1 Readings

Benenti et al., Ch. 3.11

Stolze and Suter, Quantum Computing, Ch. 8.3.4

Nielsen and Chuang, Quantum Computation and Quantum Information, Ch. 5.1

2 Quantum Fourier Transform (QFT): all about phase

The Quantum Fourier Transform (QFT) implements the analog of the classical Fourier Transform. It transforms a state space of size 2^n from the amplitude to the frequency domain (just as the Fourier transform can be viewed as a transform from 2^n numbers into a range of size 2^n containing the frequency components from the amplitude domain.

The classical Fourier Transform is defined as:

$$y_k \equiv \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} x_j e^{2\pi ijk/2^n}$$

The QFT is similarly defined:

$$|j\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n} |k\rangle$$

Thus an arbitrary quantum state is transformed:

$$\sum_{j=0}^{2^n-1} x_j|j\rangle \longrightarrow \sum_{k=0}^{2^n-1} y_k|k\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} x_j e^{2\pi ijk/2^n} |k\rangle$$

Example:

$$|00000\rangle + |01000\rangle + |10000\rangle + |11000\rangle$$

is transformed to:

$$\begin{aligned} &|00000\rangle + |00100\rangle + |01000\rangle + |011000\rangle \\ +&|10000\rangle + |10100\rangle + |11000\rangle + |11100\rangle \end{aligned}$$

i.e.: 0 8 16 24

is transformed to:

0 4 8 12 16 20 24 28

So how do we implement the QFT? This derivation is in Nielsen and Chuang at pages 216-219, but is expanded in parts here for clarity.

We are going to work with the transform of a single quantum state, defined as:

$$\big|j\big\rangle \longrightarrow \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n}\big|k\big\rangle \qquad (1)$$

Note that j is a binary number and can be decomposed into the form:

$$j = j_1 2^{n-1} + j_2 2^{n-2} + \ldots + j_n 2^0 = \sum_{i=1}^{n} j_i 2^{n-i}$$

Similarly for k

$$k = \sum_{i=1}^{n} k_i 2^{n-i}$$

Use the k decomposition and leave j alone for now, to re-express the transform as

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i j \sum_{l=1}^{n} k_l 2^{n-l}/2^n}\big|k\big\rangle$$

Canceling the 2^n terms we have:

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i j \sum_{l=1}^{n} k_l 2^{-l}}\big|k\big\rangle$$

Now write the exponent out explicitly:

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i j k_1 2^{-1}} \times e^{2\pi i j k_2 2^{-2}} \times \ldots \times e^{2\pi i j k_n 2^{-n}}\big|k\big\rangle$$

Now, decompose the summation over k as a sum over the two allowed binary values 0 and 1 of each bit k_i:

$$\frac{1}{\sqrt{2^n}}\sum_{k_1=0}^{1}\sum_{k_2=0}^{1}\cdots\sum_{k_n=0}^{1} e^{2\pi i j k_1 2^{-1}} \times e^{2\pi i j k_2 2^{-2}} \times \ldots \times e^{2\pi i j k_n 2^{-n}}\big|k_1 k_2 \ldots k_n\big\rangle$$

Now, pull out the n'th component:

$$\frac{1}{\sqrt{2^n}}\sum_{k_1=0}^{1}\sum_{k_2=0}^{1}\cdots\sum_{k_{n-1}=0}^{1} e^{2\pi i j k_1 2^{-1}} \times e^{2\pi i j k_2 2^{-2}} \times \ldots \times e^{2\pi i j k_n 2^{-n}}\big|k_1 k_2 \ldots k_{n-1}\big\rangle \sum_{k_n=0}^{1} e^{2\pi i j k_n 2^{-n}}\big|k\big\rangle$$

This last factor for the n'th component is equal to:

$$\frac{1}{\sqrt{2^n}}\Big(\big|0\big\rangle + e^{2\pi i j 2^{-n}}\big|1\big\rangle\Big)$$

where the first component comes from the $k_n = 0$ term and the second component from the $k_n = 1$ term. Repeating this for all k_i components leads to:

$$\frac{1}{\sqrt{2^n}}\Big(\big|0\big\rangle + e^{2\pi i j 2^{-1}}\big|1\big\rangle\Big)\Big(\big|0\big\rangle + e^{2\pi i j 2^{-2}}\big|1\big\rangle\Big)\ldots\Big(\big|0\big\rangle + e^{2\pi i j 2^{-n}}\big|1\big\rangle\Big)$$

So now we have a tensor product of qubit states each of which contains a different phase factor, $e^{2\pi i\left(\frac{j}{2^k}\right)}$, where $1 \le k \le n$. So if we can systematically generate these phase factors with quantum gates, we have a means of implementing the QFT. We will now put them in a form in which this generation and the resulting quantum circuit is easy to see.

