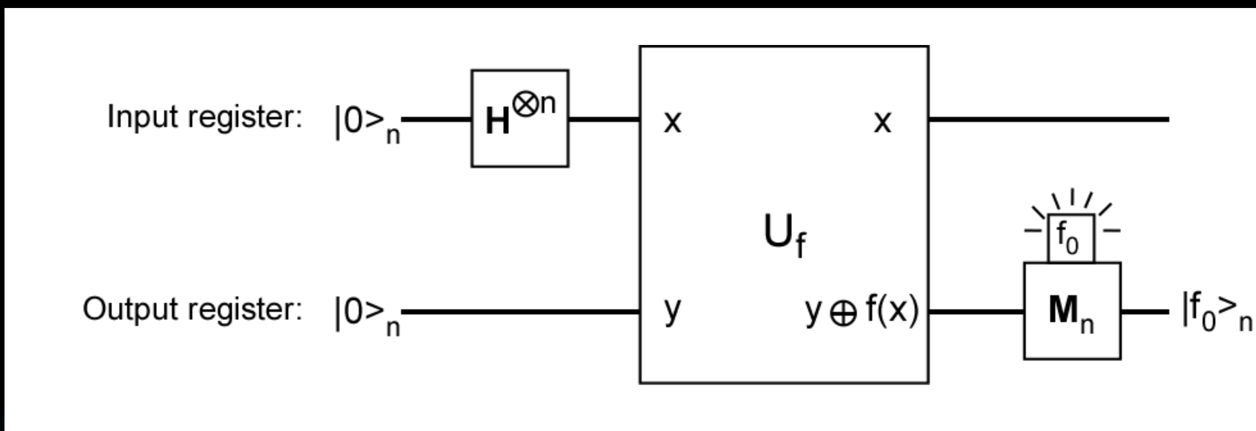
The Quantum Fourier Transform

In 1994, Peter Shor thought about using an “oracle query” approach to find the period of a function

$$f(x+r) = f(x) = b^x \text{ (mod N)}$$



$$|\psi_3\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle_n |f_0\rangle_{n_0}$$

The quantum state, after measurement of the output register, is a state which is periodic in *r*.

Goal: Explain the approach needed to solve the problem!

A. Fourier Transforms

Fourier transforms are transformation in the *representation* of a function.

1. Continuous Fourier Transforms

As originally conceived the Fourier transform allows the representation of a function as *either* a function in time or a function of frequency.

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t}\, dt$$

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t}\, df$$

In this definition, both *h(t)* and *H(f)* are complex functions. If h(t) is real, then

$$H(-f) = H^*(f)$$

An important feature of the Fourier transform is “time shifting” or “frequency shifting”

$$h(t - t_0) \Longleftrightarrow H(f) e^{2\pi i f t_0}$$

$$h(t) e^{-2\pi i f_0 t} \Longleftrightarrow H(f - f_0)$$

a. Fourier transforms and position- and momentum-representation

In physics we often consider the same state vector as a function of position or of momentum.

The state vector is $|\psi\rangle$

It is represented in the position basis as $\langle x|\psi\rangle = \psi(x)$

and in the momentum basis as $\langle p|\psi\rangle = \phi(p)$

$$\phi(p) = \langle p|\psi\rangle = \frac{1}{\sqrt{2\pi\hbar}}\int_{-\infty}^{\infty}\langle x|\psi\rangle e^{-ipx/\hbar}\,dx$$

$$\psi(x) = \langle x|\psi\rangle = \frac{1}{\sqrt{2\pi\hbar}}\int_{-\infty}^{\infty}\langle p|\psi\rangle e^{+ipx/\hbar}\,dp$$

This version of the equation is explicit in pointing out that the Fourier transform is simply a (unitary) basis transformation of the same state.

2. Discrete Fourier Transforms (DFTs)

Often data is discretized into a time-series

$$h(t_k) = h(k\Delta) \qquad k = 0, 1, 2, ..., N-1$$

Where Δ is the time interval between samples (called the "sampling rate").

The *sampling theorem* says that if a continuous function contains no frequencies greater than the *Nyquist frequency*

$$f_c \equiv \frac{1}{2\Delta}$$

then the continuous function is fully defined by samples taken at interval Δ. The Fourier transform of a discrete finite set of data is defined at a set of frequencies

$$f_j = \frac{j}{N\Delta} \qquad j = -\frac{N}{2}, -\frac{N}{2}+1, ..., \frac{N}{2}-1, \frac{N}{2}$$

In this presentation of DFTs I have used the notation of *Numerical Recipes,* by Press, Flannery, Teukolsky, and Vetterling. Lots more information on DFTs – including background information and sample code can be found in the various editions of that book.

a. Replacing the integral by a sum

For discretized data we can approximate the Fourier integral by a discrete sum.

$$H_j = \sum_{k=0}^{N-1} h_k e^{2\pi i jk/N}$$

$$h_k = \frac{1}{N} \sum_{j=0}^{N-1} h_j e^{-2\pi i jk/N}$$

The only difference between the DFT and its inverse is the change of sign in the exponent and the factor of 1/N.

