❖ Computation 2: Quantum Fourier Transform and Shor's Algorithm

"How dare we dream

Of solving problems that else would take more time

Than has passed since the cosmos's Big Bang!"

- Peter Shor

2.1 Quantum Fourier Transform

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- describe and analyze the effect of the quantum Fourier transform on a given input state.

We begin by reviewing how to transition between the bit string and integer representation. To compute the value of the bit string 11001, we multiply it as follows:

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 1 = 25. \tag{15}$$

More generally, if we have a string $x = x_1 \cdots x_n$ where x_i is the i-th bit of x, we can write this as an integer using the sum

$$x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \cdots + x_n \cdot 2^0 = \sum_{k=1}^{n} x_k \cdot 2^{n-k}. \tag{16}$$

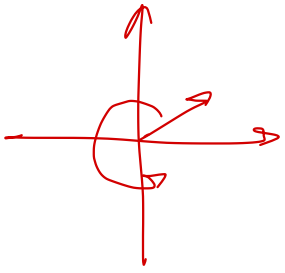
Throughout this section, we will frequently take complex numbers to the power of integers, and it will be useful to have a way to toggle between the integer representation of the power and the bit string representation.

$$\omega^x = \omega^{x_1 \cdot 2^{n-1} + x_2 2^{n-2} + \cdots + x_n 2^0} \tag{17}$$

$$= \omega^{x_1 \cdot 2^{n-1}} \cdot \omega^{x_2 \cdot 2^{n-2}} \cdot \cdots \cdot \omega^{x_n \cdot 2^0} \tag{18}$$

$$= \prod_{k=1}^{n} \omega^{x_k \cdot 2^{n-k}} \tag{19}$$

For the rest of the course, we will use the shorthand $N = 2^n$. One important reason we will be interested in complex numbers is because they are a great tool for analyzing periodic functions. In particular, the N-th roots of unity give us a way to keep track of the "period" in the way a clock would.



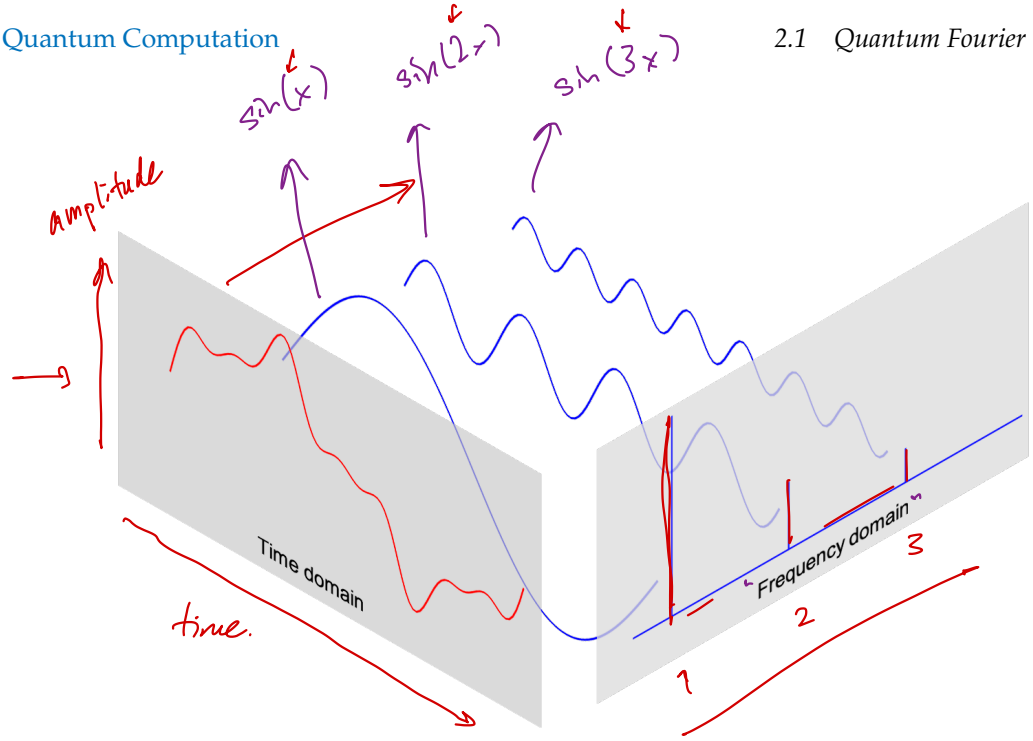
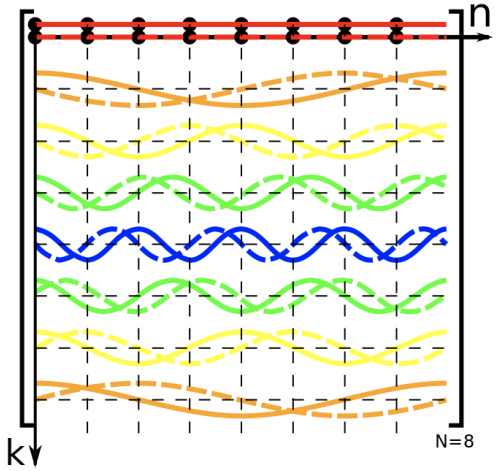


