Chapter 5

QFT, Period Finding & Shor's Algorithm

5.1 Quantum Fourier Transform

Quantum Fourier Transform is a *quantum* implementation of the discreet Fourier transform. You might be familiar with the discreet Fourier Transform or Fourier Analysis from the context of signal processing, linear algebra, or one of its many other applications. In short, Fourier Analysis is a tool to describe the internal frequencies of a function.

Here we will present a quantum algorithm for computing the discreet Fourier transform which is exponentially faster than the famous Fast Fourier Transform of classical computers. However, this algorithm is an example of the tension between exponentially faster quantum algorithms and the problems of measurement. While we can carry out the QFT algorithm to transform the n qubit state vector $|\alpha\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \cdots + \alpha_n |n\rangle$ to its Fourier transform $|\beta\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle + \cdots + \beta_n |n\rangle$, a measurement on $|\beta\rangle$ will only return one of its n components, and we are not able to recover all the information of the Fourier transform. For this reason, we describe this algorithm as quantum Fourier *sampling*.

The Quantum Fourier Transform is a generalization of the Hadamard transform. It is very similar, with the exception that QFT introduces phase. The specific kinds of phases introduced are what we call primitive roots of unity, ω. Before defining the Fourier Transform, we will take a quick look at these primitive roots of unity.

Recall that in the complex numbers, there exist n solutions to the equation $z^n = 1$. For example if $n = 2$, z could be 1 or -1. If $n = 4$, z could be $1, i, -1,$

or $-i$. You can easily check that these roots can be written as powers of $\omega = e^{2\pi i/n}$. This number ω is called a primitive nth root of unity. In the figure below ω is drawn along with the other complex roots of unity for n=5.

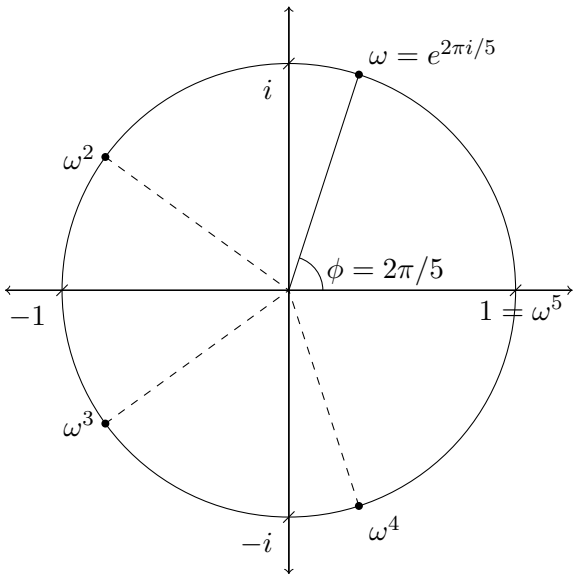


Figure 5.1: The 5 complex 5th roots of 1.

In this figure, we see that ω lies on the unit circle so $\mid \omega \mid = 1$, and the line from the origin to ω makes the angle $\phi = 2\pi/M$ with the real line. If we square ω, we double the angle. Furthermore, if we raise ω to the jth power, ω^j has phase angle $\phi = 2j\pi/M$ and is still an Mth root of unity.

Now, we can move in to the Fourier Transform itself. The discreet Fourier transform is defined by

$$QFT_M = \frac{1}{\sqrt{M}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{M-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2M-2} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3M-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2M-2} & \omega^{3M-3} & \cdots & \omega^{(M-1)(M-1)} \end{pmatrix}$$

Another way of writing this is to say that the jkth entry of QFT_M is ω^{jk}.

The transform takes the vector $\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{pmatrix}$ to its Fourier transform $\begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{pmatrix}$ as specified by the above matrix.

Examples

Ex. 1

Lets take a look at QFT_2. Because $M = 2$, $\omega = 2^{\pi i} = -1$ Therefore we have

$$QFT_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & \omega \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

As you can see, QFT_2 is simply equal to $H^{\otimes 2}$.

How about QFT_4? The primitive 4th root of unity is i, so that

$$QFT_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Ex. 2

Find the quantum Fourier transform for $M = 4$ of the functions $|f\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$; $|g\rangle = |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, and $|h\rangle = |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$.

The corresponding Fourier transforms are given by:

1. QFT_4 to $|f\rangle$.

$$\left|\hat{f}\right\rangle = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

2. QFT_4 on $|g\rangle$:

$$|\hat{g}\rangle = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

3. QFT_4 on $|h\rangle$:

$$\left|\hat{h}\right\rangle = \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\i\\-1\\-i\end{pmatrix}$$

Lets do a bit of analysis of these examples. In example 1, you might notice that the columns of QFT_4 are orthogonal. For example, the inner product of the first column with the second column is $\frac{1}{2}[(1*1)+(1*-i)+(1*-1)+(1*i)] = \frac{1}{2}(1-i-1+i) = 0$. You should also notice that, by design, the columns of QFT_4 have magnitude 1. Thus QFT_4 is unitary.

Another thing you should notice is that the vectors like $|f\rangle$ that had a lot of zeros (large spread) had Fourier transforms with few zeros (narrow spread), and vice-versa.

Finally, in examples 2 and 3, notice how the only difference between the Fourier transforms of $|g\rangle$ and $|h\rangle$ is a difference of relative phase shifts.