First we define a new binary notation for a fraction - this corresponds to the analog of a decimal in base 10. For a number lying between 0 and 1, the binary fraction is simply the expansion in powers of $1/2$, which is written in the 'decimal' form as:

$$0.j_l j_{l+1}\ldots j_m = \frac{j_l}{2} + \frac{j_{l+1}}{2^2} + \frac{j_m}{2^{m-l+1}}$$

where each $j_i = 0$ or 1.

Now since $k \leq n$, the quantity $\frac{j}{2^k}$ is clearly a number greater than or equal to one, but it is not necessarily an integer. We can use our binary fraction notation to write it as a 'rational binary' number:

$$\begin{aligned} \frac{j}{2^k} &= \sum_{\nu}^{n} j_\nu 2^{n-\nu-k} \\ &= j_1 j_2 \ldots j_{n-k}.j_{n-k+1} \ldots j_n \end{aligned}$$

For example, if $n = 8$ and $k = 3$, we have

$$\begin{aligned} j &= j_1 2^7 + j_2 2^6 + j_3 2^5 + j_4 2^4 + j_5 2^3 + j_6 2^2 + j_7 2^1 + j_8 2^0 \\ \text{and } \frac{j}{2^3} &= j_1 2^4 + j_2 2^3 + j_3 2^2 + j_4 2^1 + j_5 2^0 + j_6 2^{-1} + j_7 2^{-2} + j_8 2^{-3}. \end{aligned}$$

From this it is clear that the last three terms are the binary fraction $0.j_6 j_7 j_8$, while the first five terms constitute an integer.

Now coming back to the phase factor $e^{2\pi i\left(\frac{j}{2^k}\right)}$, we now see that the integer part of $\frac{j}{2^k}$ will merely contribute a factor of 1 and that the phase is therefore entirely determined by the binary fraction:

$$e^{2\pi i\left(\frac{j}{2^k}\right)} = 1 \cdot e^{0.j_{n-k+1} \cdots j_n}$$

We can now apply this to every term in the transform, to rewrite it as

$$\frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \ldots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right) \qquad (2)$$

To see how to actually implement this with quantum gates, lets look at any one of the qubits and how it should be transformed:

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi 0.j_l \cdots j_n}|1\rangle\right)$$

Pull off the first component:

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_l} \times e^{2\pi 0.0 j_{l-1} \cdots j_n/2}|1\rangle\right)$$

Looking at the first component only, i.e., qubit 1:

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_l}|1\rangle\right) = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i j_l/2}|1\rangle\right) = \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^{j_l}|1\rangle\right)$$

since $0.j_l = j_l/2$ and using $e^{i\pi j_l} = (-1)^{j_l}$ where $j_l = 0, 1$. This is just an H gate!

What about $e^{2\pi 0.0 j_{l-1} \cdots j_n/2}$? For this we can use a sequence of rotations of the form

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

that are controlled by the value of the j_k'th qubit. Thus we will apply this rotation conditionally to qubit 1, i.e., if j_k is equal to 1, we apply R_k, while if $j_k = 0$, we do nothing. We implement this sequence of controlled rotations starting with the least significant digit first, i.e., j_{l-1} in the above example.

Lets go through the entire procedure now. We want to achieve

$$\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right).$$

Start with $|j_1\rangle |j_2 \ldots j_n\rangle$.