Figure 1: Taken from Python Numerical Methods

You may have learned about the Discrete Fourier Transform (DFT). The DFT is often used for signal processing, where a signal could be a sound wave, radio signal, or daily temperature readings. Usually we describe these signals in the **time domain**. Instead of doing this, we can take a slice of time to describe a signal in the **frequency domain**. By doing this, we have a discrete set of items to build our wave out of, and we can safely discard frequencies that are too high or low for the human ear.



$\omega^3 = e^{(2\pi i/2^3)\cdot 3}$

$= e^{6\pi i/8}$

$= e^{i\,3\pi/4}$

$= \cos\left(\frac{3\pi}{4}\right) + i\sin\left(\frac{3\pi}{4}\right)$

3 qubit $|\vec{n}\rangle$

$\begin{bmatrix} \omega^{0\cdot n} \\ \omega^{1\cdot n} \\ \omega^{2\cdot n} \\ \omega^{3\cdot n} \\ \omega^{4\cdot n} \\ \omega^{5\cdot n} \\ \omega^{6\cdot n} \\ \omega^{7\cdot n} \end{bmatrix}$

Early in the investigation of quantum algorithms, researchers figured out that a quantum circuit can compute the Fourier transform very efficiently when the input vector is encoded in the amplitudes of a quantum state. Note that this is not necessarily a faster way to compute the classical Fourier transform, since the output is only accessible via quantum measurement.

Definition 2.1 (Quantum Fourier Transform)**.** Let $N = 2^n$. For $x \in \{0, 1, \ldots, N-1\}$:

- Standard basis state: A length N column vector with a 1 in the k-th location

$$|k\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{20}$$

- Fourier basis state ($\omega = e^{2\pi i/N}$ is the first N-th root of unity):

$$|\tilde{k}\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} \omega^0 \\ \omega^k \\ \omega^{2k} \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}. \tag{21}$$

The n-qubit **Quantum Fourier Transform** or QFT_N is the transformation that performs

$$QFT_N |x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle \tag{22}$$

We can model the action of QFT_N by the matrix

$$QFT_N := \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}. \tag{23}$$

Let's get familiar with the Fourier basis by working through an explicit example.

Question 30. Write down the 1-qubit QFT matrix. Use the matrix to find all of the Fourier basis states.

Question 31. Write down the 2-qubit QFT matrix. Use the matrix to find the first Fourier basis state, $|\tilde{1}\rangle$.

Let's see if we can describe Fourier basis states for the general case. To do this, we will use the integer to bit string mapping we discussed earlier. Recall that the mapping we are interested in is

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle \,. \tag{24}$$

Let $y = y_1 \cdots y_n$ be the bitstring representation of y.

Question 32. Write down the value of y using the bits $y_1, \ldots, y_n$.

Let's use this to rewrite the power of the N-th root of unity.

$$\omega^{xy} = \omega^{x[y_1 \cdot 2^{n-1} + y_2 \cdot 2^{n-2} + \cdots + y_n \cdot 2^0]} \tag{25}$$

$$= \prod_{j=1}^{N} \omega^{x \cdot y_j \cdot 2^{n-j}}. \tag{26}$$

We can then rewrite the Fourier basis state as

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle \tag{27}$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{j=1}^{N} \omega^{x \cdot y_j \cdot 2^{n-j}} |y\rangle \tag{28}$$

$$= \frac{1}{\sqrt{N}} \left(|0\rangle + \omega^{x \cdot 2^{n-1}} |1\rangle\right) \otimes \left(|0\rangle + \omega^{x \cdot 2^{n-2}} |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + \omega^{x \cdot 2^{0}} |1\rangle\right) \tag{29}$$

Question 33. Consider a 3-qubit Fourier transform. What is $|\tilde{7}\rangle$? What is $|\tilde{7}\rangle$ written in the form of equation (29)?

Question 34. What is the amplitude of $|101\rangle$ in $|\tilde{7}\rangle$?

Equation (29) gives us a way to view the mapping as a tensor product of n qubits:

$$|x\rangle = |x_1\rangle \otimes \cdots \otimes |x_n\rangle \tag{30}$$

$$\leftrightarrow \frac{1}{\sqrt{N}} \left(|0\rangle + \omega^{x \cdot 2^{n-1}} |1\rangle\right) \otimes \left(|0\rangle + \omega^{x \cdot 2^{n-2}} |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + \omega^{x \cdot 2^{0}} |1\rangle\right). \tag{31}$$

To summarize the algorithm, we will perform the following mapping between qubits and then reverse the order of the qubits using swap gates at the end:

$$|x_k\rangle \to \frac{1}{\sqrt{2}} \left(|0\rangle + \omega^{x \cdot 2^{k-1}} |1\rangle\right). \tag{32}$$

Note that the state is like a $|+\rangle$ state with an extra relative phase. Let's try to determine what this relative phase is.

