Unit 12: Show me your phase, Mr. Unitary

Tzu-Chieh Wei
C. N. Yang Institute for Theoretical Physics and Department of Physics and Astronomy, State University of New York at Stony Brook, Stony Brook, NY 11794-3840, USA
(Dated: September 25, 2024)

In this unit, we discuss quantum Fourier Transform, quantum phase estimation, Shor's factoring algorithm, and quantum linear system (such as the HHL algorithm).

Learning outcomes: You'll be able to understand and apply one of the most important functions: Quantum Fourier Transform and algorithms: Quantum Phase Estimation.

I. INTRODUCTION

Fourier transform and its discrete version are very useful tools in physics and engineers, these are defined as

$$\text{Fourier transform and inverse}: \hat{f}(k) \equiv \int_{-\infty}^{\infty} dx f(x) e^{i2\pi xk}, \quad f(x) \equiv \int_{-\infty}^{\infty} dk \hat{f}(k) e^{-i2\pi xk}.$$

$$\text{Discrete Fourier transform and inverse}: f(x) \longrightarrow \hat{f}(k) \equiv \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{i2\pi kx/N} f(x), \quad \hat{f}(k) \longrightarrow f(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-i2\pi kx/N} \hat{f}(k).$$

In quantum computation, we expect that the latter can be naturally implemented as

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i2\pi jk/N} |k\rangle,$$

where $|j\rangle$ is an integer encoded in the binary representation $|j = j_1 j_2 \dots j_n\rangle$, i.e., $j = \sum_{q=1}^{n} j_q \, 2^{n-q}$, and we also encode $|k = k_1 k_2 \dots k_n\rangle$ similarly. (We note that readers may find different ordering in the IBM Qiskit, where their most significant digit is labeled by j_{n-1} and the least significant digit is j_0.) Therefore, the above discrete Quantum Fourier Transform (QFT, not to be confused with quantum field theory) is written as

$$|j = j_1 j_2 \dots j_n\rangle \longrightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{i2\pi jk/2^n} |k = k_1 k_2 \dots k_n\rangle.$$

Below, we shall see how to implement this with quantum circuits and how QFT can be useful in many quantum algorithms.

II. QFT IN TERMS OF CIRCUITS

The explicit earliest construction of QFT was by Coppersmith [1].

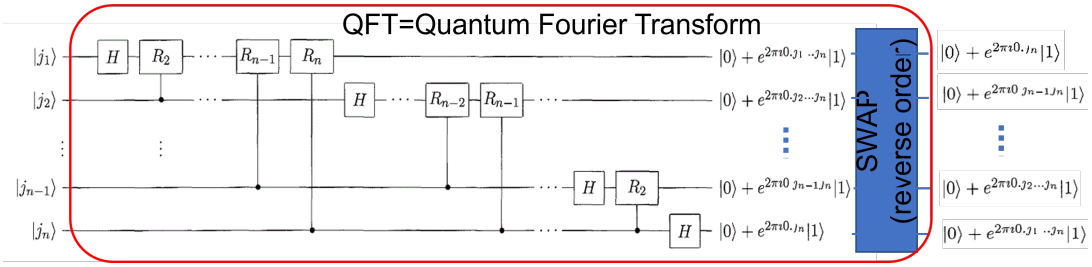


FIG. 1. Illustration of a circuit to implement the quantum Fourier transform.

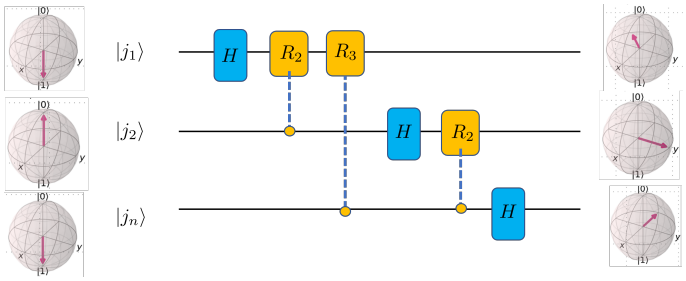


FIG. 2. Illustration of the QFT and Bloch spheres.

