Consider the following circuit which implements an oracle function $f(x)$ over 3-bit numbers, with a period a.

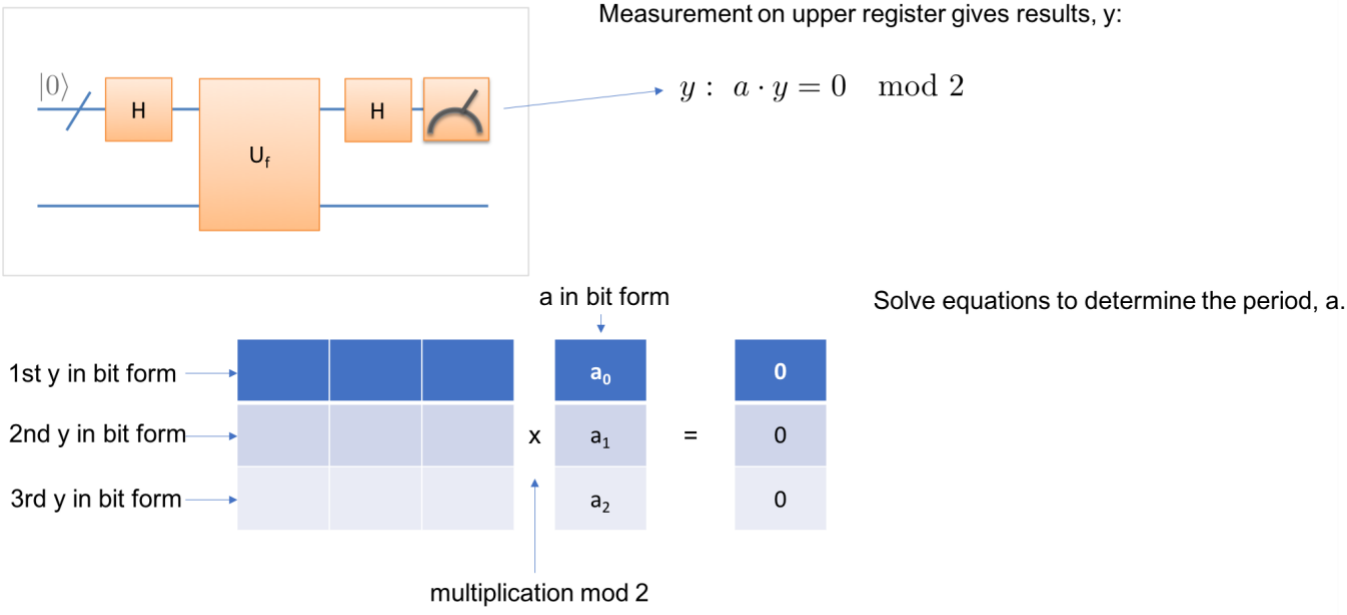




x	f(x)
000	
001	
010	
011	
100	
101	
110	
111	

a) Program and run the circuit and fill in the table. By (classical) inspection, what is the period of the function we wish to determine (both bit and decimal forms)?

b) Incorporate the oracle into a circuit in the QUI to determine the period using Simon's algorithm. Run the circuit several times to obtain three distinct integers (y_a, y_b, y_c) in the upper register. From the system of equations (as per schematic below) and solve by hand (a classical step) to obtain the period a in bit form, and hence determine the decimal (integer) form of a and compare with the classical solution above.



Run the time scrubber through the circuit and see if you can understand how it's working.

MULT90063 Introduction to Quantum Computing

Lab Session 5

Introduction

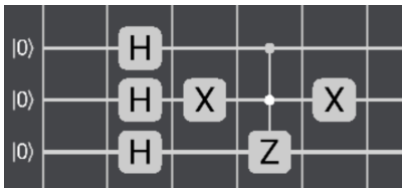
Welcome to Lab 5 of MULT90063 Introduction to Quantum Computing.

The purpose of this lab session is to:

- understand the underpinning concepts of quantum search
- implement oracle functions, inversion and inversion-about-the-mean
- implement Grover's algorithm for single and multiple solution cases
- implement amplitude amplification for single and multiple solution cases

Marking states with an oracle

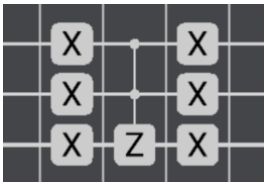
Exercise 5.2.1 a) Consider the oracle function below. Verify that it marks one state in the equal superposition. What is the corresponding binary number marked?



b) Build and run a circuit to implement an oracle function that will mark the number 13 (Most Significant Bit (MSB) at the top, i.e. 13 = 01101).

Inversion

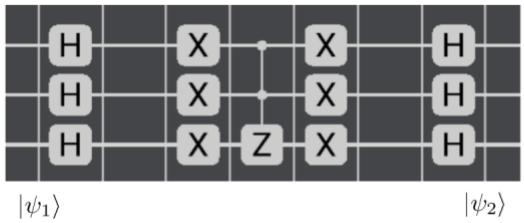
Exercise 5.3.1 a) Show that the following (3-bit) circuit implements the "inversion" operation: $I - 2|000\rangle\langle 000|$.



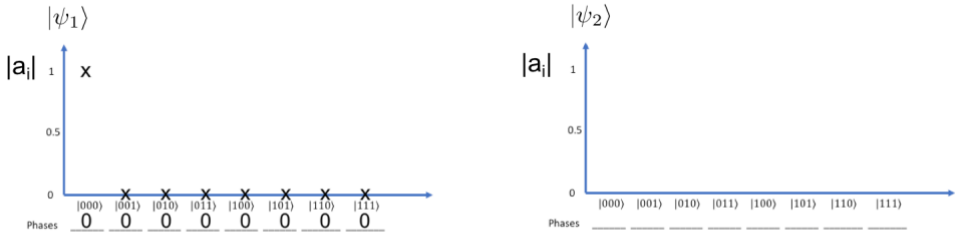
b) Generalise to the 5 qubit case and test.

Inversion about the mean

Exercise 5.4.1 Show that the following circuit implements the "inversion about the mean" operation for 3 qubits and fill in the state amplitudes and phases in the plots provided.
Check that the output state has the amplitudes reflected about the average.



$|\psi_1\rangle$ $|\psi_2\rangle$



$$A = \frac{1}{N}\sum_i a_i = 0.125 \qquad \text{Check}: a_i \to (a_i - 2A)$$

Grover search - single solution case

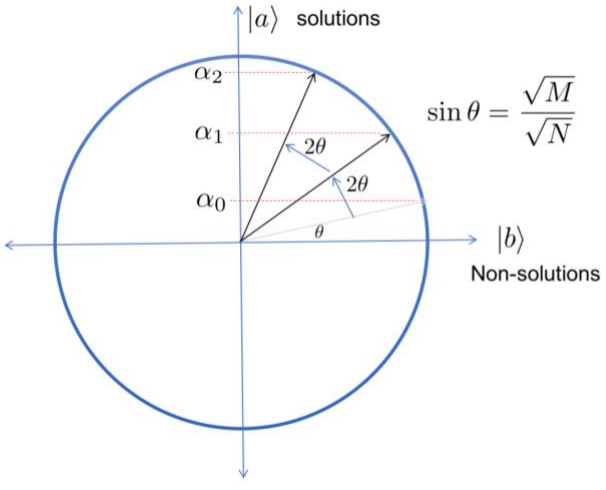
Now we'll put all that together to implement simple instances of Grover's search algorithm.

Exercise 5.5.1 Program a circuit in the QUI that implements Grover's search (at least 8 iterations) for the number 5 over a data base of 3-bit numbers represented in an equal superposition. Save the circuit as "Lab5 5.5.1 Grover 3-qubits oracle 5". Run the scrubber through the circuit to see what's happening with the states and entanglement.

From lectures we saw the geometric picture of Grover's algorithm as a series of rotations towards the search state(s).

$$|\psi_0\rangle = \alpha_0 |a\rangle + \beta_0 |b\rangle$$
$$|\psi_1\rangle = \alpha_1 |a\rangle + \beta_1 |b\rangle$$
$$|\psi_2\rangle = \alpha_2 |a\rangle + \beta_2 |b\rangle$$
$$\vdots$$
$$|\psi_i\rangle = \alpha_i |a\rangle + \beta_i |b\rangle$$

$$\alpha_j = \sin((2j+1)\theta)$$



Exercise 5.5.2 For the example in 5.5.1 (one solution, i.e. M=1) compute the Grover angle θ and fill in the table below to verify the geometric picture c.f. QUI outputs (α_j is the amplitude of the search state |101>).

Iteration, j	Inspect in QUI		Geometric Picture		
	α_j	$\|\alpha_j\|^2$	$(2j+1)\theta$ (rad)	$\alpha_j = \sin((2j+1)\theta)$	$\|\alpha_j\|^2 = [\sin((2j+1)\theta)]^2$
0	0.354	0.125	0.3614	0.3536	0.1250
1	0.884	0.781	1.0842	0.8839	0.7813
2					
3					
4					
5					
6					
7					
8					

Exercise 5.5.3 a) Program a circuit in the QUI that implements one iteration of Grover's algorithm searching for the number 12 (01100) over an integer database loaded into a 5 qubit register in equal superposition. Save as "Lab5 5.5.3 Grover 5-qubits oracle 12".

b) Add iterations and fill in the table below and determine many (minimal) iterations are optimal. Hint: use QUI's cut/paste feature, have spaces between Grover iterations to separate and identify, build and run as you go to record the data.

Iteration	Compute in QUI		Geometric Picture		
	α_j	$\|\alpha_j\|^2$	$(2j+1)\theta$ (rad)	$\alpha_j = \sin((2j+1)\theta)$	$\|\alpha_j\|^2 = [\sin((2j+1)\theta)]^2$
0	0.177	0.031	0.1777	0.1768	0.0313
1	0.508	0.258	0.5331	0.5082	0.2583
2					
3					
4					
5					
6					
7					
8					

Based on this data determine the optimal query number and compare with that derived in lectures for the one-solution (M=1) case.

Grover search - multiple solution case

Generalise to multiple solutions.

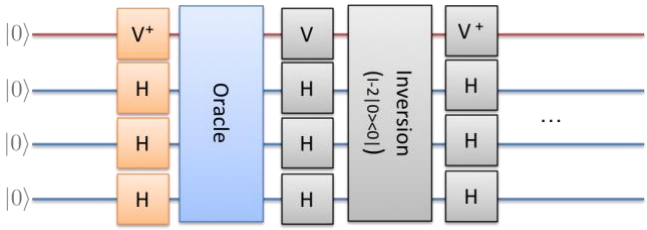
Exercise 5.6.1 Load the circuit "Lab5 5.5.3 Grover 5-qubits oracle 12" (i.e. one iteration). Now modify the oracle (grab multiple gates, copy, move, paste etc) to also mark the state corresponding to the number 3 (00011). Move the scrubber to the output of the oracle and verify the state marking. Run the circuit and fill in the table below.

Iteration, j	Compute in QUI	Geometric Picture		
	Probability measure |3> or |12>	$(2j+1)\theta$ (rad)	$\alpha_j = \sin((2j+1)\theta)$	$\|\alpha_j\|^2 = [\sin((2j+1)\theta)]^2$
0	2x0.031=0.062	0.2527	0.2500	0.0625
1	2x0.236=0.472	0.7581	0.6875	0.4727
2				
3				
4				
5				
6				
7				
8				

Based on this data determine the optimal query number and compare with that derived in lectures for the two-solution (M=2) case.

Amplitude amplification (AA)

We will now use amplitude amplification to modify the Grover search to ensure 100% probability (to within rounding errors) can be achieved after a finite number of iterations. First step is to code the basic circuit given below:



Exercise 5.7.1 Go back to the 3-qubit case (5.5.1) and load "Lab5 5.5.1 Grover 3-qubits oracle 5". Delete all but the first iteration (right click, delete all to the right). Set to 4 qubits and move all gates to qubits 2-4 to free up qubit 1 for the V operator. Using V = $R_y(\theta_R)$ set by the R-gate program the new inversion step for amplitude amplification including the top-most qubit.

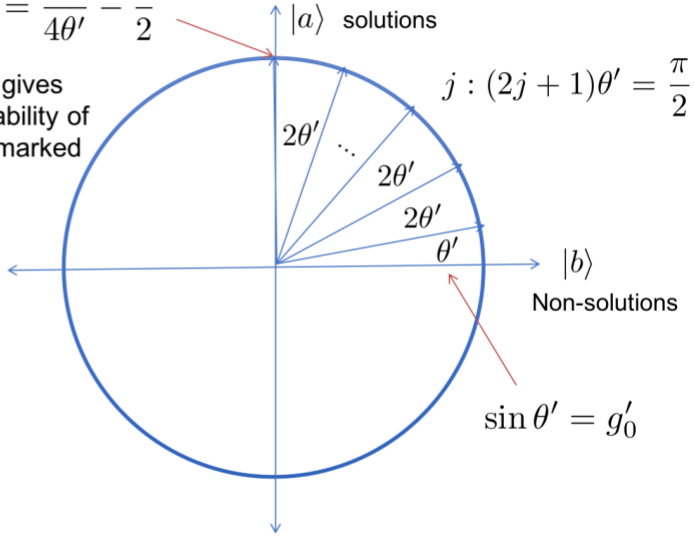
As covered in lectures, to effect amplitude amplification with a general geometric angle we introduce an extra operation (and qubit) in the process. In the QUI we will use the R-gate with a suitably chosen angle, θ_R. But which θ_R to use? We can easily calculate it – see the summary below for the maths.

Summary of Amplitude Amplification

Grover angle: $\sin\theta = \frac{\sqrt{M}}{\sqrt{N}} = g_0$

$$j = \frac{\pi}{4\theta'} - \frac{1}{2}$$

The j[th] step gives 100% probability of finding the marked state



Amplitude Amplification operator, V:

$$V|0\rangle = \sqrt{1 - \left(\frac{g_0'}{g_0}\right)^2}\,|0\rangle + \frac{g_0'}{g_0}|1\rangle$$

$$V = R_y(\theta_R)|0\rangle = \cos\frac{\theta_R}{2}|0\rangle + \sin\frac{\theta_R}{2}|1\rangle$$

$$\rightarrow \frac{g_0'}{g_0} = \sin\frac{\theta_R}{2}$$

Amplitude Amplification angle: $\sin\theta' = g_0'$

Exercise 5.7.2 Now we will determine the y-rotation angle θ_R in order to produce the search state result with 100% probability (within rounding errors). Have a look through the maths above. Use the following table as a guide to the determination of an integer value of the number of iterations, *j*, as a function of the y-rotation angle θ_R (for the case specified). Find the θ_R that gives the least number of iterations to reach 100% search result, and fill in values in the following table.

n	N	M	$g_0 = \sqrt{\frac{M}{N}}$	$j = \frac{\pi}{4\theta'} - \frac{1}{2}$	$\theta' = \sin^{-1} g_0'$	g_0'	θ_R
3	8	1	0.3536				

Exercise 5.7.3 Program the determined y-rotation angle into the R-gates in the QUI, run the program, and verify it's working as expected. Save the circuit as "Lab5 5.7.3 AA V=Ry 3-qubits oracle 5".

