PHYC90045 Introduction to Quantum Computing

Lab Session 4

4.1 Introduction

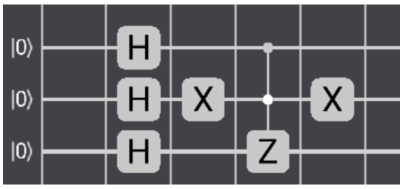
Welcome to Lab-4 of PHYC90045 Introduction to Quantum Computing.

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

4.2 Marking states with an oracle

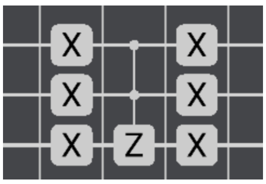
Exercise 4.2.1 a) Consider the oracle function below. Verify that it marks one state in the equal superposition. What is the corresponding binary number marked?



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).

4.3 Inversion

Exercise 4.3.1 a) Show that the following (3-bit) circuit implements the "inversion" operation: $I - 2|000\rangle\langle 000|$.

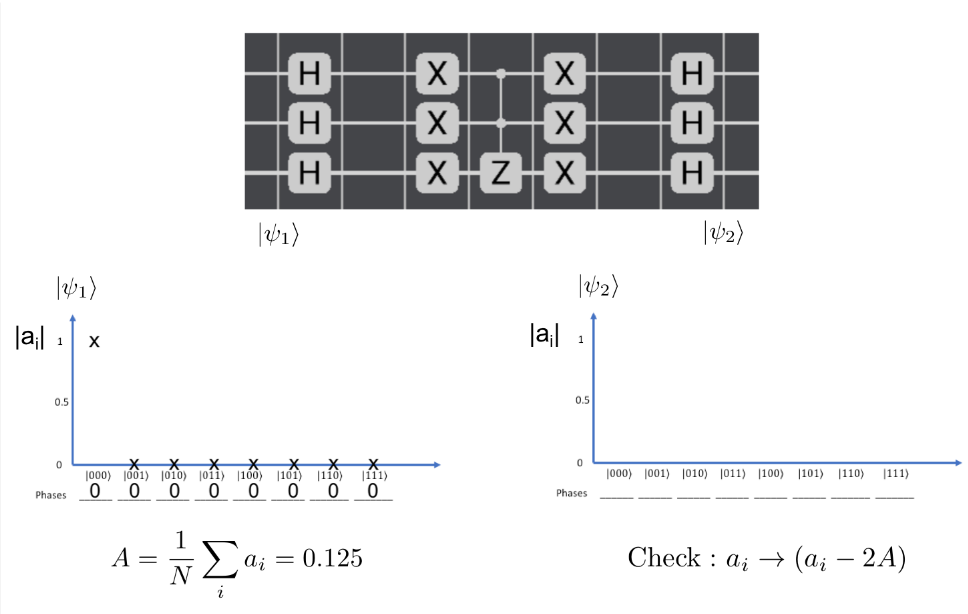


b) Generalise to the 5 bit case and test.

4.4 Inversion about the mean

Exercise 4.4.1 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.