We would like to be able to make some statements to solidify these ideas about Fourier transforms, so lets prove them.

Properties of QFT

Studying the properties of a mathematical object gives us insight into the way it works. These properties will not only be important to our use of the Fourier transform later on, but they also provide a foundation of how to understand the discreet Fourier transform.

1. QFT_M is unitary.

   Proof It is well known that an operator is unitary if its columns are orthonormal. Denote the ith and jth columns of QFT_M as F_i and F_j. Then $F_i = \frac{1}{\sqrt{M}}\begin{pmatrix}1\\ \omega^{i*1}\\ \vdots\\ \omega^{i*(M-1)}\end{pmatrix}$ and $F_j = \frac{1}{\sqrt{M}}\begin{pmatrix}1\\ \omega^{i*j}\\ \vdots\\ \omega^{j*(M-1)}\end{pmatrix}$. Thus

$$\langle F_i|F_j\rangle = \frac{1}{M}\sum_{n=0}^{M-1}\omega^{ni}\overline{\omega^{nj}} = \frac{1}{M}\sum_{n=0}^{M-1}(\omega^{i-j})^n$$

   From here it is easy to see that if $i = j$, $\langle F_i|F_j\rangle = 1$.

For the case $i \neq j$, we will notice that $\frac{1}{M}\sum_{n=0}^{M-1}(\omega^{i-j})^n$ is a geometric series, and expand the sum. Thus

$$\frac{1}{M}\sum_{n=0}^{M-1}(\omega^{i-j})^n = \frac{1}{M}\frac{\omega^{M(i-j)}-1}{\omega^{i-j}-1} = \frac{1}{M}\frac{1-1}{\omega^{i-j}-1} = 0$$

where $\omega^{M(i-j)} = 1$ because ω is an Mth root of unity.

Because the Fourier transform is a unitary operator, we can implement it in a quantum circuit. Thus if $N = 2^n$, we can apply the Fourier transform QFT_N to a n-qubit system.

2. *Linear Shift.* This, property exemplified above, states that a linear shift of a state-vector causes a relative phase shift of its Fourier transform. This is expressed mathematically by saying if $|f(x)\rangle$, $x \in \mathbf{Z}_M$ has Fourier transform $\left|\hat{f}(x)\right\rangle$, then $|f(x+j)\rangle$ has Fourier transform $\left|\hat{f}(x)\right\rangle e^{\frac{2\pi}{M}xj}$. Furthermore, because QFT_M is unitary and $QFT_M QFT_M^\dagger = \mathbb{I}$, the converse is true. A linear phase shift on $|f\rangle$ produces a linear shift in $\left|\hat{f}\right\rangle$.

So if $QFT_N \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{N-1} \end{pmatrix}$, then $QFT_N \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_0 \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \omega\beta_1 \\ \vdots \\ \omega^{N-1}\beta_{N-1} \end{pmatrix}$

and $QFT_N \begin{pmatrix} \alpha_0 \\ \omega\alpha_1 \\ \vdots \\ \omega^{N-1}\alpha_{N-1} \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_0 \end{pmatrix}$.

If you have never seen this property before, it should be shocking. We will not offer a proof of this in general here, but below show this in the example that $N = 4$.

Let $|\Theta\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}$ and $|\Phi\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_0 \end{pmatrix}$. Then

$$\left|\hat{\Theta}\right\rangle = \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{pmatrix}\begin{pmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{pmatrix} = \frac{1}{2}\begin{pmatrix}\alpha_0+\alpha_1+\alpha_2+\alpha_3\\ \alpha_0+i\alpha_1-\alpha_2-i\alpha_3\\ \alpha_0-\alpha_1+\alpha_2-\alpha_3\\ \alpha_0-i\alpha_1-\alpha_2+i\alpha_3\end{pmatrix} = \begin{pmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \beta_3\end{pmatrix}$$

$$\left|\hat{\Phi}\right\rangle = \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{pmatrix}\begin{pmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ \alpha_0\end{pmatrix} = \frac{1}{2}\begin{pmatrix}\alpha_0+\alpha_1+\alpha_2+\alpha_3\\ -i\alpha_0+\alpha_1+i\alpha_2-\alpha_3\\ -\alpha_0+\alpha_1-\alpha_2+\alpha_3\\ i\alpha_0+\alpha_1-i\alpha_2-\alpha_3\end{pmatrix} = \begin{pmatrix}\beta_0\\ -i\beta_1\\ -\beta_2\\ i\beta_3\end{pmatrix}$$

The important point here is that the only difference between $\left|\hat{\Theta}\right\rangle$ and $\left|\hat{\Phi}\right\rangle$ is a relative phase shift. But does this matter?

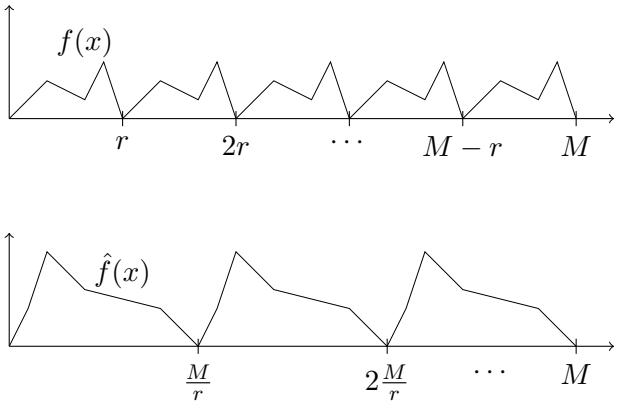
If we are going to measure a state, then the phases don't matter at all, because if the phase is ϕ, $\langle\phi|\phi\rangle = 1$. Therefore the phase of a given state does not effect the probability of measuring that state. However, there is a way we can gather information about the pahses.