With this definition the discrete Fourier transform does not depend on the time scale Δ or the frequency scale $1/(2\Delta)$.

An equally valid way of writing the DFT is:

$$H_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} h_k \left(e^{2\pi i/N}\right)^{kj} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} h_k \left(\omega\right)^{kj}$$

$$h_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} H_j \left(e^{-2\pi i/N}\right)^{jk} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} H_j \left(\omega^*\right)^{kj}$$

Where $\omega \equiv e^{2\pi i/N}$ and $\omega^* \equiv e^{-2\pi i/N}$

b. DFTs in matrix form

To put this in the context of quantum mechanics it's useful to think of *h* and *H* as vectors

$$\underset{\sim}{h} = (h_0, h_1, ..., h_{N-1})^T$$

$$\underset{\sim}{H} = (H_0, H_1, ..., H_{N-1})^T$$

The transformation between the two vectors is a *unitary* matrix

$$\begin{pmatrix} H_0 \\ H_1 \\ H_2 \\ \vdots \\ H_{N-2} \\ H_{N-1} \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & \omega^{N-2} & \omega^{2(N-2)} & \dots & \omega^{(N-2)(N-1)} \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)^2} \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{N-2} \\ h_{N-1} \end{pmatrix}$$

Notice that this is an *N x N* matrix multiply! In this form the DFT requires N^2 multiplications. For an *n*-bit number *N*, this is $(2^n)^2$, which is exponential in *n*.

c. The Fast Fourier Transform (FFT)

An algorithm for a "fast" way to evaluate the DFT was made widely known in the mid-1960s by IBM researchers James Cooley and John Tukey. In an FFT the DFT can be evaluated in $N\log N$ steps.

The approach is a recursive application of splitting the problem into two parts:

$$H_j = \sum_{k=0}^{N-1} h_k \left(e^{2\pi i/N}\right)^{kj} = \sum_{k=0}^{N-1} h_k \left(\omega\right)^{kj}$$

$$= \sum_{k=0}^{N/2-1} h_{2k} \left(\omega\right)^{(2k)j} + \sum_{k=0}^{N/2-1} h_{2k+1} \left(\omega\right)^{(2k+1)j}$$

$$= \sum_{k=0}^{N/2-1} h_{2k} \left(\omega^2\right)^{kj} + \omega^k \sum_{k=0}^{N/2-1} h_{2k+1} \left(\omega^2\right)^{kj}$$

A DFT using this equation would take $2(N/2)^2$ operations. Using the approach recursively until each individual transform is of length 1 requires $\log_2 N$ divisions. Doing N transformations gives a total operation count of $O(N\log_2 N)$.

While a huge advantage in practice for computing DFTs, $N\log N$ is STILL exponential in the number of bits, requiring $O(n2^n)$ operations.

3. The Quantum Fourier Transform (QFT)

The n-Qbit Quantum Fourier Transform is a unitary basis transformation defined in exactly the same way as the DFT. It acts on the basis states |x>

$$\mathbf{U}_{FT}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{0 \le y < 2^n} \left(e^{2\pi i/2^n}\right)^{xy} |y\rangle = \frac{1}{\sqrt{2^n}} \sum_{0 \le y < 2^n} \omega^{xy} |y\rangle$$

where *x* and *y* are n-bit integers, and *xy* represents ordinary integer multiplication. Since the operation is linear, it acts on a general superposition of states

$$|\psi\rangle = \sum_{0 \le x < 2^n} \gamma(x)|x\rangle$$

to give

$$\begin{aligned} \mathbf{U}_{FT}|\psi\rangle &= \sum_{0 \le x < 2^n} \gamma(x)\mathbf{U}_{FT}|x\rangle \\ &= \sum_{0 \le x < 2^n} \gamma(x) \frac{1}{\sqrt{2^n}} \sum_{0 \le y < 2^n} \omega^{xy} |y\rangle \end{aligned}$$

If we reverse the order of the summations:

$$\mathbf{U}_{FT}|\psi\rangle = \sum_{0\le y<2^n} \underbrace{\frac{1}{\sqrt{2^n}} \sum_{0\le x<2^n} \gamma(x)\,\omega^{xy}}_{\tilde{\gamma}(y)} |y\rangle$$

Giving:

$$\mathbf{U}_{FT}|\psi\rangle = \sum_{0\le y<2^n} \tilde{\gamma}(y)|y\rangle$$

The coefficients of the transformed state are just the Fourier transformed coefficients of the original state.

$$\tilde{\gamma}(y) = \frac{1}{\sqrt{2^n}} \sum_{0\le x<2^n} \gamma(x)\,\omega^{xy}$$

a. QFTs in matrix form

The n-Qbit Quantum Fourier Transform is a unitary basis transformation between two states, which have N = 2^n components:

$$\begin{pmatrix} \tilde{\gamma}_0 \\ \tilde{\gamma}_1 \\ \tilde{\gamma}_2 \\ \vdots \\ \tilde{\gamma}_{2^n-2} \\ \tilde{\gamma}_{2^n-1} \end{pmatrix} = \frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{2^n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(2^n-1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & \omega^{2^n-2} & \omega^{2(2^n-2)} & \dots & \omega^{(2^n-2)(2^n-1)} \\ 1 & \omega^{2^n-1} & \omega^{2(2^n-1)} & \dots & \omega^{(2^n-1)^2} \end{pmatrix} \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{2^n-2} \\ \gamma_{2^n-1} \end{pmatrix}$$