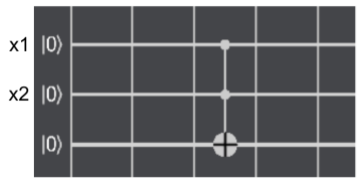
Exercise 5.7.4 Modify the circuit to a search for two numbers (M=2) over a 7-bit register again with V set to a rotation about the y-axis. Determine the required rotation angle in V to produce a minimal number of iterations to reach the ideal 100% probability of finding one (unspecified) of the target numbers. Use the table below as a guide.

n	N	M	$g_0 = \sqrt{\frac{M}{N}}$	$j = \frac{\pi}{4\theta'} - \frac{1}{2}$	$\theta' = \sin^{-1} g_0'$	g_0'	θ_R
7	128	2	0.1250	8.3741	0.2527	0.0883	π/2

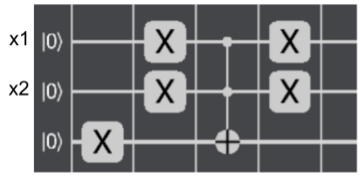
Exercise 5.7.5 Exploration: try a higher number of solutions, and/or other operators for V, and/or different oracles/search problems.

More adventurous oracles

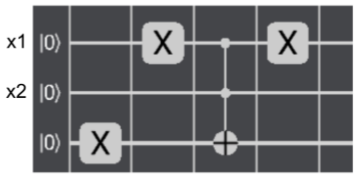
Let's now consider the more realistic case where the oracle is programmed to evaluate a function. We'll consider a function based on Boolean logic statements (i.e. akin to a SAT problem). To that end, consider the following examples of implementing simple two-variable Boolean clauses using quantum logic (generated using the tool at web.stanford.edu/class/cs103/tools/truth-table-tool/).



x1	x2	(x1 ∧ x2)
F	F	F
F	T	F
T	F	F
T	T	T



x1	x2	(x1 ∨ x2)
F	F	F
F	T	T
T	F	T
T	T	T



x1	x2	(x1 ∨ ¬x2)
F	F	T
F	T	F
T	F	T
T	T	T

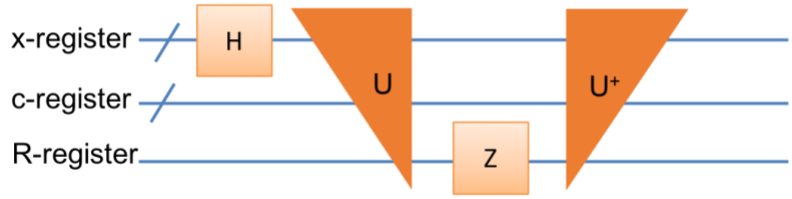
Exercise 5.8.1 Program these into the QUI and verify the truth tables (you can do all instances in one go using Hadamards to generate an equal superposition of inputs).

Exercise 5.8.2 Construct a circuit that evaluates the sequence of clauses below over a superposition of all inputs x1, x2, x3.

x1	x2	x3	(¬(x1 ∧ x2) ∧ ((x2 ∨ ¬x3) ∧ (¬x1 ∨ x3)))
F	F	F	T
F	F	T	F
F	T	F	T
F	T	T	T
T	F	F	F
T	F	T	F
T	T	F	F
T	T	T	F

Hint: Use 3 qubits for the (x1, x2, 3) register ("x-register"), one qubit for each of the outputs of the 3 clauses ("c-register"), and finally one qubit for the result of the overall expression ("R-register").

Exercise 5.8.3 If the circuit that evaluates the logical expression in 5.8.2 is defined as U, construct the corresponding oracle as per the schematic below and verify the solutions are marked. Note: the operations in U are all self-adjoint, so to construct U^+ simply reverse the order of the gates in U.



Save the circuit as “Lab5 5.8.3 SAT oracle”.

Exercise 5.8.4 Now add inversion about the mean on the x-register to complete one iteration of the Grover search. Save as “Lab5 5.8.3 SAT Grover iteration”. Run the circuit and increase the number of Grover iterations until the solution is found.

MULT90063 Introduction to Quantum Computing

Lab Session 6

6.1 Introduction

Welcome to Lab 6 of MULT90063 Introduction to Quantum Computing.

The purpose of this lab session is to:

- understand the underpinning concepts of the quantum Fourier transform (QFT)
- implement QFT circuits
- implement basic addition circuits based on QFT
- implement a simple version of Shor's quantum factoring algorithm

6.2 The quantum Fourier transform (QFT)

In lectures we went through the Quantum Fourier Transform in some detail – main points are summarised below:

Quantum Fourier Transform (QFT) - summary

Classical: $(x_0, x_1, ..., x_{N-1}) \in \mathcal{C}^N$ to $(y_0, y_1, ..., y_{N-1}) \in \mathcal{C}^N$ $\quad y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i jk/N}$

Quantum:

NB. i = sqrt(-1), j and k are integers

a) General state: $|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle \overset{\text{QFT}}{\rightarrow} |\psi'\rangle = \sum_{j=0}^{N-1} y_j |j\rangle$

with $y_k = \sum_{j=0}^{N-1} F_{kj} x_j \qquad F_{kj} = \frac{1}{\sqrt{N}} e^{2\pi i jk/N}$

b) On an individual basis state $|a\rangle$

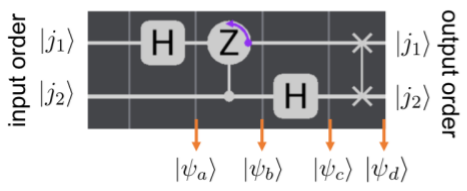
$$\text{QFT}\,|a\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} ka} |k\rangle$$

c) Product form:

$$|j_1, \ldots, j_n\rangle \rightarrow \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle}{\sqrt{2}}}_{|j_1\rangle} \otimes \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n} |1\rangle}{\sqrt{2}}}_{|j_2\rangle} \otimes \ldots \otimes \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_1 j_2 \ldots j_{n-1}j_n} |1\rangle}{\sqrt{2}}}_{|j_n\rangle}$$

qubit ordering

Exercise 6.2.1 a) Consider the 2-qubit QFT circuit below. Program in the QUI leaving a time-slice at the start to program different input states. Note the SWAP gate restores the qubit ordering to that of the input (to make the analytics in b) easier).



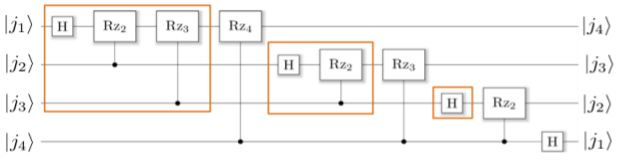
$$|j_1, \ldots, j_n\rangle \rightarrow \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle}{\sqrt{2}}}_{|j_1\rangle} \otimes \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n} |1\rangle}{\sqrt{2}}}_{|j_2\rangle} \otimes \ldots \otimes \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_1 j_2 \ldots j_{n-1}j_n} |1\rangle}{\sqrt{2}}}_{|j_n\rangle}$$

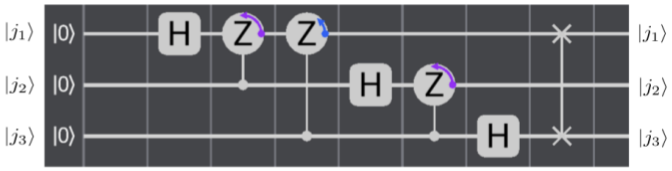
$$R_{z_2} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix} \equiv R_Z\left(\frac{\pi}{2}\right) \qquad \text{with} \quad \theta_g = \frac{\pi}{4}$$

b) Work out by hand the 2-qubit state at the points a-d indicated above for all possible basis state inputs, fill in the table below and run the scrubber through the circuit to verify your understanding.

$\|j_1\rangle\|j_2\rangle$	$\|\psi_a\rangle$	$\|\psi_b\rangle$	$\|\psi_c\rangle$	$\|\psi_d\rangle$
$\|00\rangle$	$\frac{1}{\sqrt{2}}(\|0\rangle+\|1\rangle)\|0\rangle = \frac{1}{\sqrt{2}}(\|00\rangle+\|10\rangle)$	$\frac{1}{\sqrt{2}}(\|0\rangle+\|1\rangle)\|0\rangle = \frac{1}{\sqrt{2}}(\|00\rangle+\|10\rangle)$	$\frac{1}{2}(\|00\rangle+\|01\rangle+\|10\rangle+\|11\rangle)$	$\frac{1}{2}(\|00\rangle+\|01\rangle+\|10\rangle+\|11\rangle)$
$\|01\rangle$				
$\|10\rangle$				
$\|11\rangle$				

Exercise 6.2.2 a) Program the 3-qubit circuit below (save as "QFT3"). Make sure you understand the construction of the required R_z rotation gates using the R-gate in the QUI (including setting the global phase – which is required now that the gates are controlled and no longer on a single qubit).





$$R_{z_k} = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

Rotation gates in the QUI

$$R_Z(\theta_R) = e^{i\theta_g}\left[I\cos\frac{\theta_R}{2} - iZ\sin\frac{\theta_R}{2}\right] = e^{i\theta_g}e^{-i\theta_R/2}\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta_R} \end{pmatrix}$$

$$R_{z_2} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix} \equiv R_Z\left(\frac{\pi}{2}\right) \quad \text{with} \quad \theta_g = \frac{\pi}{4}$$

$$R_{z_3} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \equiv R_Z\left(\frac{\pi}{4}\right) \quad \text{with} \quad \theta_g = \frac{\pi}{8}$$

b) Run the circuit for various input starting states – i.e. single basis states of the form $|a\rangle$ where a is an integer represented in bit form by the state $|j_1 j_2 j_3\rangle$. Examine the output phases and verify that the circuit produces the correct Fourier transform, i.e.:

$$\text{QFT}\,|a\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi i}{N}ka}\,|k\rangle$$

Exercise 6.2.3 Repeat 6.2.2 for a 4-qubit QFT circuit with SWAPs introduced to ensure the output has the same bit ordering as the input (save as "QFT4"). A nice way to check if the circuit is performing properly is to set the input to the maximum integer 1111 = 15 and watch the output phases run around the circle with the appropriate angle step.

Exercise 6.2.4 Using the editing features in the QUI (copy the controlled rotation gate, insert time slice and move, edit the angles etc) build on "QFT4" to create "QFT5" and so on up to "QFT8". For each circuit check the output using the maximum integer state.

6.3 Inverse QFT

As we saw in lectures, to construct the inverse QFT we simply need to make the Hermitian conjugate of every gate (for the Rz gates this amounts to changing the sign of both the rotation and global phase angles) and reverse the order in which they are applied.

Exercise 6.3.1 Load "QFT3", copy and paste the QFT circuit to the right and edit the R-gates to construct the inverse QFT. Test it out by sending various states through the circuit. Save the circuit as "QFT3-InvQFT3"

Exercise 6.3.2 Repeat for the other QFT circuits of increasing qubit number.

6.4 Phase-based arithmetic – using the QFT to add numbers

Consider two integers a and b in binary form over n-bits:

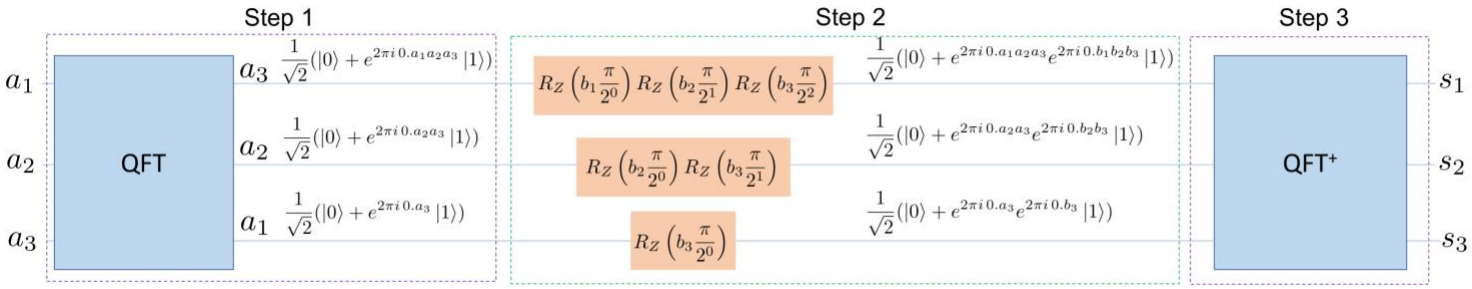
$$a = a_1 2^{n-1} + a_2 2^{n-2} + ... + a_{n-2} 2 + a_n$$
$$b = b_1 2^{n-1} + b_2 2^{n-2} + ... + b_{n-2} 2 + b_n$$

(note the convention is the same as our bit-ordering for the QFT lectures).

We wish to code a circuit based on your QFT circuits that calculates the sum (modulo N=2^n):

$$s = a + b = s_1 2^{n-1} + s_2 2^{n-2} + ... + s_{n-2} 2 + s_n$$

The schematic below shows how we are going to do this by changing the phase of each qubit according to the value of the bit of the number b. This is the basic idea behind the "Draper" adder.



Let's go through the steps:

Step 1: The a-register undergoes a QFT. We have no SWAPs so the bit ordering is reversed as shown so the least significant bit is up the top at the output.

Step 2: The way the circuit works is by incorporating phases predicated on the values of the bits in the number b. We do this by putting in the Z-rotation gates as shown (same convention as for the QFT: global phase is half the rotation angle etc). As an example, here is how the bottom qubit acquires a phase from the corresponding Z-rotation:

$$R_Z\left(b_3\frac{\pi}{2^0}\right)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, 0.a_3}|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, 0.a_3}e^{\pi i\, b_3}|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, 0.a_3}e^{2\pi i\, b_3/2}|1\rangle\right)$$
$$= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, 0.a_3}e^{2\pi i\, 0.b_3}|1\rangle\right)$$

Make sure you understand how the rotations (in the orange panels) produce the resulting states in Step 2. Remember the bit-wise fractions in the exponential products **add** (hint):
e.g.

$$e^{2\pi i\, 0.a_2a_3}\, e^{2\pi i\, 0.b_2b_3} = e^{2\pi i\, [0.a_2a_3 + 0.b_2b_3]}$$

Note we can specify the bits b1, b2, b3 in the gates themselves ("single register Draper adder"), as suggested by the above form, or we could control these rotations from a "b-register of qubits ("two register Draper adder").