We won't be able to tell by measuruing the difference between $\frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$ and $\frac{1}{2}\begin{pmatrix}1\\i\\-1\\-i\end{pmatrix}$ by making a measurment. However, if we apply QFT, we see that $QFT_4\frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ and $QFT_4\frac{1}{2}\begin{pmatrix}1\\i\\-1\\-i\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$. Thus, measuring the Fourier Transform of the states will reveal the relative phases.

3. *Period/Wavelength Relationship.* Suppose f is periodic with period r, for example

   Then $\hat{f}$ (the Fourier transform of f) is periodic with period M/r. Thus, $\hat{f}$ would look something like below figure.

   If r is the period of f, we can think of M/r as the wavelength of f. If you already have intuition for Fourier transform this should come as no



surprise. In general, the wider the range of a function, the sharper the range in the Fourier domain; and vise versa. In example, the fourier transform of a delta function is an even spread, while the transform of an even spread is a delta function.

We will prove this in a special case, where $f(x) = \begin{cases} \sqrt{\frac{r}{M}} & \text{if } x = 0 \pmod{r} \\ 0 & \text{otherwise.} \end{cases}$
While this is a very special case, it is actually the only case that we will need to develop Shor's algorithm. Furthermore this property *can* be proved in general.

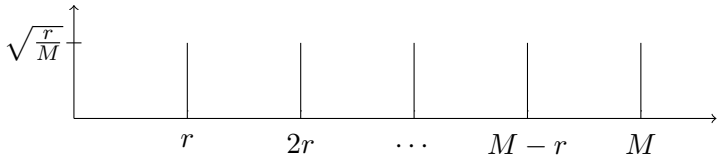


Figure 5.2: $f(x)$

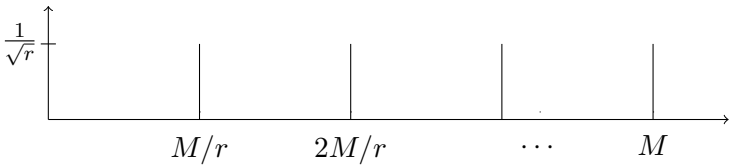


Figure 5.3: $\hat{f}(x)$

Because this function is relatively simple, we can prove the desired relationship by brute force. Suppose $|\alpha\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix}$ has Fourier transform $|\beta\rangle = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_N \end{pmatrix}$, then the jth component of its Fourier transform is given by

$$\beta_j = \frac{1}{\sqrt{M}} \sum_{i=0}^{N-1} \alpha_i \omega^{ij} \tag{5.1}$$

This expression comes from matrix multiplication, you should take a look at F_N to verify this if it looks unfamiliar. In our special case, $f(x) = \begin{cases} \sqrt{\frac{r}{M}} & \text{if } x = 0 \pmod{r} \\ 0 & \text{otherwise.} \end{cases}$. Using (1.1) we can calculate its Fourier transform

$$\hat{f}(x) = \frac{\sqrt{r}}{M} \sum_{i=0}^{\frac{M}{r}-1} \omega^{rix} \tag{5.2}$$

Where we have written $\hat{f}(x)$ instead of $\hat{f}_j$ because the $\hat{f}_j$ is equal to the value of the Fourier transform evaluated at $x = j$.

We have seen sums like this before. It is just a geometric series, and it is not too difficult to compute.

$$\sum_{i=0}^{\frac{M}{r}-1} \omega^{rix} = \frac{\omega^{Mx} - 1}{\omega^{rx} - 1}$$

But $\omega^{Mj} = 1$ (recall $\omega^M = 1$), so the numerator is always equal to 0. The only time the denominator can equal zero is when $rx = kM$, or $x = \frac{kM}{r} \equiv 0 \pmod{\frac{M}{r}}$. In this case, the numerator and the denominator are equivalently 0, so we must compute the limit using l'Hospitals rule.

$$\begin{aligned}\lim_{x\to k\frac{M}{r}} \frac{\omega^{Mx}-1}{\omega^{rx}-1} &= \lim_{x\to k\frac{M}{r}} \frac{M}{r}\frac{\omega^{Mx-1}}{\omega^{rx-1}} \\ &= \lim_{x\to k\frac{M}{r}} \frac{M}{r}\frac{\omega^{Mx}}{\omega^{rx}}\frac{\omega^{-1}}{\omega^{-1}} \\ &= \frac{M}{r}\end{aligned}$$

When we plug this result back into (1.2), the outcome is the desired result.

$$\hat{f}(x) = \begin{cases} \frac{1}{\sqrt{r}} & \text{if } x = 0 \pmod{\frac{M}{r}} \\ 0 & \text{otherwise.} \end{cases}$$

Notice that the normalization factor makes good sense.

To get a look at how we could prove this property in general, imagine periodic functions (in r) of this type as a basis for any periodic function. Allow the possibility of relative phase shifts, and you can prove this property in general.