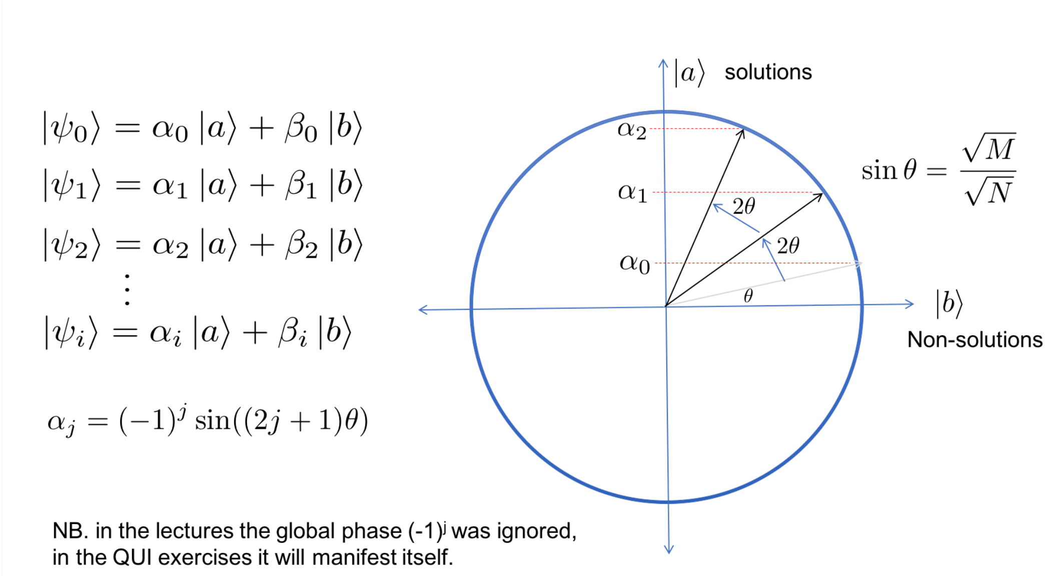


4.5 Grover search - single solution case

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

Exercise 4.5.1 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 "Lab4 4.5.1 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).



NB. In this geometric picture we have effectively applied an overall global phase factor of $(-1)^j$ between iterations to make all the angles positive. When program the circuit in the QUI we will see this manifest itself so we have re-inserted into the figure above.

Exercise 4.5.2 For the example in 4.5.1 (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		
	$(-1)^j \alpha_j$	$\lvert\alpha_j\rvert^2$	$(2j+1)\theta$ (rad)	$\alpha_j = (-1)^j \sin((2j+1)\theta)$	$\lvert\alpha_j\rvert^2 = [\sin((2j+1)\theta)]^2$
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 4.5.3 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 "Lab4 4.5.3 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, j	Compute in QUI		Geometric Picture		
	$(-1)^j \alpha_j$	$\lvert\alpha_j\rvert^2$	$(2j+1)\theta$ (rad)	$\alpha_j = (-1)^j \sin((2j+1)\theta)$	$\lvert\alpha_j\rvert^2 = [\sin((2j+1)\theta)]^2$
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.

4.6 Grover search - multiple solution case

Generalise to multiple solutions.

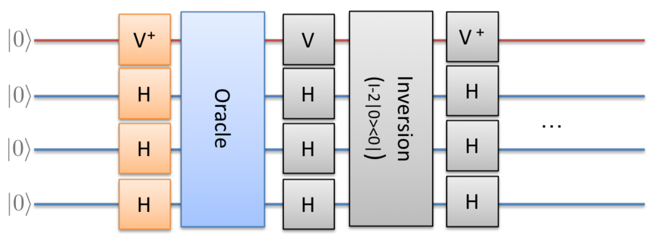
Exercise 4.6.1 Load the circuit "Lab4 4.5.3 Grover 5-qubits oracle 12" (i.e. one iteration). Now modify the oracle (grab multiple gates, copy, move, paste etc) to also mark the state corresponding to the number 3 (00011). Move the scrubber to the output of the oracle and verify the state marking. Run the circuit and fill in the table below.

Iteration, j	Compute in QUI	Geometric Picture		
	Probability measure |3⟩ or |12⟩	$(2j+1)\theta$ (rad)	$\alpha_j = (-1)^j \sin((2j+1)\theta)$	$\|\alpha_j\|^2 = [\sin((2j+1)\theta)]^2$
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.

4.7 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 4.7.1 Go back to the 3-qubit case (4.5.1) and load "Lab4 4.5.1 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 = $R_X(\theta_R)$ 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: $\sin\theta = \frac{\sqrt{M}}{\sqrt{N}} = g_0$

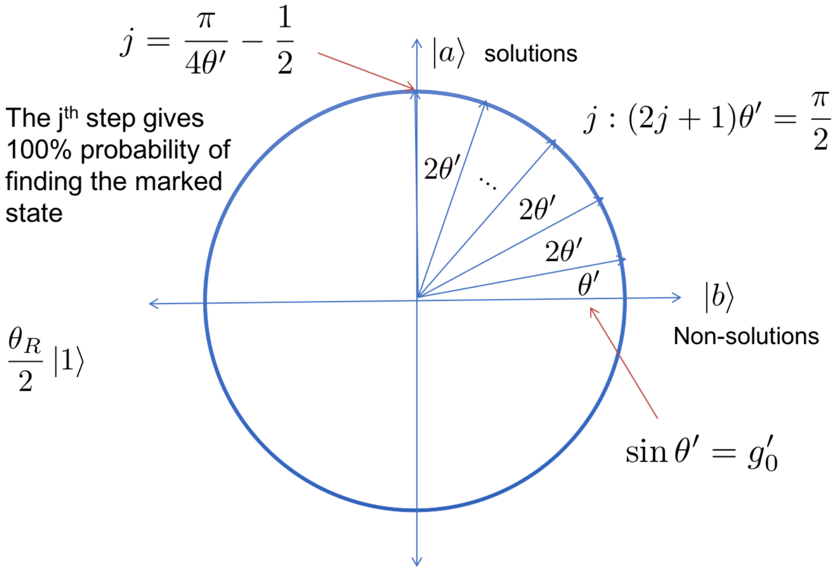
Amplitude Amplification operator, V:

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

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

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

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



Exercise 4.7.2 Now we will determine the X-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. Note, for convenience we can set the global phase in the X-rotation to make g_0' real (What is the global phase required?). 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 X-rotation angle θ_R (for the case specified). Find the θ_R that gives the least number of iterations to reach 100% search result.

n	N	M	$g_0 = \sqrt{\frac{M}{N}}$	θ_R (units of π)	g_0'	$\theta' = \sin^{-1} g_0'$ (rad)	$j = \frac{\pi}{4\theta'} - \frac{1}{2}$
3	8	1	0.3536	0.5	0.25	0.2527	2.6082

Exercise 4.7.3 Program the determined X-rotation angle into the R-gates in the QUI, run the program, and verify it's working as expected. Save the circuit as "Lab4 4.7.3 AA V=RX 3-qubits oracle 5".

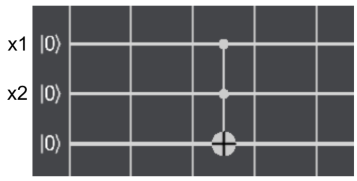
Exercise 4.7.4 Modify the circuit to a search for two numbers (M=2) over a 7-bit register again with V set to a rotation about the X-axis. Determine the required rotation angle in V to produce a minimal number of iterations to reach the ideal 100% probability of finding one (unspecified) of the target numbers. Use the table below as a guide.

n	N	M	$g_0 = \sqrt{\frac{M}{N}}$	θ_R (units of π)	g_0'	$\theta' = \sin^{-1} g_0'$ (rad)	$j = \frac{\pi}{4\theta'} - \frac{1}{2}$
7	128	2	0.1250	0.5	0.0883	0.2527	8.3741

Exercise 4.7.5 Exploration: try a higher number of solutions, and/or other operators for V, and/or different oracles/search problems.

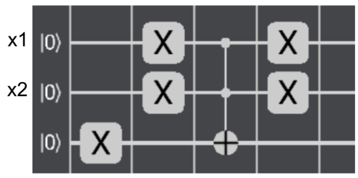
4.8 More adventurous oracles

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/).



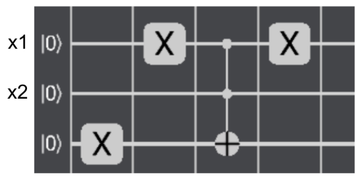
x1/\x2

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



x1\/x2

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



x1\/~x2

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

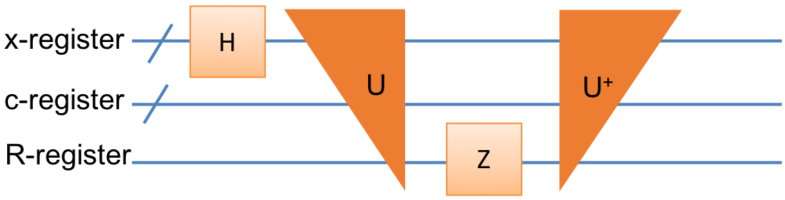
Exercise 4.8.1 Program these into the QUI and verify the truth tables (you can do all instances in one go using Hadamards to generate an equal superposition of inputs).

Exercise 4.8.2 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 4.8.3 If the circuit that evaluates the logical expression in 4.8.2 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 "Lab4 4.8.3 SAT oracle".

Exercise 4.8.4 Now add inversion about the mean on the x-register to complete one iteration of the Grover search. Save as "Lab4 4.8.3 SAT Grover iteration". Run the circuit and increase the number of Grover iterations until the solution is found.