Step 3: Here's where the magic happens – performing an inverse QFT the state produced by Step 2 puts the resulting addition of phases back into the "normal" state form of the binary number $|s\rangle = |s_1 s_2 s_3\rangle$. The bit-wise operations in the inverse QFT also ensure the carry is done automatically and the resulting addition is exact modulo $N=2^n$.

Exercise 6.4.1 Load the file "QFT3-InvQFT3", delete and SWAP gates and insert some time-blocks in the middle just after the QFT. Now add R-gates in the centre and specify the rotation angles according to your choice of the number b (and hence b1, b2, b3). Show that the circuit does indeed add b to the input a, modulo $N=2^n$. Change the inputs a and b. Try putting the register a into various superpositions of several numbers (not necessarily with equal amplitudes!) and run the circuit.

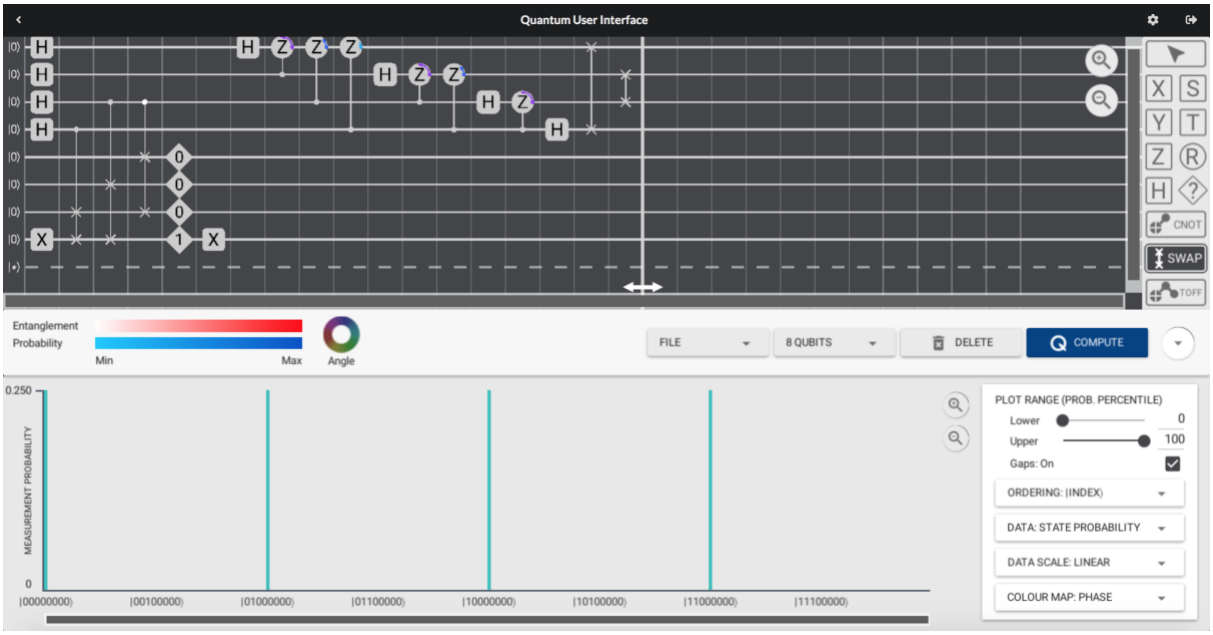
Exercise 6.4.2 Add 3 more qubits to represent the b register. Add controls to the central phase gates and show the circuit performs the addition a + b (modulo $N=2^n$). Try it with various superpositions of inputs.

Exercise 6.4.3 Repeat 6.4.1 and 6.4.2 for the larger QFT circuits.

6.5 Shor's quantum factoring algorithm

With the knowledge you know have of basic quantum arithmetic and QFT we have finally reached the point where we can encode and understand an instance of Shor's algorithm, albeit a specially designed case to make it not too lengthy on the QUI – we'll call this "Baby Shor". The full circuit for Shor's algorithm including the modular arithmetic is in general quite complicated. However, with some prior knowledge about the number we are factoring we can make some simplifications and still convey the same information. The circuit for Baby Shor is shown in the screenshot and described below (ref: Greg White).

Baby Shor, as one would expect, factors 15 by computing $f(x) = 2^x \bmod 15$ (i.e. the parameter a is set to a = 2), measuring out the $|f(x)\rangle$ register (bottom 4 qubits), and then performing a QFT on the resulting superposition over the $|x\rangle$ register (top 4 qubits).



Recall from lectures that the more general circuit for Shor's algorithm has two registers: an $|x\rangle$ register which is 2L qubits, and an L qubit register – initialised to $|1\rangle$, on which we compute $|f(x)\rangle$ (L is the number of bits in N). The first simplification we make is reducing the $|x\rangle$ register to L qubits, a move which will be later justified.

We compute the modular exponential through repeated modular multiplication. Writing $x = x_1x_2x_3x_4 = 2^3x_1 + 2^2x_2 + 2^1x_3 + 2^0x_4$, we do four multiplications on the $|f(x)\rangle$ register, controlled by each qubit of the $|x\rangle$ register. With each step, we multiply $|f(x)\rangle$ by 2^{2^i}, controlled by the x_{4-i} qubit. This is the same as multiplying by $2^{2^i x_{4-i}}$, since if $x_{4-i} = 0$, then we multiply by 1 – which is the same as doing nothing.

Step-by-step, the $|f(x)\rangle$ register follows the pathway:

$$|1\rangle \rightarrow |1 \times 2^{2^0 x_4} \bmod 15\rangle \rightarrow |1 \times 2^{2^0 x_4} \times 2^{2^1 x_3} \bmod 15\rangle \rightarrow |1 \times 2^{2^0 x_4} \times 2^{2^1 x_3} \times 2^{2^2 x_2} \bmod 15\rangle$$
$$\rightarrow |1 \times 2^{2^0 x_4} \times 2^{2^1 x_3} \times 2^{2^2 x_2} \times 2^{2^3 x_1} \bmod 15\rangle = |2^{2^3 x_1 + 2^2 x_2 + 2^1 x_3 + 2^0 x_4} \bmod 15\rangle$$
$$= |2^x \bmod 15\rangle$$

Now comes the main hack : general multiplications are difficult to perform on quantum circuits, but we have chosen a=2 (i.e. $2^x \bmod 15$) to make things easy. Consider each multiplication:

$$2^{2^0} \bmod 15 = 2$$
$$2^{2^1} \bmod 15 = 4$$
$$2^{2^2} \bmod 15 = 1$$
$$2^{2^3} \bmod 15 = 1$$

We notice firstly that the last two multiplications are multiplications by 1, which is no action. So we need only to compute the first two multiplications. We next notice that the first two are multiplications by powers of 2. There is something special about multiplying binary numbers by a power of 2. If we write $j = 2^3j_1 + 2^2j_2 + 2^1j_3 + 2^0j_4$, then $2^n \times j = 2^{3+n}j_1 + 2^{2+n}j_2 + 2^{1+n}j_3 + 2^{0+n}j_4$. That is, we are shifting each of our numbers to the left by one position. This is not a new concept. Think about how you multiply numbers by 10, or 100 in base 10. This is done by shifting all of your numbers to the left by 1 or 2 positions. Also, recall that binary numbers are inherently defined modulo 2^n.

In terms of gates, shifting our bits is the same as swapping their positions. We start out at 0001 and have a controlled multiplication of 2, which would give 0010. This is the same as swapping x_3 and x_4. Similarly, bit-shifting by two positions means taking $x_1x_2x_3x_4 \rightarrow x_3x_4x_1x_2$ – achieved through two SWAPS. Result – our complicated exponentiation circuit can be reduced to just three controlled SWAP gates for this simple case!

This means now that our circuit for Shor's algorithm can be summarised by:
1) a controlled shift by x_4 of 1 position acting on the $|f(x)\rangle$ register
2) a controlled shift by x_3 of 2 positions acting on the $|f(x)\rangle$ register
3) a measurement of the $|f(x)\rangle$ register to isolate one superposition in the $|x\rangle$ register
4) a QFT on the $|x\rangle$ register
5) Measure the $|x\rangle$ register (or in the QUI just inspect the state)

The reason why we can reduce the $|x\rangle$ register down to L qubits is related to the period. A domain of 2^{2L} is usually chosen in order to guarantee that we can find the period through continued fractions, in the case that we do not have exact periodicity. However, in the simple scenario here the period is 4, which divides 2^4, and so we do not need the extra L qubits.

Finally, note that the X operation on the last qubit is for convenience in reading out the final measurement when the bottom register is measured as 0001, which after the X operation becomes 0000.

Exercise 6.6.1 Code up Baby Shor in the QUI (use a previous instance of QFT4). Save it and run it several times to understand how it works (you can edit appropriately to verify the function f(x) evaluation), what the output means (you can add measurements on the x register, or just interpret the final state), and how to determine the prime factors of 15 (refer back to lecture 10).

MULT90063 Introduction to Quantum Computing

Lab Session 7

7.1 Introduction

Welcome to Lab 7 of MULT90063 Introduction to Quantum Computing.

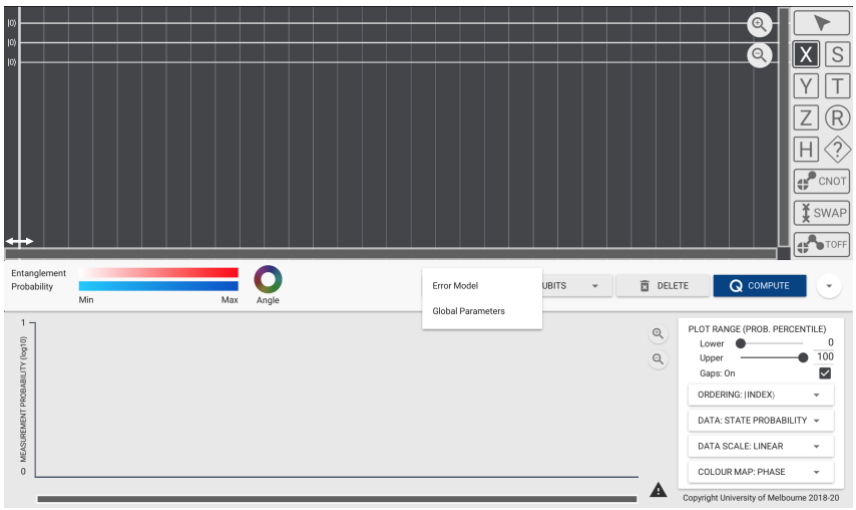
The purpose of this lab session is to:

- understand and implement rotation errors in quantum circuits
- understand and implement quantum supremacy (QS) circuits
- implement rotations errors in QS circuits and understand their effect

7.2 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 an error model. You can set the error model under the "Settings" drop-down in QUI:



When you select "Error Model" you are given a choice between no errors, a discrete or continuous error model:

We will first consider control errors, which can be modelled by a continuous error model. We saw in lectures that a Z-rotation error (or just "Z-error") can be considered an unwanted 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 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" continuous 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. These are modelled in QUI as continuous errors.

Exercise 7.2.1 Let's go back to our favourite 3-gate (interference) example, HTH, and examine the effect of rotation errors on the circuit:

a) Z-errors. Program the HTH_ circuit in the QUI. Apply an error model with angles having a mean of zero and a standard deviations of ε around the z-axis. Notice that when you run the circuit with a non-zero error model, the location of errors are indicated in your circuit. Gather statistics about the outcomes which would be measured after this circuit, by running this circuit by running the circuit many times, and averaging over the outcome probabilities reported by QUI for the zero state and one state.

Compare the effect of running the the "pristine" no-error output to those with increasingly large error by filling in the table below:

Table 1: The effect of Z-errors on HTH

ε	Output state $\|\psi\rangle = a_0\|0\rangle + a_1\|1\rangle$	Output state change	Output state relative change
None	$p_0\|_{\text{exact}} = 0.854$ $p_1\|_{\text{exact}} = 0.146$	$\Delta p_0 = 0$ $\Delta p_1 = 0$	$\frac{\Delta p_0}{p_0\|_{\text{exact}}} = 0$

			$\frac{\Delta p_1}{p_1\|_{\text{exact}}} = 0$
0.1 π			
0.2 π			
0.5 π			
π			

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

b) X-errors. Repeat for the case of x-errors, and examine the effect of different error rates.

Table 1: The effect of X-errors on HTH

ε	Output state $\|\psi\rangle = a_0\|0\rangle + a_1\|1\rangle$	Output state change Compared to	Output state relative change
None	$p_0\|_{\text{exact}} = 0.854$ $p_1\|_{\text{exact}} = 0.146$	$\Delta p_0 = 0$ $\Delta p_1 = 0$	$\frac{\Delta p_0}{p_0\|_{\text{exact}}} = 0$ $\frac{\Delta p_1}{p_1\|_{\text{exact}}} = 0$
0.1 π			
0.2 π			
0.5 π			
π			

Compare these results with those you found in part (a).

7.3 Effect of rotation errors in quantum circuits

Obviously, not all errors have the same effect – it will depend on the type of error 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 7.3.1 Load the file "Lab 5 Grover 3 oracle marks 5", run it and 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%.

Add a continuous error model, with different levels standard deviation in the angle, and examine the resulting states, determine the probability of success of the algorithm:

ε (z-error)	Probability of finding the marked state
None	94.5%
0.01 π	
0.05 π	
0.10 π	
π	

Repeat for Z and Y errors independently (also at the $\varepsilon = 0.1\pi$ level).

Exercise 7.3.3 Repeat for the 3-qubit QFT adder circuit. What is the maximum error level you would tolerate in the adder?

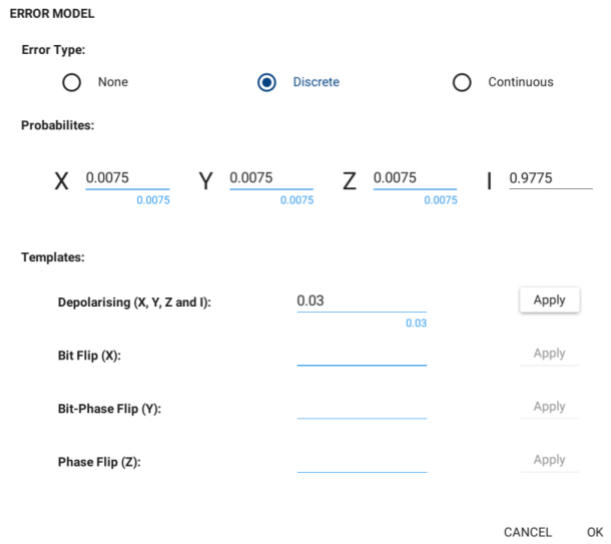
7.4 Types of errors: control precision and decoherence

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.