Before stating it generally, let's take a closer look at this for a particular example. Let's look at $|\tilde{7}\rangle$ from the 3 qubit Fourier transform we were studying earlier.

Question 35. The input to the QFT circuit will be $|7\rangle = |1\rangle \otimes |1\rangle \otimes |1\rangle$. Write down the relative phase of each qubit after the QFT circuit is applied using the bitstring representation of x.

$$QFT_{2^3}|7\rangle = QFT_8(|1\rangle \otimes |1\rangle \otimes |1\rangle)$$

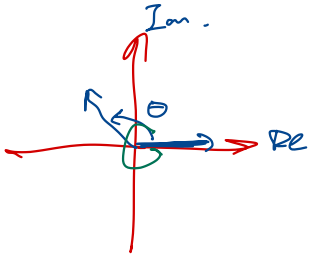
$$= \frac{1}{\sqrt{2^3}}(|0\rangle + \omega^{7\cdot 2^2}|1\rangle) \otimes (|0\rangle + \omega^{7\cdot 2}|1\rangle) \otimes (|0\rangle + \omega^{7\cdot 2^0}|1\rangle)$$

Qubit 2: $|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + \omega^{7\cdot 2^2}|1\rangle)$

$$(x^a)^b = x^{a\cdot b}$$

$$\omega^{7\cdot 2^2} = \left(e^{i2\pi/2^3}\right)^{7\cdot 2^2} = e^{i2\pi\left(\frac{2^2}{2^3}\right)7} = e^{i2\pi\left(\frac{2^2}{2^3}\right)(1\cdot 2^2 + 7\cdot 2^1 + 1\cdot 2^0)}$$

$$= e^{i2\pi\left(\frac{2^2}{2^3}\right)2^2} \cdot e^{i2\pi\left(\frac{2^2}{2^3}\right)2} \cdot e^{i2\pi\left(\frac{2^2}{2^3}\right)\cdot 1}$$



$$= e^{i2\pi\cdot 2} \cdot e^{i2\pi\cdot 1} \cdot e^{i2\pi/2}$$

$e^{i\theta}$ If θ is $k\cdot 2\pi$. $\quad 1 \quad 1 \quad e^{i\pi} = -1$

$$= -1.$$

We can analyze the amplitudes more generally for an n-qubit QFT circuit as follows.

$$\omega^{x\cdot 2^{k-1}} = e^{\frac{2\pi i\cdot 2^{k-1}}{2^n}\cdot x} \tag{33}$$

$$= e^{2\pi i\left[\frac{2^{k-1}}{2^n}\right]\left[x_1\cdot 2^{n-1}+x_2\cdot 2^{n-2}+\cdots+x_n\cdot 2^{n-n}\right]} \tag{34}$$

$$= \prod_{j=1}^{n} e^{2\pi i\left[\frac{2^{k-1}}{2^n}\right]\cdot x_j\cdot 2^{n-j}} \tag{35}$$

$$= \prod_{j=k}^{n} e^{\frac{2\pi i}{2^{j-k+1}}\cdot x_j}. \tag{36}$$

Note the index change in the last line. In the case where $(n-j)+(k-1)\geq n$, then the exponent is an integer multiple of $2\pi i$, which makes the term in the product always equal 1. The condition can be simplified to $k-1\geq j$, meaning we can discard the terms in the product less than k.

The final line gives insight into what the circuit may need to look like. Since x_j is the j-th bit of x, we see that when $x_j=0$, it will kill off that entire term (set it to 1). In other words, we only want to apply the phase when $x_j=1$. This sounds a lot like a controlled gate!

Question 36. What is the relative phase of the second qubit after applying a QFT circuit to the three qubit input state $|101\rangle$?

The controlled gate we want to apply has to apply a relative phase conditioned on the x_j-th qubit being 1. To do this, let's define a new gate:

$$P_a = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^a} \end{bmatrix}. \tag{37}$$

We have all the pieces now to construct the algorithm.

Algorithm 1 Quantum Fourier Transform

1: **for** $k = 1$ to n **do** ▷ Apply $|x^k\rangle \longrightarrow |0\rangle + e^{2\pi i \cdot x \cdot 2^{k-1}} |1\rangle$
2: Apply H to $|x_k\rangle$
3: **for** $j = k + 1$ to n **do**
4: if $x_j = 1$, apply R_{j-k+1}
5: **end for**
6: **end for**
7: Reorder the qubits using swap gates

Question 37. Draw the Quantum Fourier Transform circuit for 4 qubits.