Classical Fast Fourier Transform

The FFT was a major breakthrough for classical computers. Because the Fourier transform is an $M * M$ matrix, straightforward multiplication by F_M would take $O(M^2)$ steps to carry out, because multiplication of f on each row takes M multiplications. The FFT reduced this to $O(M \log M)$ steps. The FFT is incredibly important in signal processing that essentially all of your electronics rely on it. Without the FFT, modern electronics would have far fewer capabilities and would be much slower than they are today.

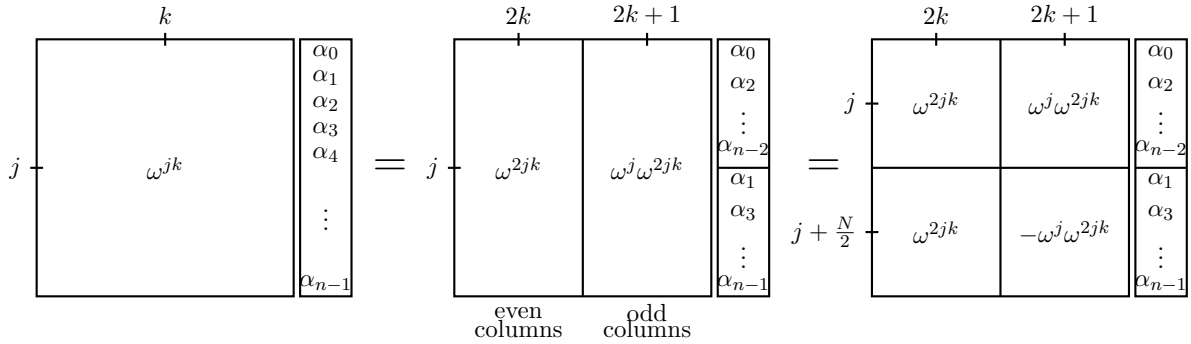
The FFT requires only that $M = 2^m$ for some integer m, but this is a relatively easy requirement because the computer can simply choose their domain.

The *fast* Fourier transform uses the symmetry of the Fourier transform to reduce the computation time. Simply put, we rewrite the Fourier transform of size M as two Fourier transforms of size $M/2$ - the odd and the even terms. We then repeat this over and over again to exponentially reduce the time. To see how this works in detail, we turn to the matrix of the Fourier transform. While we go through this, it might be helpful to have QFT_8 in front of you

to take a look at. Note that the exponents have been written modulo 8, since $\omega^8 = 1$.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix}$$

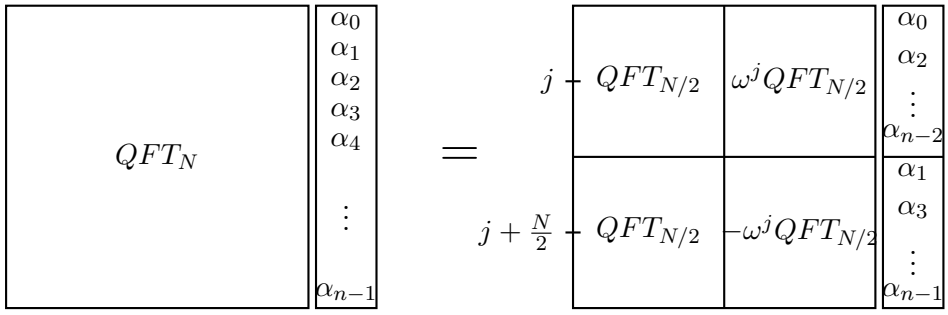
Notice how row j is very similar to row $j+4$. Also, notice how column j is very similar to column $j+4$. Motivated by this, we are going to split the Fourier transform up into its even and odd columns.



In the first frame, we have represented the whole Fourier transform matrix by describing the jth row and kth column: ω^{jk}. In the next frame, we separate the odd and even columns, and similarly separate the vector that is to be transformed. You should convince yourself that the first equality really is an equality. In the third frame, we add a little symmetry by noticing that $\omega^{j+N/2} = -\omega^j$ (since $\omega^{n/2} = -1$).

Notice that both the odd side and even side contain the term ω^{2jk}. But if ω is the primitive Nth root of unity, then ω^2 is the primitive $N/2$nd root of unity. Therefore, the matrices whose j,kth entry is ω^{2jk} are really just $QFT_{N/2}$! Now we can write QFT_N in a new way:

Now suppose we are calculating the Fourier transform of the function $f(x)$. We can write the above manipulations as an equation that computes the jth term $\hat{f}(j)$.



$$\hat{f}(j) = \left(F_{M/2}\overrightarrow{f_{\text{even}}}\right)(j) + \omega^j \left(F_{M/2}\overrightarrow{f_{\text{odd}}}\right)(j)$$

This turns our calculation of QFT_N into two applications of $QFT_{N/2}$. We can turn this into four applications of $QFT_{N/4}$, and so forth. As long as $N = 2^n$ for some n, we can break down our calculation of QFT_N into N calculations of $QFT_1 = 1$. This greatly simplifies our calculation.

Quantum Fourier Transform w/ quantum gates

The strength of the the FFT is that we are able to use the symmetry of the discreet Fourier transform to our advantage. The circuit application of QFT uses the same principle, but because of the power of superposition QFT is even faster.