In the QUI this is implemented as a discrete error model.

Exercise 7.4.1 Load the three qubit Grover's search. In this algorithm, 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.

Choose the discrete error model, and implement "Depolarising" noise, with a probability of 3% error, as shown below:



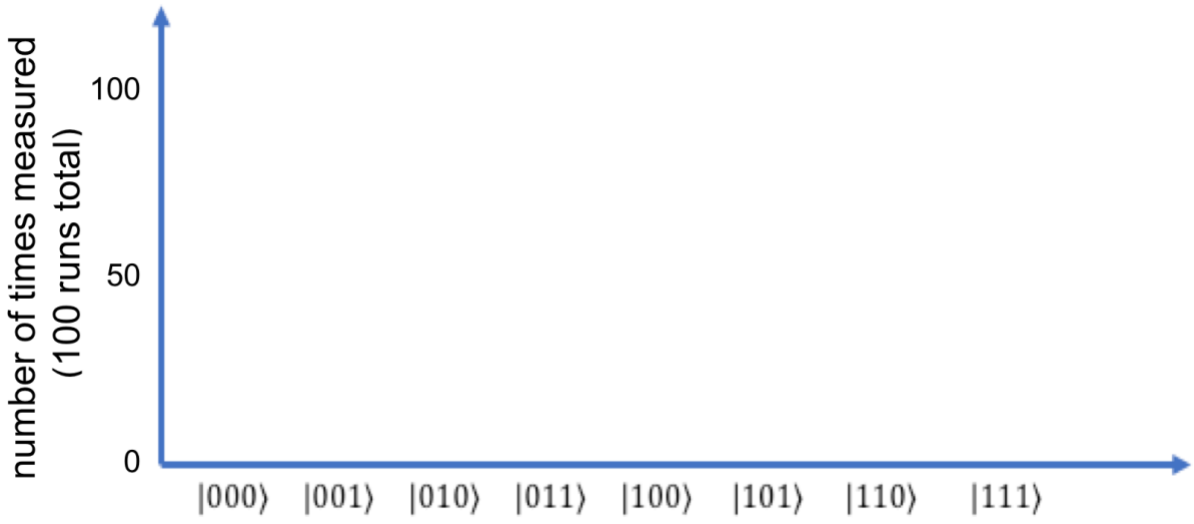
Run the circuit multiple times, and record the outcome of each run:

Table of results:

Measurement outcome	Record 𝍸…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%



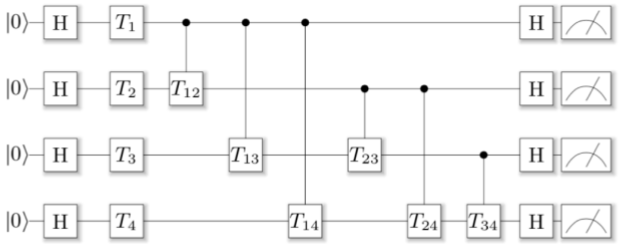
From these results estimate the probability of the algorithm producing a correct result in the presence of the 3% error rate.

To do: Now incorporate X, Y and Z errors at random (at various effective densities, and effective error angles) in some of the circuits you have programmed previously and examine the effect on the outputs. For example:

- Toffoli based adders
- Grover search
- QFT and QFT adders
- Baby Shor

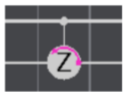
7.5 Quantum supremacy – IQP circuits

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



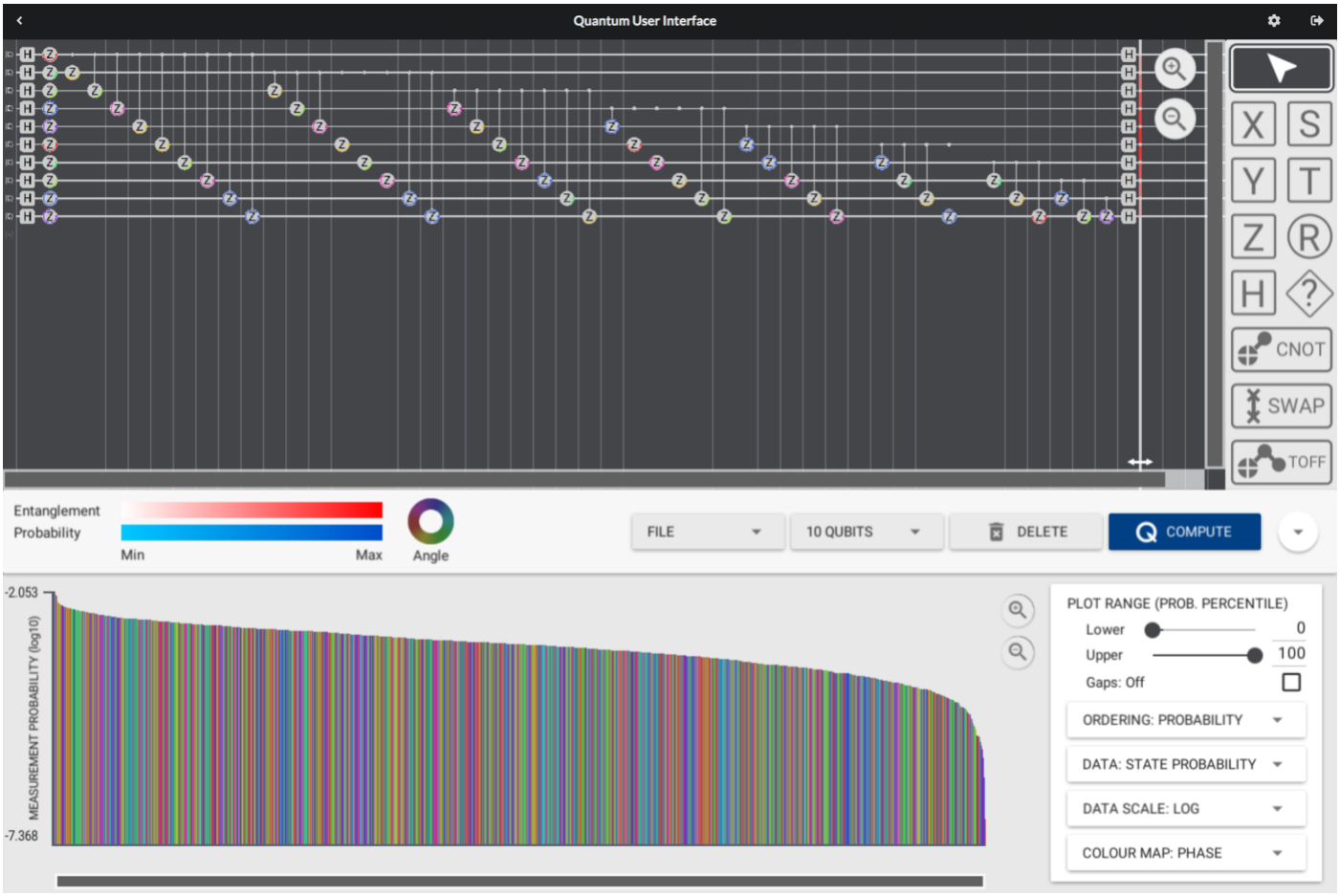
$$T_m = \cos\left(\frac{k_m\pi}{8}\right) I + i\sin\left(\frac{k_m\pi}{8}\right) Z_m \quad \rightarrow R_Z\left(\frac{-k_m\pi}{4}\right)\Big|_{\theta_g=0}$$

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



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

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. For the two-qubit gates, use controlled-z rotations with a random angle which is a multiple of $\pi/4$. Apply a discrete error model and observe the change in outcome as different errors are applied.



7.5 Randomised benchmarking

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($\pi/2$) 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 Program a random 5-gate sequence (with spaces as shown) using gates chosen from the set {X, Y, Z, H, RX($\pi/2$), RY($\pi/2$) RZ($\pi/2$)} and run to check it produces the $|0\rangle$ state at the end (i.e. that the inverse sequence is correct). Choose a depolarising error model, with 3% probability and rerun the sequence. Run the sequence several times and determine the probability of measuring the $|0\rangle$ 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 $|0\rangle$ 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 using the formula:

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

MULT90063 Introduction to Quantum Computing

Lab Session 8

8.1 Introduction

Welcome to Lab 8 of MULT90063 Introduction to Quantum Computing. The purpose of this lab session is to get some experience with quantum error correction (QEC).

8.2 Quantum Error Correction preliminaries

As you saw in lectures classical concepts of error correction need to be adapted to the quantum framework. The first thing we will explore is the notion of commuting and anti-commuting operators on states, and the products of operators that "stabilise" certain states.

Exercise 8.2.1 Operators and stablisers.

a) In the QUI we will create an "arbitrary" 1-qubit superposition state $|\psi\rangle$ to experiment on – I used $R_x(0.345\pi)$ with global phase set at 0.5π. Show by inspecting the states in the QUI that the following operator commutation properties hold on this state (or additionally any other state by doing some maths for the general case):

$$XZ\,|\psi\rangle = -ZX\,|\psi\rangle \rightarrow XZ = -ZX$$
$$XY\,|\psi\rangle = -YX\,|\psi\rangle \rightarrow XY = -YX$$
$$ZY\,|\psi\rangle = -YZ\,|\psi\rangle \rightarrow ZY = -YZ$$

Beware the trap: when write $XZ|\psi\rangle$ we mean the Z operator is applied first followed by the X operator, so in the QUI time ordering left to right it's programmed as –Z–X– etc.

b) In the QUI create an "arbitrary" 2-qubit superposition state $|\psi\rangle$ to experiment on (e.g. using some R-gates and CNOT). Show that the following operator commutation property holds:

$$X_1X_2Z_1Z_2\,|\psi\rangle = Z_1Z_2X_1X_2\,|\psi\rangle \rightarrow X_1X_2Z_1Z_2 = Z_1Z_2X_1X_2$$

Again, beware the operator ordering trap.

c) Show that the states $|000\rangle$ and $|111\rangle$ (the 3-bit code) are stabilised as:

$$Z_1Z_2\,|000\rangle = |000\rangle \qquad Z_1Z_2\,|111\rangle = |111\rangle$$
$$Z_2Z_3\,|000\rangle = |000\rangle \qquad Z_2Z_3\,|111\rangle = |111\rangle$$

d) Show using an example in the QUI that an arbitrarily chosen linear combination of states $|000\rangle$ and $|111\rangle$ is also stabilised by the operator products Z_1Z_2 and Z_2Z_3 (the "stabilisers"). Hint: to create the states, think back to the GHZ states you encountered in previous labs.

Exercise 8.2.2 In the QUI create a non-trivial linear combination of $|000\rangle$ and $|111\rangle$. Apply an "X-error" on the first qubit and determine the "syndrome" for each of the stabilisers products Z_1Z_2 and Z_2Z_3 as per:

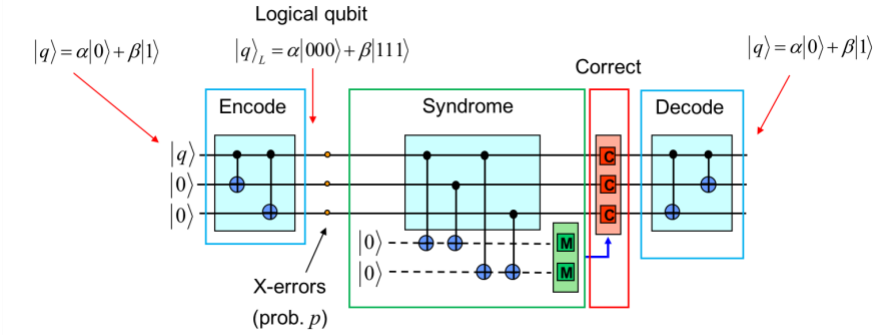
$$Z_1Z_2\,|\psi'\rangle = Z_1Z_2X_1\,|\psi\rangle = -X_1Z_1Z_2\,|\psi\rangle = -X_1\,|\psi\rangle = -\,|\psi'\rangle$$

Now complete the stabiliser table for the 3-bit code:

Error	State	$Z_1 Z_2$	$Z_2 Z_3$
I	$\alpha\|000\rangle + \beta\|111\rangle$	+1	+1
X_1	$\alpha\|100\rangle + \beta\|011\rangle$	-1	
X_2	$\alpha\|010\rangle + \beta\|101\rangle$		
X_3	$\alpha\|001\rangle + \beta\|110\rangle$		

8.3 3-bit Quantum Error Correction code

In the lectures we presented the 3-bit code implemented in the following circuit where ancilla qubits are introduced to measure the stabilisers to obtain the error syndrome for correction:



We have distinguished the various step of the QEC process – encode, error exposure, syndrome, correct, decode. Note that the set of stabilisers chosen in the above is $\{Z_1Z_2, Z_1Z_3\}$ – the syndrome assignments will change, but there will be 4 distinct ancilla measurement outcomes to instruct us how to correct.

Exercise 8.3.1 The encode circuit is relatively easy to follow (see 8.2.2 above). Refer to your lecture notes to understand how the syndrome circuit works in terms of measuring the stabilisers using ancilla qubits.

Exercise 8.3.2 In the QUI, program the encode and syndrome steps of the circuit based on the stabilisers $\{Z_1Z_2, Z_2Z_3\}$. Starting from an arbitrarily chosen state $|\psi\rangle = [\alpha|0\rangle + \beta|1\rangle]\otimes|0\rangle\otimes|0\rangle$ (e.g. using the state in 8.2.1a) run the circuit with the four X-error scenarios and record the ancilla result for each. Fill in the table:

Scenario	Error	State	$Z_1 Z_2$	$Z_2 Z_3$	Ancillas
No error	I	$\alpha\|000\rangle + \beta\|111\rangle$	+1	+1	00
Error on 1st	X_1	$\alpha\|100\rangle + \beta\|011\rangle$	-1	+1	10
Error on 2nd	X_2	$\alpha\|010\rangle + \beta\|101\rangle$	-1	-1	11
Error on 3rd	X_3	$\alpha\|001\rangle + \beta\|110\rangle$	+1	-1	01

Exercise 8.3.3 Now add the decode circuit, leaving some room for a correction step. Run the circuit with various error scenarios and manually apply the correction required to show the original state $|\psi\rangle = [\alpha|0\rangle + \beta|1\rangle]\otimes|0\rangle\otimes|0\rangle$ is recovered. When comparing with the system state before the error was applied don't forget the final state has the final two qubits (ancillas) changed – to make the direct comparison insert the appropriate X operator(s) to flip the ancillas back to their initial values.