Apply H on qubit 1 to obtain

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right) |j_2 \ldots j_n\rangle$$

Apply a controlled R_2 rotation on qubit 1, with qubit 2 the control, to obtain

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i 0.j_1 j_2}\left|1\right\rangle\right)\left|j_2 \dots j_n\right\rangle$$

Apply controlled R_3 on qubit 1, with qubit 3 the control, to obtain

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i 0.j_1 j_2 j_3}\left|1\right\rangle\right)\left|j_2 \dots j_n\right\rangle$$

Continue down to qubit n, to obtain

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}\left|1\right\rangle\right)\left|j_2 \dots j_n\right\rangle$$

This entire procedure is then repeated for the other qubits, i.e., j_2, then j_3, etc. etc., resulting in the final transformed state:

$$\frac{1}{\sqrt{2^n}}\left(\left|0\right\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}\left|1\right\rangle\right)\left(\left|0\right\rangle + e^{2\pi i 0.j_2 \dots j_n}\left|1\right\rangle\right)\dots\left(\left|0\right\rangle + e^{2\pi i 0.j_n}\left|1\right\rangle\right)$$

That's *almost* it: comparing this with Eq. (2), you may notice that the result has ended up with the bits in reverse order. This is not a problem, we can just swap them pairwise, starting from the ends and moving to the middle, using the SWAP circuit of lecture 8 (3 CNOT gates in alternating orientation). Then finally we have the QFT state of Eqs. (2) and (1):

$$\frac{1}{\sqrt{2^n}}\left(\left|0\right\rangle + e^{2\pi i 0.j_n}\left|1\right\rangle\right)\left(\left|0\right\rangle + e^{2\pi i 0.j_{n-1} j_n}\left|1\right\rangle\right)\dots\left(\left|0\right\rangle + e^{2\pi i 0.j_2 \dots j_n}\left|1\right\rangle\right)\left(\left|0\right\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}\left|1\right\rangle\right)$$

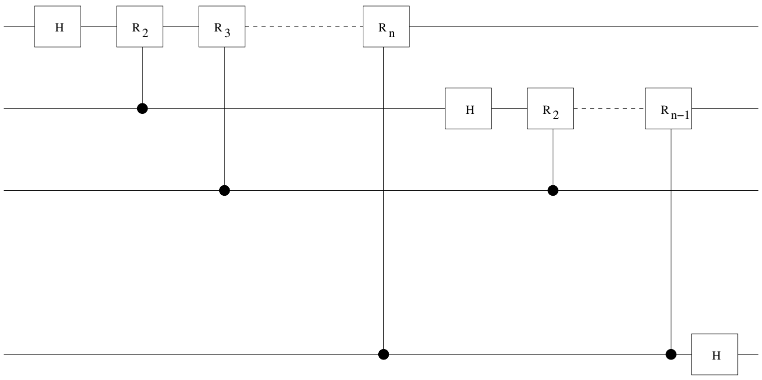

Figure 1: Quantum circuit implementing the Quantum Fourier Transform (QFT) on a quantum state input at the left. The first qubit is at the top, as usual. Note that the outputs are reversed in their bit-significance, i.e., qubit 1 contains the state of qubit n, etc. Following this circuit by a series of SWAP gates will then produce the final QFT state.

How many gates are required? Qubit 1 required H and $n-1$ controlled R gates, so a total of n gates. Qubit 2 required H and $n-2$ controlled R gates, so a total of $n-1$ gates. Continuing, we see that altogether $n+(n-1)+(n-2)\dots+1 = n(n+1)/2$ gates are required, plus the final series of SWAP gates. These are of order $n/2$ (depending whether n is even or odd), so that the overall scaling of the QFT is $O(n^2)$. So we have polynomial scaling of the number of gates with the number of input qubits - an efficient quantum algorithm!

How does this compare with classical Fourier Transforms? Well, the simple Fourier transform shown at the very beginning of the lecture can be written as a matrix times a vector, where the matrix is of size $N = 2^n$. Thus the direct classical Fourier Transform scales as $O((2^n)^2)$, which is clearly exponential. In physics the scaling is often written as N^2 but don't let that fool you - remember to ask how many bits n there are! There exists a more efficient classical algorithm, the fast fourier transform or FFT, which improves on this to give a scaling $O(NlnN)$. Clearly this is still exponential in n. So the QFT provides a truly significant quantum speedup.

Note that the QFT is unitary (since we could construct a unitary circuit for it). The classical transform is also unitary, as you can show by analyzing the FT matrix (see additional notes).