PHYC90045 Introduction to Quantum Computing

Lab Session 6

6.1 Introduction

Welcome to Lab 6 of PHYC90045 Introduction to Quantum Computing.

The purpose of this lab session is to:

- understand 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 $|a\rangle$

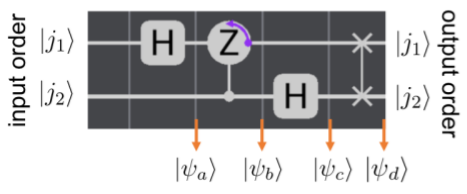
$$\text{QFT}\,|a\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} ka} |k\rangle$$

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).



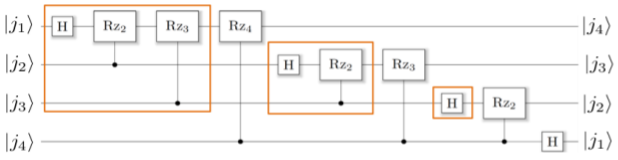
$$|j_1, \ldots, j_n\rangle \rightarrow \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle}{\sqrt{2}}}_{|j_1\rangle} \otimes \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n} |1\rangle}{\sqrt{2}}}_{|j_2\rangle} \otimes \ldots \otimes \underbrace{\frac{|0\rangle + e^{2\pi i\, 0.j_1 j_2 \ldots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

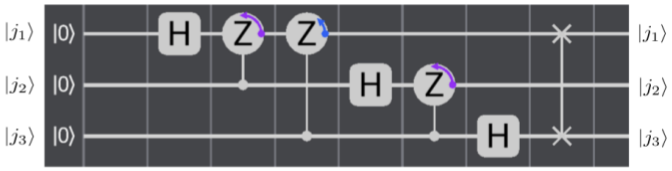
$$R_{z_2} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix} \equiv R_Z\left(\frac{\pi}{2}\right) \qquad \text{with} \quad \theta_g = \frac{\pi}{4}$$

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:

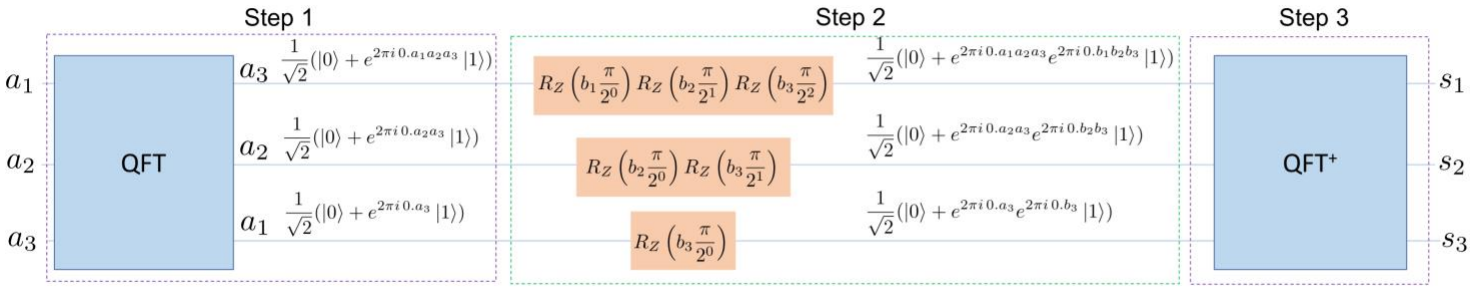
$$a = a_1 2^{n-1} + a_2 2^{n-2} + ... + a_{n-2} 2 + a_n$$
$$b = b_1 2^{n-1} + b_2 2^{n-2} + ... + b_{n-2} 2 + b_n$$

(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.

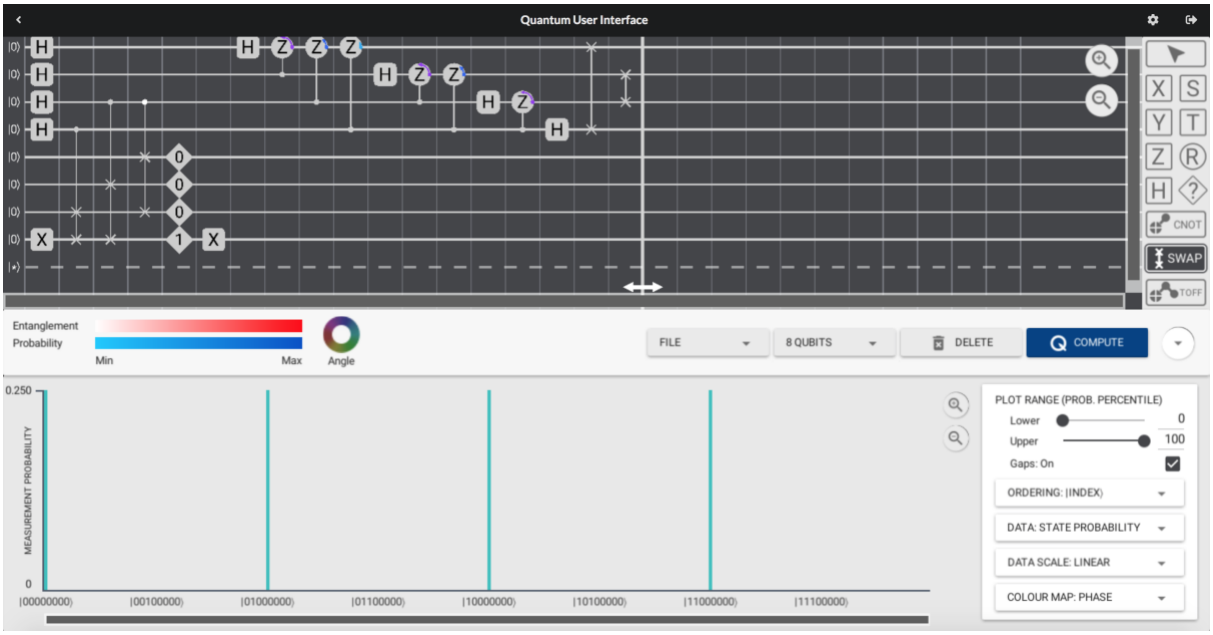
Exercise 6.4.2 Add 3 more qubits to represent the b register. Add controls to the central phase gates and show the circuit performs the addition a + b (modulo $N=2^n$). Try it with various superpositions of inputs.

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:

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

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).