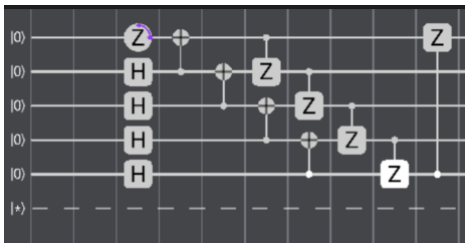
Exercise 8.3.4 Now we can automate the process. Using your knowledge of Toffoli based classical logic build a circuit to perform the correction based on the measured ancilla values. Yes, this is possible in the QUI. Test your circuit on the four error scenarios. Again, to achieve an easy direct comparison manually flip the ancillas where required (OK, this you have to do manually in the QUI).

8.4 5-bit Quantum Error Correction code

In the lectures we presented the 5-bit code, which can correct X, Z and Y errors, in terms of the following stabilisers:

$$\begin{cases} IXZZX \\ XIXZZ \\ ZXIXZ \\ ZZXIX \end{cases}$$

The encoding circuit for the 5-bit code is:



Exercise 8.4.1 Program the above 5-bit encoding circuit into the QUI and run it (with appropriate input states) to obtain the logical states $|0\rangle_L$ and $|1\rangle_L$ in the expressions below (cross out the states not present and add signs).

$$|0\rangle_L = \frac{1}{4}[|00000\rangle \quad |00001\rangle \quad |00010\rangle \quad |00011\rangle$$
$$|00100\rangle \quad |00101\rangle \quad |00110\rangle \quad |00111\rangle$$
$$|01000\rangle \quad |01001\rangle \quad |01010\rangle \quad |01011\rangle$$
$$|10000\rangle \quad |10001\rangle \quad |10010\rangle \quad |10011\rangle]$$

$$|1\rangle_L = \frac{1}{4}[|00000\rangle \quad |00001\rangle \quad |00010\rangle \quad |00011\rangle$$
$$|00100\rangle \quad |00101\rangle \quad |00110\rangle \quad |00111\rangle$$
$$|01000\rangle \quad |01001\rangle \quad |01010\rangle \quad |01011\rangle$$
$$|10000\rangle \quad |10001\rangle \quad |10010\rangle \quad |10011\rangle]$$

Exercise 8.4.2 From the example in the lecture notes build the syndrome extraction circuit. There are a total of 16 stabiliser combinations, so you will need 4 ancilla qubits (testing the very limits of the QUI configuration).

Scenario	Error	$I_1 X_2 Z_3 Z_4 X_5$	$X_1 I_2 X_3 Z_4 Z_5$	$Z_1 X_2 I_3 X_4 Z_5$	$Z_1 Z_2 X_3 I_4 X_5$	Ancillas
No error	I	+1	+1	+1	+1	0000
X error on 1st	X_1					
X error on 2nd	X_2					
X error on 3rd	X_3					
Y error on 1st	Y_1					
Y error on 2nd	Y_2					
Y error on 3rd	Y_3					
Z error on 1st	Z_1					
Z error on 2nd	Z_2					
Z error on 3rd	Z_3					

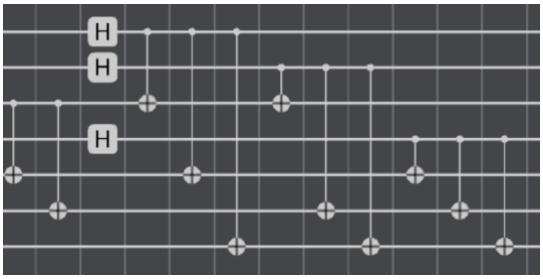
Exercise 8.4.3 Now build the correction circuit to implement the appropriate correction for each ancilla measurement scenario. (is this possible given Y and Z correction?). Or just do it for X errors. Save your circuit as “Lab 8 5 bit QEC template”.

8.5 The Steane 7-bit Quantum Error Correction code

In the lectures we presented the 7-bit code, which can correct X, Z and Y errors, in terms of the following stabilisers:

$$\begin{cases} I & I & I & X & X & X & X \\ I & X & X & I & I & X & X \\ X & I & X & I & X & I & X \\ I & I & I & Z & Z & Z & Z \\ I & Z & Z & I & I & Z & Z \\ Z & I & Z & I & Z & I & Z \end{cases}$$

The encoding circuit for the Steane code is:



Exercise 8.5.1 Program the encoding circuit into the QUI and verify that the logical states have the following structure.

$$\begin{aligned} |0_L\rangle = \frac{1}{\sqrt{8}}(&|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\ &+ |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle) \\ |1_L\rangle = \frac{1}{\sqrt{8}}(&|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\ &+ |111000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle) \end{aligned}$$

For the stabilizers given in labs, determine the corresponding measurement result expected for the ancilla qubits.

Scenario	Error	Ancillas	Scenario	Error	Ancillas	Scenario	Error	Ancillas
No error	I	000000						
X error on 1st	X_1		Y error on 1st	Y_1		Z error on 1st	Z_1	
X error on 2nd	X_2		Y error on 2nd	Y_2		Z error on 2nd	Z_2	
X error on 3rd	X_3		Y error on 3rd	Y_3		Z error on 3rd	Z_3	
X error on 4th	X_4		Y error on 4th	Y_4		Z error on 4th	Z_4	

X error on 5th	X_5		Y error on 5th	Y_5		Z error on 5th	Z_5	
X error on 6th	X_6		Y error on 6th	Y_6		Z error on 6th	Z_6	
X error on 7th	X_7		Y error on 7th	Y_7		Z error on 7th	Z_7	

8.6 Logical operations on the 5-bit Quantum Error Correction code

Now we can start doing some logical operations within a QEC code – i.e. perform logical gates on the logical encoding itself. To make things easier and tractable in the QUI we will focus on the 5 qubit code.

Exercise 8.6.1 Load the circuit "Lab 8 5 bit QEC template". Modify the circuit to perform one round of syndrome and correction, followed by a transverse logical operation (e.g. try each of X_L = XXXXX, Y_L = YYYYY, Z_L = ZZZZZ) followed by another round of syndrome and correction as per the following sequence:

a) In the no-error scenario, does your circuit work, i.e. is the logical operation performed correctly on the state?

b) Add single X, Y or Z errors in the locations specified and check how your circuit performs. Can you correctly perform the logical operation in the presence of errors? What happens to the output when there is more than one error in any given (indicated) location?

MULT90063 Introduction to Quantum Computing

Lab Session 9

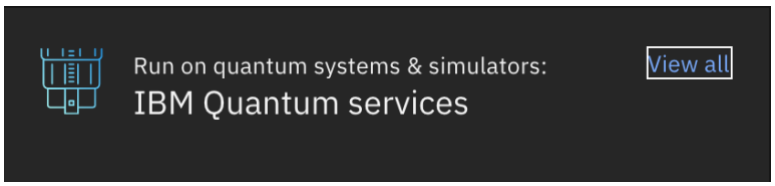
9.1 Introduction

Welcome to Lab 9 of MULT90063 Introduction to Quantum Computing. The purpose of this lab session is to program and run circuits in the IBM Quantum Experience.

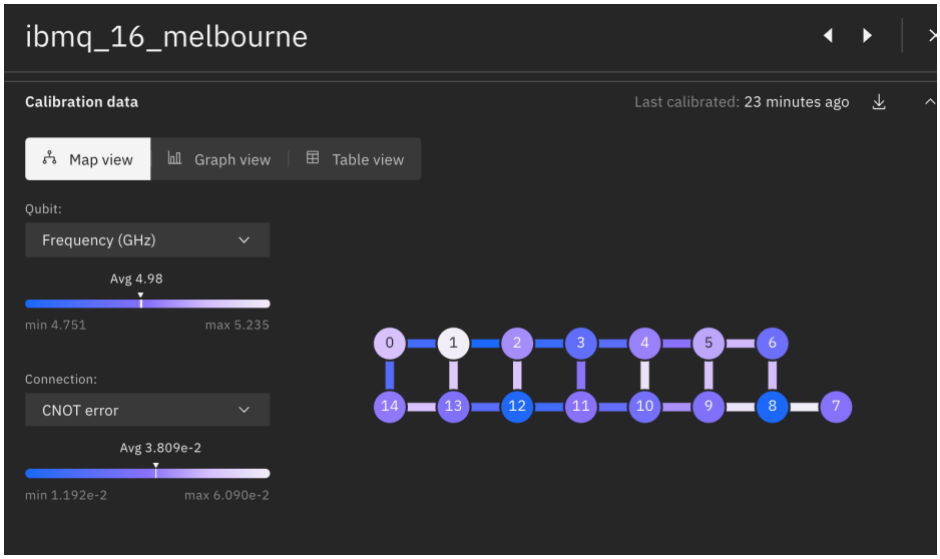
9.2 IBM Quantum Experience

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.

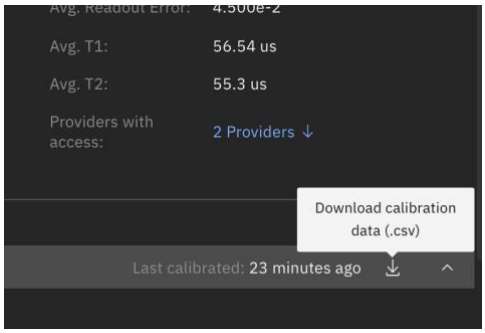
Hint: if you're programming a lot of gates switch to the QASM editor and then back to the composer to check. Before you run any experiment, be sure to save the circuit. If there is a cached version of your circuit, which you have used before, re-use it as waiting for machines may take time!



Exercise 9.2.1 When you access the system, if you click on "View all" IBM quantum services (as above) 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 Melbourne device:

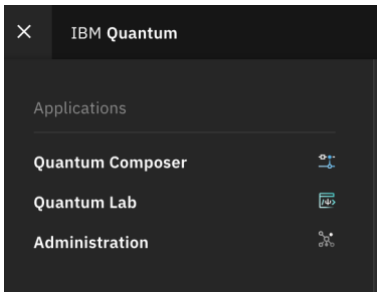


Choose a device which you will use during this session, and examine the details of error rates on CNOT and single qubit gates. Note that it also indicates which CNOT gates are allowed on this device. Record this as a screenshot for reference.

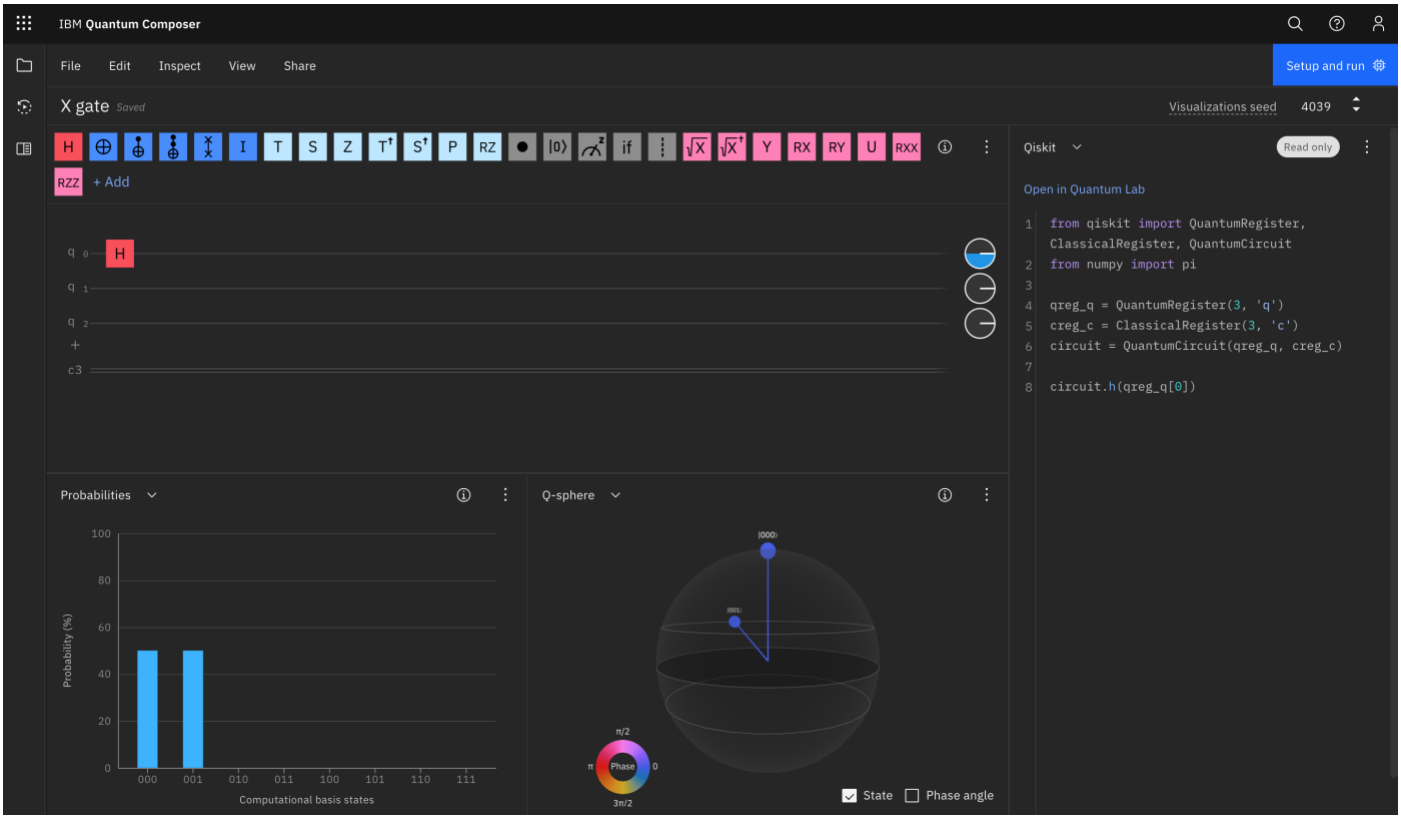


Additional details can be found by clicking on the "Download calibration data" button. Do this, and examine the file that you download. The T1 and T2 times quoted are an indication of how long qubits remain faithful to their state (i.e. stay coherent). T1 is related to how long it takes for say a $|1\rangle$ 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^o (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).

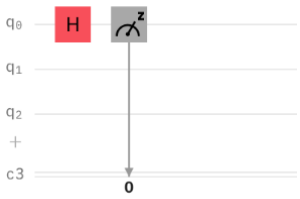
9.3 Superposition states



We will make use of the Circuit Composer functionality in the IBM experience website. Click on the "Circuit Composer" icon in the left sidebar.