The QFT is motivated by the FFT so we will follow the same steps, but because this is a quantum algorithm the implementation of the steps will be different. That is, we first take the Fourier transform of the odd and even parts, then multiply the odd terms by the phase ω^j.

In a quantum algorithm, the first step is fairly simple. The odd and even terms are together in superposition: the odd terms are those whose least significant bit is 1, and the even with 0. Therefore, we can apply $QFT_{M/2}$ to both the odd and even terms together. We do this by applying we will simply apply $QFT_{M/2}$ to the m-1 most significant bits, and recombine the odd and even appropriately by applying the Hadamard to the least significant bit.

Now to carry out the phase multiplication, we need to multiply each odd term j by the phase ω^j. But remember, an odd number in binary ends with a 1 while an even ends with a 0. Thus we can use the *controlled phase shift*, where the least significant bit is the control, to multiply only the odd terms by the phase without doing anything to the even terms. Recall that the controlled

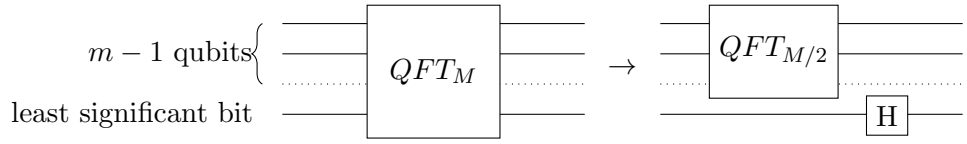


Figure 5.4: $QFT_{M/2}$ and a Hadamard gate correspond to $FFT_M/2$ on the odd and even terms

phase shift is similar to the CNOT gate in that it only applies a phase to the target if the control bit is one.

The phase associated with each controlled phase shift should be equal to ω^j where j is associated to the kth bit by $j = 2^k$.

Thus, apply the controlled phase shift to each of the first $m-1$ qubits, with the least significant bit as the control. With the controlled phase shift and the Hadamard transform, QFT_M has been reduced to $QFT_M/2$.

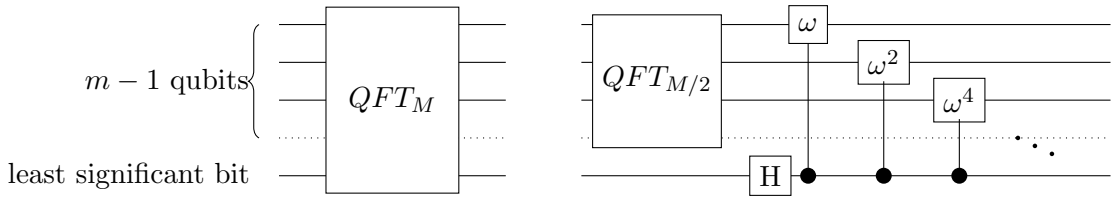


Figure 5.5: QFT_M is reduced to $QFT_{M/2}$ and M additional gates

Example

Lets construct QFT_3. Following the algorithm, we will trun QFT_3 into QFT_2 and a few quantum gates. Then continuing on this way we turn QFT_2 into QFT_1 (which is just a Hadamard gate) and another few gates. Controlled phase gates will be represented by R_ϕ.

then run through another iteration to get rid of QFT_2

You should now be able to visualize the circuit for QFT on more qubits easily. Furthermore, you can see that the number of gates necessary to carry out QFT_M it takes exactly $\sum_{i=1}^{\log M} i = \log M(\log M + 1)/2 = O(\log^2 M)$.

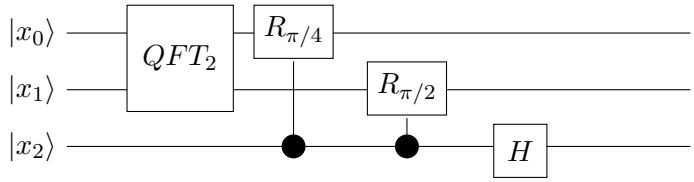


Figure 5.6: First Iteration

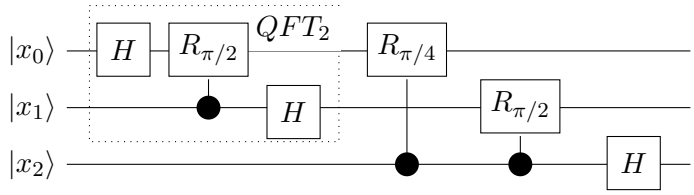


Figure 5.7: Second Iteration. Recall that $H = QFT_1$

5.2 Period Finding

Suppose we are given a black box for computing a periodic function, i.e. a function such that

$$f(x) = f(y) \text{ if and only if } x \equiv y \pmod{r}$$

The goal of a period finding algorithm is to find r.

The algorithm for period finding is very similar to Simon's algorithm, in fact we can think of it as a generalization of Simon's algorithm. The steps we follow are very similar.

Classically, we could solve this problem by querying our function with subsequent inputs until the function repeats. This takes $O(r) = O(2^n)$ queries to the function. There are other ways to solve this problem, but it can be shown that all classical algorithms solve this problem in exponential time.

With a Quantum computer, we can access the function in *superposition* to query the function with $N = 2^n$ inputs for each n qubits at the same time. The key ingredients to our approach will be the period/wavelength and linear shift properties of the Fourier transform. We first access the function in superposition to create a periodic superposition (that may include a linear shift), and then take its Fourier transform to get rid of the linear shift.