Where the parameter $\omega \equiv e^{2\pi i/2^n}$

Again, this is an *N x N* matrix multiply! In this form the QFT requires N^2 multiplications. For an *n*-bit number *N*, this is $(2^n)^2$, which is exponential in *n* and therefore is **not** efficient.

Example: The 3-Qbit QFT matrix

For n = 3 there are 2^3 = 8 basis states |x>, and the matrix can be written out as

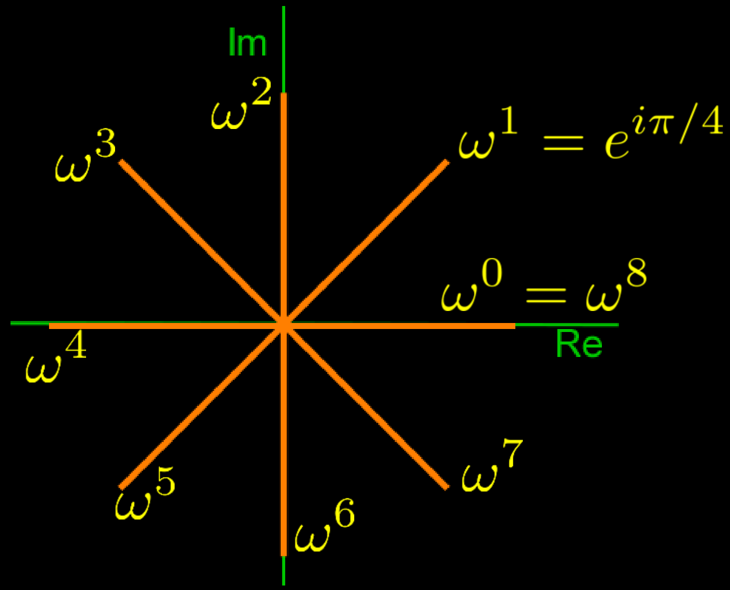
$$\begin{pmatrix} \tilde{\gamma}_0 \\ \tilde{\gamma}_1 \\ \tilde{\gamma}_2 \\ \tilde{\gamma}_3 \\ \tilde{\gamma}_4 \\ \tilde{\gamma}_5 \\ \tilde{\gamma}_6 \\ \tilde{\gamma}_7 \end{pmatrix} = \frac{1}{\sqrt{2^3}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & \omega^8 & \omega^4 & \omega^8 & \omega^4 & \omega^8 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & \omega^8 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix} \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \\ \gamma_7 \end{pmatrix}$$

where, in this case

$$\begin{aligned} \omega = e^{2\pi i/(2^3)} &= e^{i\pi/4} \\ &= \sqrt{i} \\ &= \frac{1}{\sqrt{2}}(1+i) \end{aligned}$$

Note that $\omega^8 = 1$, and the matrix elements repeat after reaching a power 8.

Example: The 3-Qbit QFT matrix



$$\omega^1 = \sqrt{i} \qquad \omega^5 = -\sqrt{i}$$
$$\omega^2 = i \qquad \omega^6 = -i$$
$$\omega^3 = i\sqrt{i} \qquad \omega^7 = -i\sqrt{i}$$
$$\omega^4 = -1 \qquad \omega^8 = +1$$

$$\begin{pmatrix} \tilde{\gamma}_0 \\ \tilde{\gamma}_1 \\ \tilde{\gamma}_2 \\ \tilde{\gamma}_3 \\ \tilde{\gamma}_4 \\ \tilde{\gamma}_5 \\ \tilde{\gamma}_6 \\ \tilde{\gamma}_7 \end{pmatrix} = \frac{1}{\sqrt{2^3}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \sqrt{i} & i & i\sqrt{i} & -1 & -\sqrt{i} & -i & -i\sqrt{i} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & i\sqrt{i} & -i & \sqrt{i} & -1 & -i\sqrt{i} & i & -\sqrt{i} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\sqrt{i} & i & -i\sqrt{i} & -1 & \sqrt{i} & -i & i\sqrt{i} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & -i\sqrt{i} & -i & -\sqrt{i} & -1 & i\sqrt{i} & i & \sqrt{i} \end{pmatrix} \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \\ \gamma_7 \end{pmatrix}$$

B. Quantum Fourier Transforms and Period Finding

There are two things left to show.

First, what does a QFT do to a state like

$$|\psi\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle_n$$

and how can we use it to determine the period *r*.

Second, can (really, how can) a QFT be evaluated efficiently (in a number of operations that is polynomial in the number of Qbits – not exponential.

Those are the subjects of the next ScreenCast