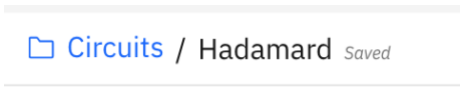


Exercise 9.3.1 Program an Hadamard gate to create an equal superposition state. Examine the different animations and information screens available to you, including the "Q-Sphere", measurement probabilities, and the code (both Qiskit and QASM) for the circuit. Be sure to also add a measurement gate to measure the outcome of the circuit. Your circuit should now look like this:

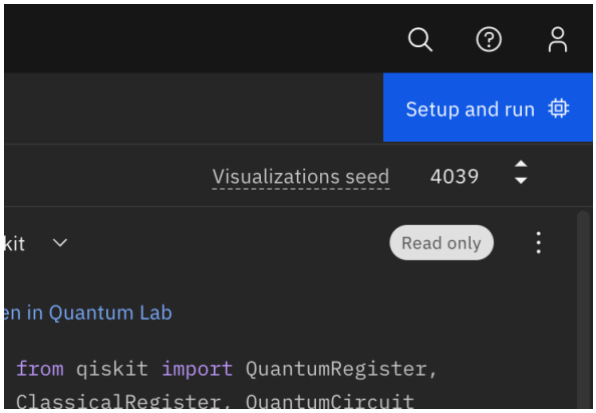


Note that you can add or remove qubits, by clicking on the qubit's name (q0, q1, q2…) or the plus sign.

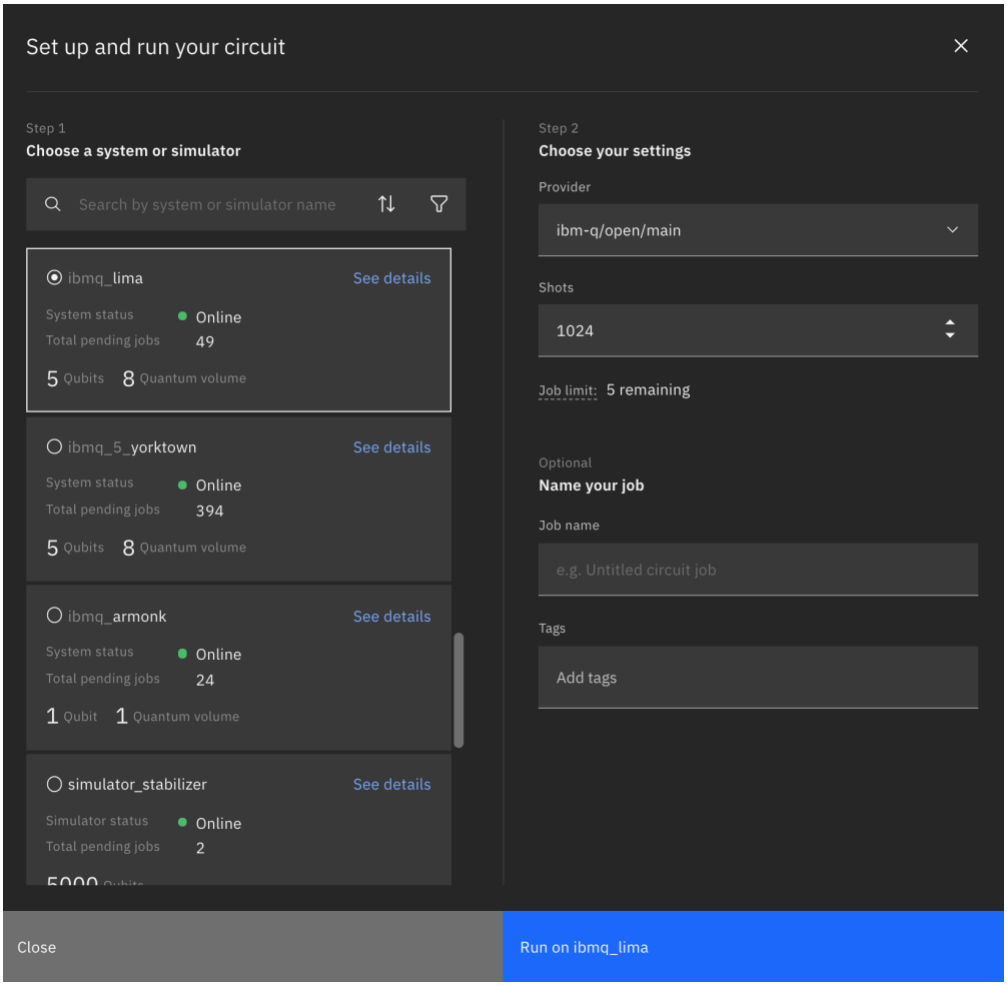
Choose a suitable filename of "Hadamard" (or similar) by clicking on the filename in the top left hand corner.



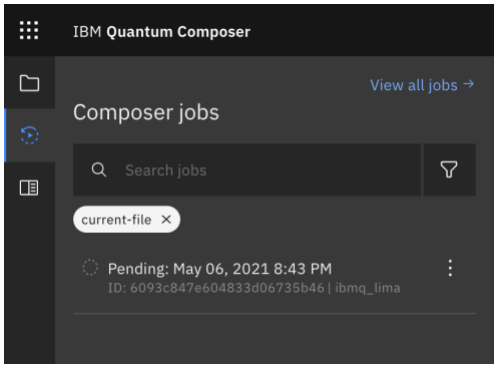
Now let us execute the circuit. First, select the "Setup and run" button in the top right.



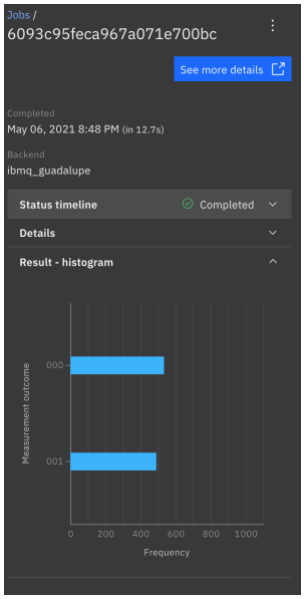
As directed, choose a device to run on, select a number of shots (1024 is fine), and optionally name your job. Once you are happy with your settings, press the blue “Run on …” button.



By clicking on the “Composer Jobs” tab (in the left hand side-bar), you should see your job queued, when it is has been run, you can also access the results of the jobs here. The queue might be several minutes, as there are many users of the quantum devices. You can submit 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, and by clicking "See more details" can additionally see 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.

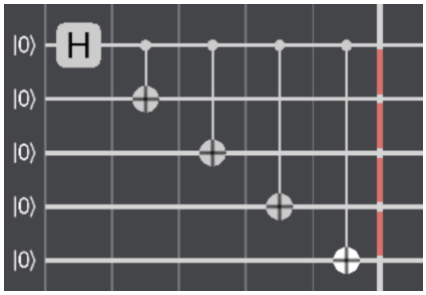
Pause for a second and appreciate what you're doing here – you are programming real qubits, getting actual results, and understanding (hopefully) what's happening.

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.

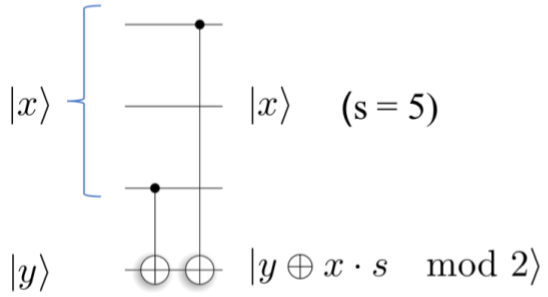
Exercise 9.5.1 Program and run a (non-trivial) instance of Deutsch-Josza. Again, you can use the QUI to investigate and understand how well the machine performs.

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 $f = x \cdot s \bmod 2$ 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.

MULT90063 Introduction to Quantum Computing

Lab Session 10

10.1 Introduction

Welcome to Lab 10 of MULT90063 Introduction to Quantum Computing. The purpose of this lab session is to get some experience with mapping and solving optimisation problems using hybrid approaches, e.g. Quantum Approximate Optimisation Algorithm (QAOA).

10.2 Programming quadratic forms

As you saw in lectures many optimisation problems can be cast into the "spin-glass" form where the task is to minimise the energy of a particular function E over variables $\{z_i\}$. The quadratic form (QUBO) involve products $z_i\, z_j$, so when we map the problem onto a quantum computer and convert variables z_i to operators Z_i we need to be able to program operator products $Z_i\, Z_j$ acting on a particular trial state.

Exercise 10.2.1 Consider the circuit below over two qubits: 1 (top) and 2 (bottom). For the case $\theta_{ij} = \pi$ (global phase $\pi/2$) verify that the circuit form below is equivalent to the operation Z_1Z_2. Hint – one way to do this is to write out the gates as 4x4 matrices over the two qubit space and multiply them out. Verify the action on the QUI.

$$H = \ldots + J_{ij}\, Z_i Z_j + \ldots$$

$$\theta_{ij} = -\frac{J_{ij}\alpha}{2}$$

10.3 Solving a number partitioning problem

Let's consider an instance of the number partitioning problem and solve it on the QUI. The problem is stated as follows: given a set S of numbers $\{w_i\}$, is there a partition of this set of numbers into two disjoint subsets R and S − R, such that the sum of the elements in both sets is the same? In lectures we saw the solution is equivalent to finding the basis state with the minimum energy in the system given by the "Hamiltonian" over qubit operators:

$$H = \left(\sum_i w_i Z_i\right)^2 = \sum_{i \neq j} 2 w_i w_j Z_i Z_j + \sum_i w_i^2 I$$

where and each number is assigned a qubit ($w_i \leftrightarrow q_i$) and the qubit value 0 or 1 indicates which of the two sets, R and S–R, the numbers are associated with.

We will solve the number partitioning problem for the set S = {1, 2, 3} using QAOA on the QUI.

Exercise 10.3.1 Classically solving the problem (i.e. in your head), what are the solution sets R and S–R?

Exercise 10.3.2 Convert classical solutions to quantum land. Assigning "R" ↔ "0" and "S–R" ↔ "1", translate the expected possible solutions from 10.3.1 into qubit form $|q_1 q_2 q_3\rangle$ where that values of each q_i = 0, 1 indicates which set the corresponding number w_i is assigned to in the solution. The solutions are:

_____ and _____

Exercise 10.3.3 The Hamiltonian of this number partitioning problem is

$$H = 4Z_1Z_2 + 6Z_1Z_3 + 12Z_2Z_3 + 14I$$

Make sure you understand how to get this from the general expression above. Show that the energy of this system is zero for the solution basis states $|q_1q_2q_3\rangle$ you found in 10.3.2, and non-zero for other states.

Exercise 10.3.4 Now we will program the QAOA circuit to solve this system. For the construction of the ZZ operators in the circuit we have from the Hamiltonian the individual energies: J_{12} = 4, J_{13} = 6, J_{13} = 12. Consider the basic QAOA circuit (k=1, i.e. one iteration), program this in the QUI for the choice $\alpha = 0.01\pi$ and $\beta = 0.3\pi$, and the values E_{ZZ}^{ij} given above for the problem. Run to compute the state $|\phi(\alpha,\beta)\rangle$.

To calculate the energy, we take the expectation value of the Hamiltonian H in the state $|\phi(\alpha,\beta)\rangle$ i.e.

$$\langle H\rangle_{\alpha\beta} = 4\langle Z_1Z_2\rangle_{\alpha\beta} + 6\langle Z_1Z_3\rangle_{\alpha\beta} + 12\langle Z_2Z_3\rangle_{\alpha\beta} + 14$$

where, $\langle\mathcal{O}\rangle_{\alpha\beta} = \langle\phi(\alpha,\beta)|\mathcal{O}|\phi(\alpha,\beta)\rangle$. Note that for a 3 qubit state written in terms of the basis states we have in general:

$$|\varphi\rangle = a_0|000\rangle + a_1|001\rangle + a_2|010\rangle + a_3|011\rangle + a_4|100\rangle + a_5|101\rangle + a_6|110\rangle + a_7|111\rangle,$$

The measurement of $\langle Z_1Z_2\rangle$ amounts to averaging the of Z_1Z_2 in each of the basis states (recall $Z|0\rangle = +|0\rangle$, $Z|1\rangle = -|1\rangle$ etc), weighted by the probabilities of the basis states,

$$\langle Z_1Z_2\rangle = \langle\varphi|Z_1Z_2|\varphi\rangle = (+1)|a_0|^2 + (-1)|a_1|^2 + (-1)|a_2|^2 + (+1)|a_3|^2$$
$$+ (-1)|a_4|^2 + (-1)|a_5|^2 + (+1)|a_6|^2 + (+1)|a_7|^2.$$

On a real QC we would have to run many times to get these expectation values. In the QUI we have the probabilities $|a_i|^2$ for each basis state (i = decimal index), and can sum the expectation values of the terms in the Hamiltonian. Notice that we could also work out the expectation value of the sum of terms in the Hamiltonian with respect to each of the basis states and sum those over the basis states (which we will do in the next exercise).

Exercise 10.3.5 From the QUI, let's break down the expectation values over the individual basis states. The table you want to fill out from the QUI output is given below. Best to do this in excel because you will be repeating it for other choices of α and β (yes, it's one of those labs, but we'll try to make it as painless as possible).

Basis state, i	$\langle Z_1Z_2\rangle_i$	$\langle Z_1Z_3\rangle_i$	$\langle Z_2Z_3\rangle_i$	$E_i = 4\langle Z_1Z_2\rangle_i + 6\langle Z_1Z_3\rangle_i + 12\langle Z_2Z_3\rangle_i + 14$	QUI $\lvert a_i\rvert^2$ $\alpha = 0.01\pi$ $\beta = 0.3\pi$	$\lvert a_i\rvert^2 E_i$
$\lvert 000\rangle$	1	1	1	36	0.215	7.74
$\lvert 001\rangle$						
$\lvert 010\rangle$						
$\lvert 011\rangle$						
$\lvert 100\rangle$						
$\lvert 101\rangle$						
$\lvert 110\rangle$						
$\lvert 111\rangle$						
					Total energy	20.16

Exercise 10.3.6 For the best values of parameter which you found: What does the QUI output state look like? Are the states corresponding to the solutions higher in probability?

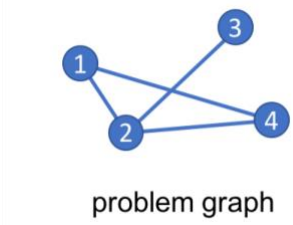
Exercise 10.3.7 From whatever is your best choice of α and β, add another QAOA iteration (see lectures: i.e. k=2) and see if you can get a better fidelity solution.