This approach is very similar to the approach of Simon's algorithm, and is the historical motivation for the period finding algorithm. Lets examine the details.

Step 1: Prepare the periodic superposition $\sqrt{\frac{r}{N}}\sum_{j=0}^{N/r-1}|x_0+jr\rangle$

Step 2: Fourier sample to produce $y=\frac{kN}{r}$ for $k\in\{0,1,\cdots,r-1\}$.

Step 3: Repeat until there are enough such y's so that we can compute their greatest common divisor and solve for r.

Step 1. It is best to start a quantum algorithm with the easily prepared state $|0\rangle$, but we need to prepare the quantum state $\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|0\rangle$ so that we can access our function in superposition. Noticing that the Fourier transform of the state $|0\rangle$ produces the disired state, we will implement it on the first n qubits, so that

$$|0\rangle|0\rangle \xrightarrow{QFT_N} \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|0\rangle$$

Next, as in Simon's algorithm, we access our function in superposition. Let U_f be the unitary transformation that carries out our function, and implement it so that

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|0\rangle \xrightarrow{U_f} \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|f(x)\rangle$$

To get a periodic superposition out of this, we measure $|f\rangle$. Then $|f\rangle$ must collapse into some value $f(x_0)$. Furthermore, because measuring $|f\rangle$ reveals information about $|x\rangle$, the state $|x\rangle$ will also collapse into the pre-image of $f(x_0)$. But because f is periodic, the pre-image of $f(x_0)$ is $\{x_0, x_0+r, x_0+2r,\cdots,x_0+(\frac{N}{r}-1)r\}$. So our measurement makes this change:

$$\frac{1}{\sqrt{8}}\sum_{x=0}^{N-1}|x\rangle\otimes|f(x)\rangle \xrightarrow{\text{measure}|f\rangle} \sqrt{\frac{r}{N}}\sum_{i=0}^{N/r-1}|ir+x_0\rangle|f(x_0)\rangle$$

Now our first register is in a periodic superposition, where the period is the same as the period of the function! But we can't just measure, because each time we run the algorithm, we might measure a different value of $|f\rangle$, thus obtaining a periodic superposition that is linearly shifted.

Step 2: We can't just measure our superposition right away, because that would destroy the superposition, and because of the random linear shift x_0 we wouldn't acquire any useful information. Instead, we will rely on the properties of the Fourier transform to retrieve the information we want. Remember that if f is periodic with period r such that $N/r=k$, then $\hat{f}$ is periodic with period k. Furthermore, remember that we only see the effect of the linear shift x_0 in the phase of $\hat{f}$. Therefore if we take the Fourier transform of the first register, we will be left only with states that are multiples of N/r.

$$\sqrt{\frac{r}{N}} \sum_{i=0}^{N/r-1} |ir + x_0\rangle \xrightarrow{QFT_N} \frac{1}{\sqrt{r}} \sum_{i=0}^{r-1} \left| i\frac{N}{r} \right\rangle \phi_i$$

where ϕ_i is some unimportant phase associated with each term due to the linear shift x_0.

Now we can measure and retrieve $k\frac{N}{r}$ for some integer k!

Step 3 Now we repeat the algorithm to retrieve several distinct multiples of N/r. Once we have enough values, we can compute their GCD to retrieve N/r. N is a given in the problem, so it is easy to compute r. Computing GCD is easy thanks to Euclid's algorithm.

How long should we expect this to take? Let us compute the chance of finding the correct period after t samples. Suppose after finding t distinct multiples of N/r, we have not found the desired period N/r, but instead a multiple, say $\lambda N/r$. This means that each of the t samples must be a multiple of $\lambda N/r$. There are exactly $N/(\lambda N/r) = r/\lambda$ multiples of $\lambda N/r$, and since there are r multiples in total, the probability of getting a multiple of $\lambda N/r$ is $1/\lambda$. Therefore,

$$\Pr[\text{gcd is a multiple of } N/r] = \left(\frac{1}{\lambda}\right)^t \leq \left(\frac{1}{2}\right)^t,$$

and we err with probability

$$\Pr[\text{gcd} > N/r \text{ after } t \text{ samples}] \leq N\left(\frac{1}{2}\right)^t.$$

Therefore we must repeat the period finiding circuit $O(\log N)$ times to be confident in our solution.

The above algorithm can be summed up by the circuit:

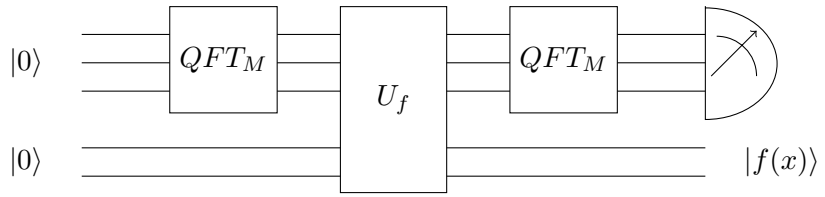


Figure 5.8: Circuit for period finding

Example

In this example we find the period of the function $f(x) = x \pmod 2$. It is easy to see that the period of this function is $r = 2$