10.4 Graph partitioning

In this part of the lab you will be using a hybrid quantum variational method to solve a graph partitioning problem. We will divide this task into three stages (i) Write down the target Hamiltonian, (ii) encode a trial wavefunction (iii) measure the energy and the probability of success and (iv) optimize the parameters in the circuit to increase the probability of success.

The computational problem which we would like to solve is this: Given a graph, partition the vertices into two equal subsets with the minimum number of edges between the subsets.

Here is the example we will use in the lab:



problem graph

Exercise 10.4.1 Find the solution for the example given. Which two subsets, each with two vertices, which give the minimum number of edges between the two subsets? Draw answer here (there may be more than one).

Exercise 10.4.2 Quantum version of the solution. In the lecture we discussed how to map this to a QUBO (quadratic unconstrainted binary optimisation) problem – and its associated Hamiltonian/total energy. The first step is to associate each vertex with a particular qubit. If the qubit is in state 0, that indicates that vertex is in subset "0", if it is in state 1, that indicates the vertex is in subset "1".

Given your answer for 10.4.1, which two states correspond to correct solutions to this problem?

_____ and _____

Our next task is to find the Hamiltonian whose minimum energy encodes the answer to this optimization problem. We do this in two parts – considering vertices and edges. If there are an equal number of vertices in subset "0" and subset "1", we should have that the sum of the z-eigenvalues in our solution states will be zero:

$$\sum_{\substack{\text{vertices,}\\ i}} z_i = 0$$

Equivalently, if we can find a state is the lowest energy (ground state) of the operator form of the energy acting on qubits (each corresponding to a vertex):

$$H_A = \left(\sum_{\text{qubit, } i} Z_i \right)^2$$

then it will have an equal number of 0's as 1's in the solution, and hence an equal number of vertices in each subset.

Exercise 10.4.3 For the problem graph given, write out the (vertex) Hamiltonian, H_A:

We also need to minimize the number of edges which connect between the two. A Hamiltonian whose energy is equal to the number of edges between the two subsets is:

$$H_B = \sum_{i,j \in E} \frac{I - Z_i Z_j}{2}$$

where E is the set of edges (i.e. only include edges in the graph, and don't double count – the operators in the product commute).

Exercise 10.4.4 For the problem graph given, write out the (edge) Hamiltonian, H_B:

The final Hamiltonian is linear combination of these two Hamiltonians:

$$H = AH_A + BH_B$$

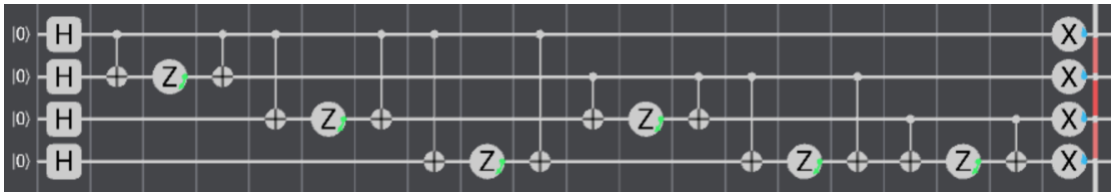
Exercise 10.4.5 If A=1 and B=1 (this choice works for our example), then write out the total Hamiltonian:

Evaluate the energy of all 16 computational basis states, and check that the ground state (lowest energy states) corresponds to a partitioning that does indeed match with your answer for 10.4.2 (again, exel might help). Don't forget to use Z eigenvalues.

Basis state, i	$E[H_A]$	$E[H_B]$	Basis state energy	Basis state, i	$E[H_A]$	$E[H_B]$	Basis state energy
$\|0000\rangle$	16	0	16	$\|1000\rangle$			
$\|0001\rangle$	4	2	6	$\|1001\rangle$			
$\|0010\rangle$				$\|1010\rangle$			
$\|0011\rangle$				$\|1011\rangle$			
$\|0100\rangle$				$\|1100\rangle$			
$\|0101\rangle$				$\|1101\rangle$			
$\|0110\rangle$				$\|1110\rangle$			
$\|0111\rangle$				$\|1111\rangle$	16	0	16

Now we will construct a trial wavefunction, using a method adapted from QAOA, in order to find the ground state(s) of this Hamiltonian, and hence solve the problem.

For this we will need to construct a circuit which initially has a set of Hadamard gates, then a series of phase operations, and finally a series of X-rotations, similar to that shown below (this circuit for H_A alone):



Exercise 10.4.6 First we will just attempt to find the ground-state of H_A alone. Note here the coefficients of the ZZ operators are all the same. Making each z-axis rotation angle of

$$\alpha = 0.3\pi$$

and final x-rotations by an angle of

$$\beta = \theta_X = 0.3\pi$$

construct the circuit above using the QUI. **NB. In the QUI you can highlight many R-gates at once and edit the rotation angles in one go (particularly useful when changing these angles later).**

a) Which states are most likely to be measured?

b) What is the probability that we measure a state with an equal number of vertices in each subset?

c) Optional. By systematically varying the angle of z-rotations, and x-rotations try to find α and β which give a higher probability of being in a state which has equal number of vertices in each subset (a balanced state).

α	β	Probability of being in any balanced state.

Now, let us attempt to find a solution to the problem by adding in the edge Hamiltonian H_B. We need to modify the circuit to accommodate the fact that the coefficients of ZZ terms are not all the same.

Exercise 10.4.7 Noting the coefficients in the Hamiltonian E_{ZZ}^{ij} modify the QUI circuit to the required rotations using

$$\alpha = 0.2\pi$$
$$\beta = 0.3\pi$$

a) Which two states are most likely to be measured?

_____ and _____

b) What is the probability that we measure a state corresponding to the correct solution?

c) What is the expectation value of the energy for this trial wavefunction?

d) Optional. By varying the angle of the z-rotations, and x-rotations through α and β (keeping each angle proportional to the corresponding term in the Hamiltonian) try to reduce the expectation value of the energy, and record the probability of a correct solution:

α	β	Energy	Probability

Exercise 10.4.8 If you got this far (well done!), try another graph partitioning problem with more vertices.

MULT90063 Introduction to Quantum Computing

Lab Session 12

12.1 Introduction

Welcome to Lab 12 of MULT90063 Introduction to Quantum Computing. The purpose of this lab session is to get some experience with the HHL (Harrow, Hassidim, Lloyd) algorithm for solving linear equations.

In this lab we will set up one instance of solving a set of linear equations. First we will solve the system of equations using classical techniques (i.e. you solve them), and then we will implement the quantum circuit for solving this set of equations in the QUI. You may to do your algebraic calculations on a separate piece of paper, but where indicated write you answers here so we can easily check your progress.

12.2 Solving the set of linear equations classically

The set of linear equations which we will be solving in this lab is:

$$\begin{aligned} +15w + 9x + 5y - 3z &= 4 \\ +9w + 15x + 3y - 5z &= 4 \\ +5w + 3x + 15y - 9z &= 4 \\ -3w - 5x - 9y + 15z &= 4 \end{aligned}$$

Exercise 12.2.1 Write this set of equations as a matrix equation, $A\vec{x} = \vec{b}$ where $\vec{x} = (w, x, y, z)^T$:

Exercise 12.2.2 Using any method of your choosing, solve the equations for w, x, y, z and write the solutions below:

Exercise 12.2.3 Find the eigenvalues, λ_i, and corresponding eigenvectors, $\vec{u}_i$ of A. Please normalize your eigenvectors so that the have a norm of 1, and make a valid quantum state. Write them out below:

Exercise 12.2.4 From the eigenvalues, λ_i, and eigenvectors, $\vec{u}_i$, find A^{-1}:

$$A^{-1} = \sum_i \frac{1}{\lambda_i} |u_i\rangle\langle u_i| = \sum_i \frac{1}{\lambda_i} \vec{u}_i (\vec{u}_i^*)^T$$

Exercise 12.2.5 Express b as a linear combination of the eigenvectors $\vec{u}_i$ of A, i.e. finding amplitudes β_i in the decomposition of the vector $\vec{b}$ in terms of the vectors $\vec{u}_i$:

$$\vec{b} = \sum_i \beta_i \vec{u}_i$$

Exercise 12.2.6 Solve the system of linear equations by equating $\vec{x} = A^{-1}\vec{b}$.

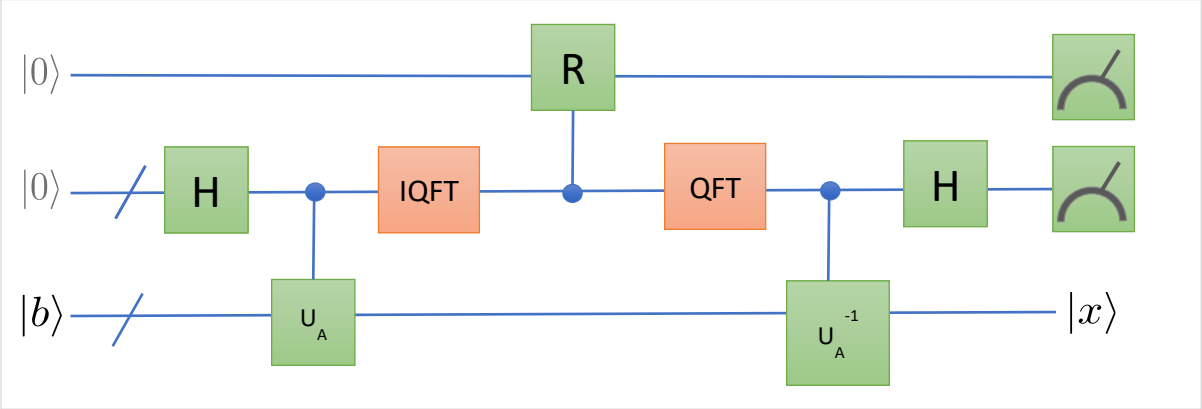
Exercise 12.2.7 Check that this answer can also be obtained by equating:

$$\vec{x} = \sum_i \frac{\beta_i}{\lambda_i} \vec{u}_i$$

12.3 Solving the system using HHL in the QUI

We will now program QUI solve the same set of linear equations in the HHL approach. You will need a total of 7 qubits, in order to solve the system of equations using HHL.

Recall from the lectures the basic HHL circuit is:



Here U represents a matrix of the form:

$$\exp(iAt)$$

The first qubit is an ancilla initialised to $|0\rangle$. The next 4 qubits form the "upper register" or "control register" which are initialised in an equal superposition. The "bottom register" of 2 qubits are initialised in the state corresponding to $\vec{b}/|\vec{b}| = (\hat{b}_0, \hat{b}_1, \hat{b}_2, \hat{b}_3)^T = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^T$.

We now make the correspondence between this (nicely balanced) column 4-vector and the associated column 4-vector of *two* qubits (in a separable state in this instance)

$$|b\rangle = |q_1\rangle \otimes |q_2\rangle$$

where $|q_1\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|q_2\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. Make sure you can see how $|b\rangle$ represents $\vec{b}/|\vec{b}|$.

Exercise 12.3.1 Program the first step of the algorithm into the QUI, initializing the control/upper register and the lower register.

In the second step of the algorithm, we need to implement controlled rotations of the form,

$$\exp(iAt)$$

To see how to do this, we need to write out the matrix A as a linear combination of Pauli matrices.

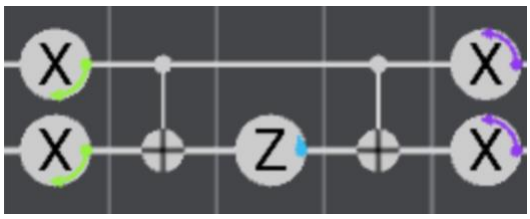
$$A = \begin{pmatrix} 15 & 9 & 5 & -3 \\ 9 & 15 & 3 & -5 \\ 5 & 3 & 15 & -9 \\ -3 & -5 & -9 & 15 \end{pmatrix} = 15\, I\otimes I + 9Z\otimes X + 5X\otimes Z + 3Y\otimes Y$$

Note: Each term commutes with the other two (i.e. all inter-commute).

Exercise 12.3.2 Because each of these terms commute, we can consider each separately. i.e.

$$\begin{aligned}\exp(iAt) &= \exp(i[15\, I\otimes I + 9Z\otimes X + 5X\otimes Z + 3Y\otimes Y]t) \\ &= \exp(i[15\, I\otimes I]t)\exp(i[3Y\otimes Y]t)\exp(i[5Z\otimes X]t)\exp(i[5X\otimes Z]t)\end{aligned}$$

Verify, as far as you can using the QUI, that the following circuit makes a rotation by $\exp(i\theta Y\otimes Y)$:



Hint: refer back to Lecture 19 for the decomposition of exponentials of such operators, e.g.

$$\exp(i\theta Y \otimes Y) = \cos(\theta)\, I \otimes I + i \sin(\theta) Y \otimes Y$$

Use this to write out the effect of this sequence of gates on two qubit states (such as $|00\rangle$) which you can check in the QUI directly. Use a rotation angle of -0.1875π (and no global phase) in the z-rotation.

In the same manner, verify that this circuit (depending on what angles you put in for the X rotations) makes an operation, $\exp(i\theta Z \otimes X)$:

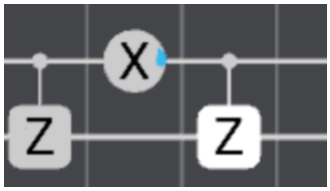
$$\exp(i\theta Z \otimes X) = \cos(\theta)\, I \otimes I + i \sin(\theta) Z \otimes X$$

Use an x-rotation angle of -0.3125π, and no global phase.

And that similarly that the following circuit makes an operation, $\exp(i\theta X \otimes Z)$:

$$\exp(i\theta X \otimes Z) = \cos(\theta)\, I \otimes I + i \sin(\theta) X \otimes Z$$

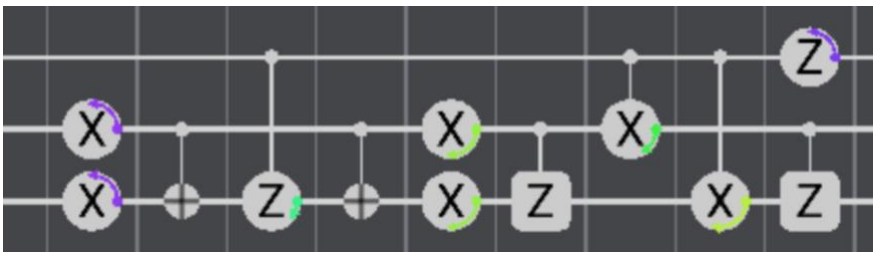
In this case, make the rotation angle -0.5625π, and no global phase.