We will use a 3-qubit system so that $N = 8$. It is a good rule of thumb to choose $N \gg r$. The first step is to apply the quantum Fourier transform:

$$|0\rangle |0\rangle \overset{QFT_8}{\longrightarrow} \frac{1}{\sqrt{8}} \sum_{x=0}^{7} |x\rangle |0\rangle$$

Next we apply our function.

$$\frac{1}{\sqrt{8}} \sum_{x=0}^{7} |x\rangle |0\rangle \overset{U_f}{\longrightarrow} \frac{1}{\sqrt{8}} \sum_{x=0}^{7} |x\rangle |x \bmod 2\rangle$$

The next step is to measure $|f\rangle$. Then $|f\rangle$ must collapse into either $|0\rangle$ or $|1\rangle$. For the purpose of demonstration, lets say our measurement returns $|f(x)\rangle = |1\rangle$. Then x must be odd.

$$\frac{1}{\sqrt{8}} \sum_{x=0}^{7} |x\rangle \otimes |f(x)\rangle \overset{measure|f\rangle}{\longrightarrow} \frac{1}{2}(|1\rangle + |3\rangle + |5\rangle + |7\rangle) \otimes |1\rangle$$

Now we need to extract the period of the first register without the obnoxious linear shift. So once again we apply the Fourier transform.

$$\frac{1}{2}(|1\rangle + |3\rangle + |5\rangle + |7\rangle) \overset{QFT_8}{\longrightarrow} \frac{1}{\sqrt{2}}(|0\rangle - |4\rangle)$$

Note: If instead of measuring $|f\rangle = |1\rangle$ we had measured $|f\rangle = |0\rangle$, there would be a different linear shift. But the properties of Fourier transform dictate that this only effects the *phase* of the Fourier transform. In other words, that last step would have looked like $\frac{1}{2}(|0\rangle + |2\rangle + |4\rangle + |6\rangle) \overset{QFT_8}{\longrightarrow} \frac{1}{\sqrt{2}}(|0\rangle + |4\rangle)$. This agrees with what we know about the principal of deferred measurement.

Finally, if we take a few measurements we will be sure to measure both $|0\rangle$ and $|4\rangle$. Therefore $N/r = 4$, and since $N = 8$, it is clear that $r = 2$.

Summary

Now that we understand how the algorithm works, we can write it without some of the fluff.

$$|0\rangle\,|0\rangle \xrightarrow{QFT_M} \frac{1}{\sqrt{M}} \sum_{x \in \mathbf{Z}_M} |x\rangle\,|0\rangle \tag{5.1}$$

$$\xrightarrow{f} \frac{1}{\sqrt{M}} \sum_{x \in \mathbf{Z}_M} |x\rangle\,|f(x)\rangle \tag{5.2}$$

$$\xrightarrow{\text{measure 2nd register}} \sqrt{\frac{r}{M}} \sum_{k=0}^{\frac{M}{r}-1} |x_0 + kr\rangle\,|f(x_0)\rangle \tag{5.3}$$

$$\xrightarrow{QFT_M} \sqrt{\frac{r}{M}} \frac{1}{\sqrt{M}} \sum_{y \in \mathbf{Z}_M} \alpha_y\,|y\rangle \tag{5.4}$$

where $\alpha_y = \sum_{k=0}^{M/r-1} \omega^{(x_0+kn)y} = \omega^{x_0 y} \sum_k \omega^{kry}$.

There are two cases for y:

1. Case 1: y is a multiple of $\frac{M}{r}$.

   In this case, then $\omega^{kry} = e^{2\pi i r y/M} = e^{n2\pi i} = 1$. So $\alpha_y = \frac{\sqrt{r}}{M}\frac{M}{r} = \frac{1}{\sqrt{r}}$. This should be thought of as *constructive interference* due to the final QFT_M.

   Note that there are r multiples of M/r. Because $\sum_1^r \frac{1}{\sqrt{r}}^2 = 1$, we know that $\alpha_y = 0$ for any y that is *not* a multiple of $\frac{M}{r}$ by normality.

2. Case 2: y is not a multiple of $\frac{M}{r}$.

   We already know that α_y must be 0 from the previous case. Furthermore, note that $\omega^{ry}, \omega^{2ry}, \ldots$ are evenly spaced vectors in the complex plain of unit length around the origin. Summing over these vectors we see that α_y is 0. This can be viewed as *destructive interference* due to the final QFT_M.

The interference that occurs in the final step is one reason quantum computers are so well equipped for period finding. We call it interference becuase it is additions in the *phase* that cause the cancellations.

5.3 Shor's Quantum Factoring Algorithm

One of the most celebrated algorithms for quantum computers is Shor's Algorithm for factoring. The time it takes for a classical computer to factor some number N grows as a polynomial in N, while the time it takes a quantum computer grows as a polynomial in log(N).

The reason we focused so much attention on period finding is because the problem of factoring can be reduced to the problem of period finding by using modular arithmetic. This isn't obvious, but we can understand it with the following two lemmas.

Setup

In modular arithmetic, we call a number x a non-trivial square root of 1 modulo N if $x^2 \equiv 1 \pmod{N}$ and $x \neq \pm 1$. For example, 2 is a non-trivial square root of unity modulo 3 because $2^2 = 4 \equiv 1 \pmod{3}$. It turns out that if we can find such an x, we can factor N. Furthermore, we can use period finding to find x. This idea is summed up in the following lemmas.