Exercise 12.3.3 Implement the following circuit, combining all three rotations, which therefore applies a gate exp (iAt)

Exercise 12.3.4 Implementing a controlled version of the circuit for exp $(iA\theta)$.

We now would like to implement a controlled version of this gate. One way to do this is to control every single gate from the control register (lowest qubit of the "upper register"), however this is not required. A simpler way is to only control the two X, and one Z rotations. One extra z-rotation is required, corresponding to the identity term in the matrix, A. This term leads to a z-rotation on the *control* qubit.



Show (using the QUI or otherwise) that this circuit is equivalent to controlling every rotation.

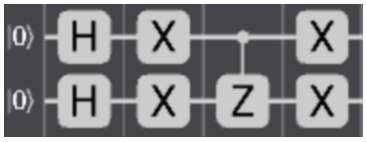
Finally, implement a the z-rotation on the control qubit of with an angle of 0.46875π (global phase of 0.234375π). This rotation comes from the identity term in A.

Exercise 12.3.5 Verifying your circuit using an eigenstate.

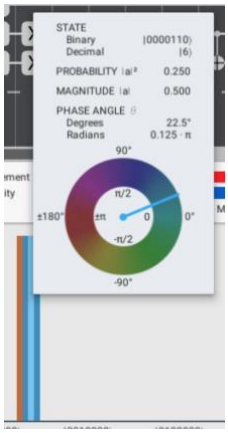
To verify that you have coded your circuit correctly, we will apply the circuit to an eigenstate of A. One such eigenstate is:

$$|u_1\rangle = \frac{1}{2}\begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

To verify, prepare this eigenstate (instead of b) using the circuit :



And verify that if the control qubit is initialized in this state, after the application of the circuit in 12.3.4 (and the control qubit set to 1) that the eigenstate receives a phase of $\frac{\pi}{8}$.

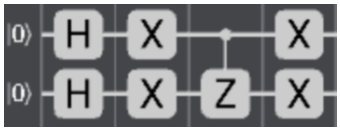


Exercise 12.3.6

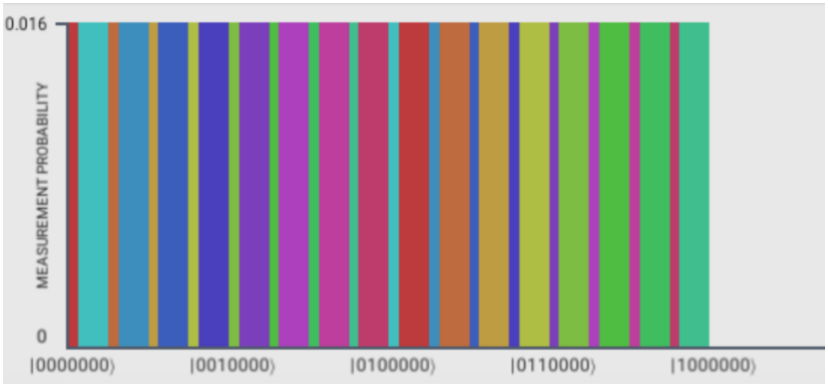
Extend your circuit, repeating the sequence for each of the four control qubits in the "upper register", remembering to double the angle of each the rotation gates with each round/control qubit. Each round should be controlled from a different qubit in the control register. In terms of the original angle, the angles should be 1, 2, 4, and 8 times as large as the angles given in 12.3.4.

Exercise 12.3.7 Verify your circuit by applying to an eigenstate.

To verify your circuit, prepare the lower register to eigenstate of A in a similar way to 12.3.5, and Hadamard gates to the control/upper register.



Check that your circuit gives phases as shown below, with a phase proportional to the state, t, in the upper/control register:

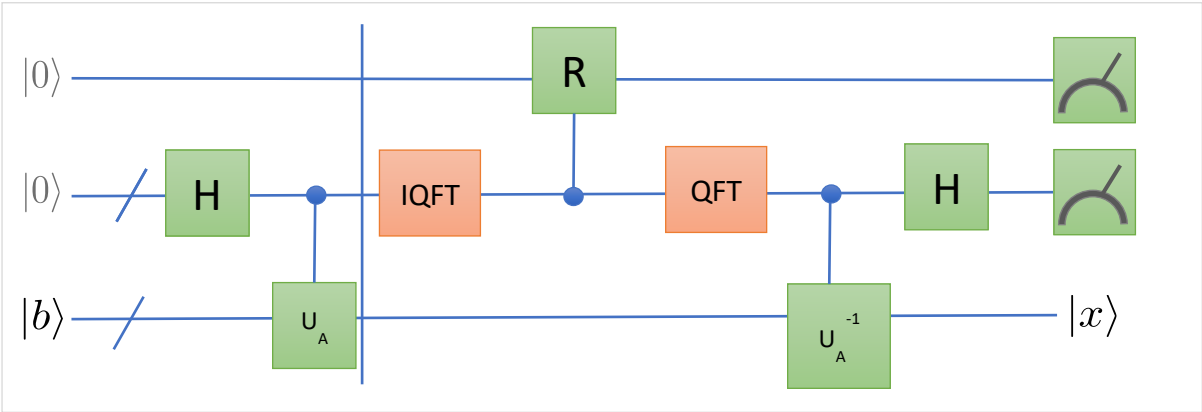


You should be able to verify that the phase is reported by QUI is

$$\frac{\pi}{8}t$$

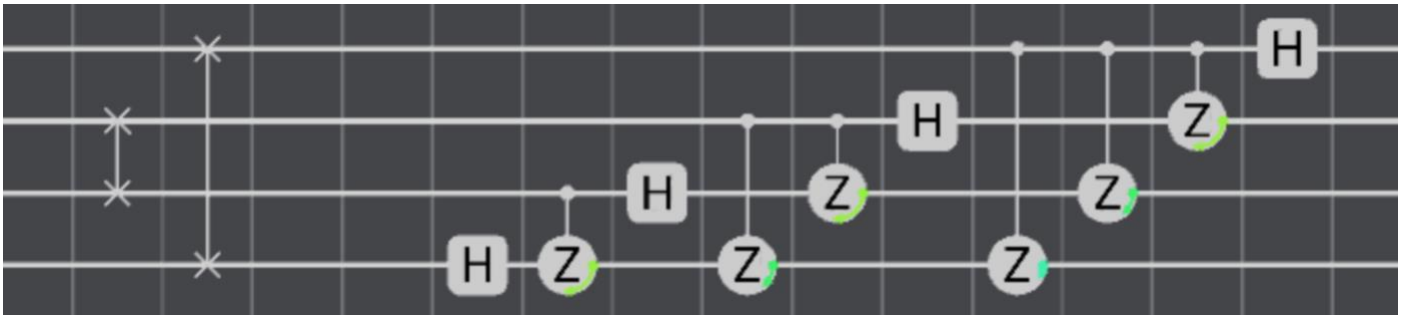
where t is the state of the control/upper register.

After this step has been implemented correctly, you have finished the exponentiation step of the algorithm:



Exercise 12.3.8 Finding the eigenvalues.

To find the eigenvalues, we need to implement a IQFT, pictured below. This IQFT is the same as those you have implemented previously.



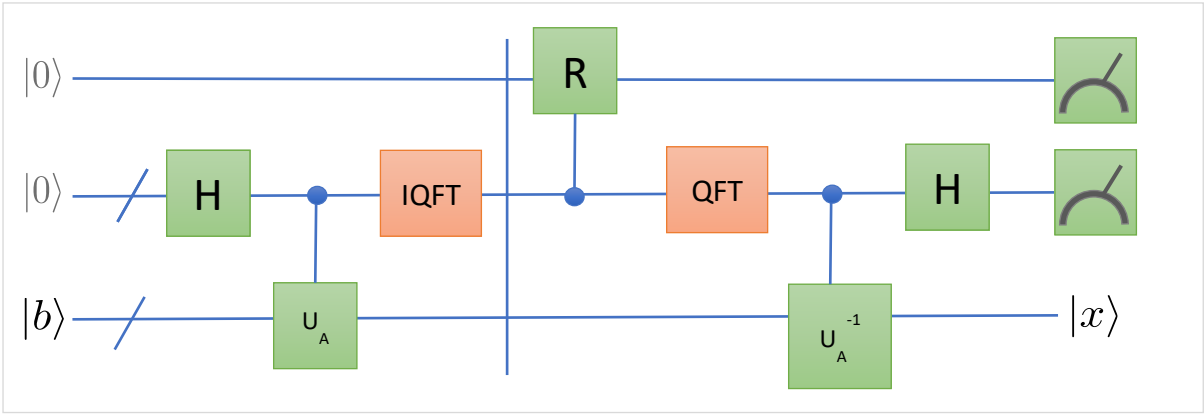
Go ahead and implement an IQFT on the control register.

Exercise 12.3.9 Check the eigenvalues.

After the application of the IQFT, the control/upper register should contain the eigenvalue corresponding to the input state (or a superposition of these values). For the eigenstate we have used in previous exercises, check that the control register is in the state, $|1\rangle$.

Prepare all four eigenstates as inputs to the circuit and check that after the IQFT that the control register contains the corresponding eigenvalue: 1, 2, 4 or 8.

After this step, you have implemented the circuit up to the end of the IQFT:



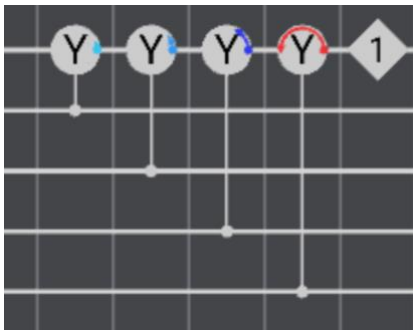
1.2.10 Invert the Eigenvalues

The next part of the circuit inverts the eigenvalues. In this operation, we will slightly cheat, and assume that we know they will be one of these four values: 1, 2, 4 or 8. We wish to implement operations, so that after applied to an initial $|0\rangle$ state, the amplitudes of the $|1\rangle$ state is, 1, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ respectively.

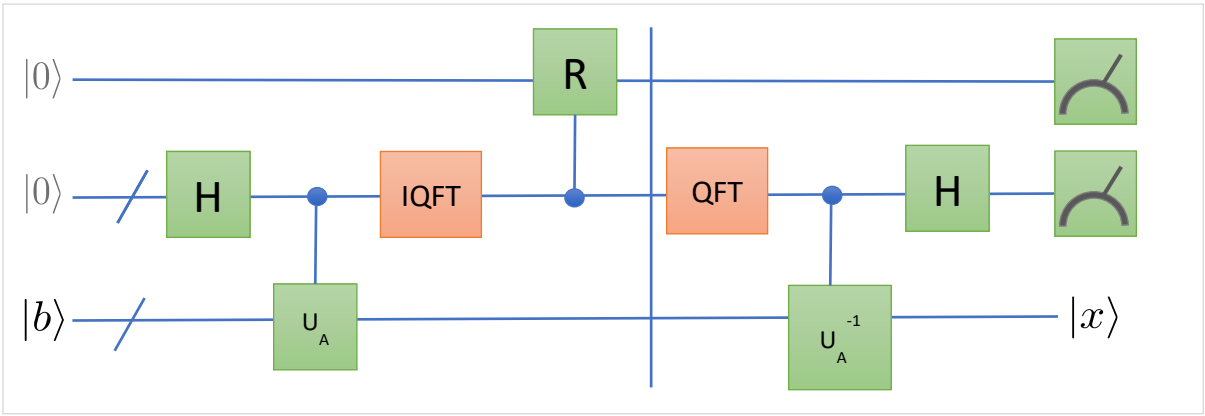
Applying to the initial $|0\rangle$ state, write out your angles which angle y-rotation gives each of these amplitudes:

Amplitude	Angle of y-rotation
$\frac{1}{1}$	π
$\frac{1}{2}$	
$\frac{1}{4}$	
$\frac{1}{8}$	

Implement these y-rotations in the QUI, controlled from the appropriate register in the control/upper register



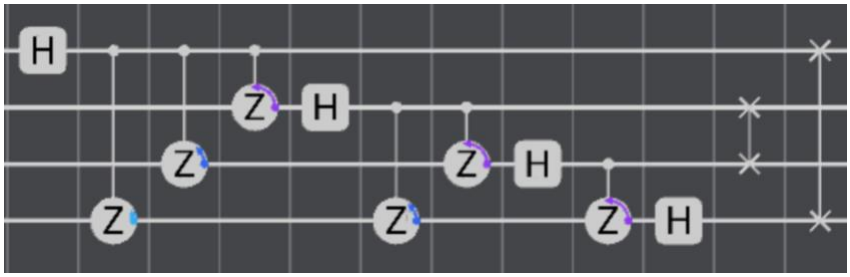
Finally, place a measurement after these rotations. On each run of the algorithm, we will only accept the result if this measurement is a "1". The collapses the state so the resulting amplitude is multiplied by $\frac{1}{\lambda}$.



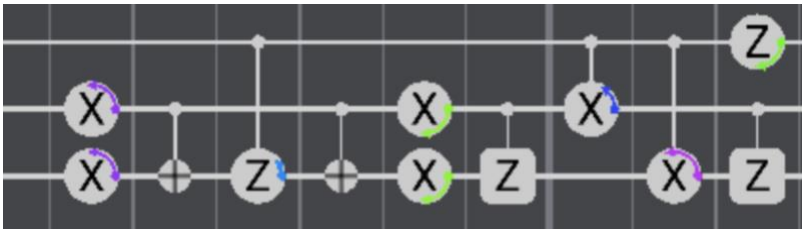
12.2.11 Uncalculate the Eigenvalues

We now need to "uncalculate" the IQFT and exponentiation. Do this by applying the inverse of each gate: reverse the sequence of gates, and change the sign of each rotation.

For the IQFT, this should be similar to the following circuit:



And a similar sequence for of the four exponentiation steps:



Finally, invert the initial Hadamard gates, as well as resetting the ancilla qubit from “1” to “0” using an x-gate:

After the application of these gates, provided a “1” is measured in the ancilla qubit, all of the control register should be in the “0” state.

12.2.12 Read the answer

Repeatedly run the algorithm until the measurement of the top register (the ancilla) is “1”. When it is, read the amplitudes of the resulting four states of the lower register. Are they proportional to, x, the solution to the linear equations?

If so, congratulations, you have demonstrated a version of HHL, one of the more difficult quantum algorithms, running in the QUI!

If not, debug your circuit! One way to do this is to verify that each of the four eigenstates are unchanged by the algorithm.

Your output should look similar to the amplitudes shown above.