Lemma Factoring is equivalent to finding a nontrivial squareroot of 1 mod N.

Proof Let $x \neq \pm 1 \mod N$ and $x^2 = 1 \mod N$. Then $x^2 - 1 = 0 \pmod{N}$ so that $x^2 - 1$ is a multiple of N. Factoring, we see that $N \mid (x+1)(x-1)$, but because $x \neq \pm 1 \pmod{N}$, $N \nmid (x \pm 1)$.

Therefore, $\gcd(N, x+1)$ and $\gcd(N, x-1)$ are factors of N, and greatest common divisor is easy to compute with Euclid's algorithm.

Example: Suppose we want to factor the number 15. It is easy to see that $4^2 = 16 \equiv 1 \mod 15$, but $4 \neq \pm 1 \mod 15$. So 4 is a non-trivial square root of unity modulo 15. Then gcd(15,5) and gcd(15,3) are factors of 15. Sure enough we see that $5 \cdot 3 = 15$.

Now, all we need to do is find this nontrivial squareroot of unity, and we can factor whatever number we need. As promised, we can do this with period finding, specifically by computing the order of a random integer.

The **order** of some integer x modulo N is the smallest integer r such that $x^r = 1 \mod N$. For example, the order of 2 modulo 3 is 2 since $2^2 \equiv 1$, the order of 3 modulo 5 is 4 since $3^2 = 9 \equiv 4$; $3^3 = 25 \equiv 2$; and $3^4 = 81 \equiv 1 \pmod{5}$. Another way to say this is that the order of x is just the period of the function $f(i) = x^i \mod N$.

Lemma Suppose $N = p \cdot q$, and $x \in \mathbf{Z}_N$, $x \neq p, q$. Then with probability $\geq 1/2$, the order s of x is even, and $x^{s/2}$ is a nontrivial square root of 1 mod N.

The proof of this statement requires results from number theory (Fermat's little Theorem, Chinese remainder Theorem) that are outside the scope of this course, so we will state it without proof. However, it should be intuitive: if you imagine the order of a number to vary randomly from one number to the next, you expect the order of a number to be even with probability about half.

Example: Find the order of $2^a \pmod{63}$, and use it to factor 63.

1. $2 = 2$
2. $2^2 = 4$
3. $2^3 = 8$
4. $2^4 = 16$
5. $2^5 = 32$
6. $2^6 = 64 \equiv 1 \pmod{63}$

so that the order of 2 is 6. Note that a quantum computer wouldn't have to compute each of these powers, it would simply use the period finding algorithm described earlier. Now we compute $2^3 = 8 \neq \pm 1$, so that $\gcd(63, 8+1) = 9$ and $\gcd(63, 8-1) = 7$ are factors of 63.

The Algorithm

When finding order using the period finding algorithm, it is important to use enough qubits. A sensible rule is that you need to use m qubits so that $2^m \gg N^2$, where N is the number we are trying to factor, because the order of a random number might be as large as N.

We now have all the necessary tools to carry out Shor's algorithm. Start by picking a random number, then use the period finding algorithm to compute its order. If the order is even, we can use it to find a nontrivial square root of unity. If the order is odd or $x^{s/2} = -1$, throw it out and start with a new number.

Because we know that the order of x will be even and $x^{s2/}$ will be a nontrivial square root with probability at least $1/2$, we can be confident that we will be able to factor N in just a few runs of the algorithm. Because the time it takes to find the period grows as a polynomial in the number of bits, and the number of bits grows like $2 \log N$(by the above requirement), we expect the time it takes to factor N to grow as a polynomial in $\log N$.

Here is the circuit for Shor's Algorithm. It relies heavily on period finding, and so the circuit looks a lot like the circuit for period finding. The key difference is that we are finding the period of $f(i) = x^i$, and the number of bits we need to input is very large.

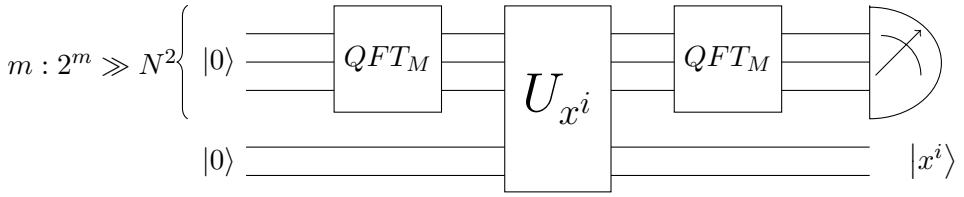


Figure 5.9: Circuit for factoring

Example

Here's an example that's a little more fun. Lets factor 119. Suppose we pick the number 16 to start with.

First, we compute it's order.

1. $16 = 16$
2. $16 \cdot 16 = 256 \equiv 18$
3. $18 \cdot 16 = 288 \equiv 50$
4. $50 \cdot 16 = 800 \equiv 86$
5. $86 \cdot 16 = 1376 \equiv 67$
6. $67 \cdot 16 = 1072 = 119 \cdot 7 + 1 \equiv 1$

so that the order of 16 mod 119 is 6. Now, we compute $16^3 \equiv 50$. Gcd(49,119) = 7, so 7 is a factor of 119, and gcd(51, 119) = 17 which is another factor of 119.