$$|\mathrm{QFT}(j)\rangle = \frac{1}{2^{n/2}} \sum_{k's=\{0,1\}} e^{i2\pi \sum_{q,r=1}^{n} j_q 2^{n-q} k_r 2^{n-r}/2^n} |k = k_1 k_2 \dots k_n\rangle,$$

which can be simplified as

$$|\mathrm{QFT}(j)\rangle = \frac{1}{2^{n/2}} \otimes_{r=1}^{n} \Big(\sum_{k_r=\{0,1\}} e^{i2\pi(\sum_{q=0}^{n} j_q k_r 2^{n-q-r})} |k_r\rangle \Big),$$

which gives

$$\frac{1}{2^{n/2}} \otimes_{r=1}^{n} \left(|0\rangle + e^{i2\pi(\sum_{q=n-r+1}^{n} j_q 2^{n-q-r})} |1\rangle \right)_r,$$

where in the exponent we can begin q at $n-r+1$ because for $q <= n-r$ we have $2^{n-q-r} \geq 1$ and thus the phase is equivalent to 1. Moreover $\sum_{q=n-r+1}^{n} j_q 2^{n-q-r} = 0.j_{n-r+1} j_{n-r+2} \dots j_n$. Thus, we conclude that

$$|\mathrm{QFT}(j)\rangle = \frac{1}{2^{n/2}} (|0\rangle + e^{i2\pi 0.j_n}|1\rangle)(|0\rangle + e^{i2\pi 0.j_{n-1}j_n}|1\rangle) \dots (|0\rangle + e^{i2\pi 0.j_1 \dots j_n}|1\rangle).$$

Can this be implemented by elementary one- and two-qubit gates? The answer is yes. In the above, we see phases of the form $e^{i2\pi/2^{q-r+1}}$ and it is convenient to define a phase gate,

$$R^{q-r+1} = R^{[q \to r(q>r)]} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^{q-r+1}} \end{pmatrix},$$

and we will need a controlled version of the phase gate. The complete circuit is shown in Fig. 1. Even if you do not fully understand the deriviation above, you can simply use the circuit diagram as how the input $j = j_1 j_2 \dots j_n$ gets mapped to the phases of the output qubits. It is interesting to note that the phase information is encoded in the digits shifted stepwise as one progresses from one qubit to another. Furthermore, you may have noticed the swap (to reverse the qubit ordering) at the end. Depending on specific applications, it may not be needed as one can adjust the order accordingly. Alternatively, neighboring swaps can be implemented along the way to achieve the reversal of the qubit ordering.

III. QFT AND BLOCH SPHERES AND AN APPLICATION ON ADDING INTEGERS

Given a product state as an input, the output of the QFT is also a product state, and therefore we can visualize the output using Bloch spheres. We show the input spheres of $|101\rangle$ and the output spheres of $(|0\rangle + e^{i2\pi * 0.101_2}|1\rangle) \otimes (|0\rangle + e^{i2\pi * 0.01_2}|1\rangle) \otimes (|0\rangle + e^{i2\pi * 0.1_2}|1\rangle)$.

We note that Qiskit has opposite ordering to most other literatures, which we mentioned in an earlier lecture; basically their qubits are arranged so that the most significant qubit (labelled e.g. q_{n-1}) of n qubits is placed to the left most position and the least significant qubit (labelled q_0): $|q_{n-1}, q_{n-2}, \dots, q_1, q_0\rangle$. However, in the circuit, the top-most qubit is q_0 and the bottom qubit is q_{n-1}. The difference is illustrated in Fig. 3.

Application: adding two integers. Suppose we have to integers j and p to add. What we can do is first perform on QFT on $|j\rangle \to |\tilde{j}\rangle \equiv \mathrm{QFT}|j\rangle$ (e.g. without using the final swap). Then the bits' information of j gets transformed to phases of the qubits. To add p, we can apply phase gates to all qubits, depending on the value of p, e.g. the first qubit will be multiplied by a phase $e^{i2\pi 0.p_0 p_1 \dots p_n}$ to $|1\rangle$; the second qubit by $e^{i2\pi 0.p_1 \dots p_n}$, etc. After this we can perform iQFT, then the qubits represent $|j+p\rangle$ (if there are sufficient number of qubits to encode the integer and there is no overflow).

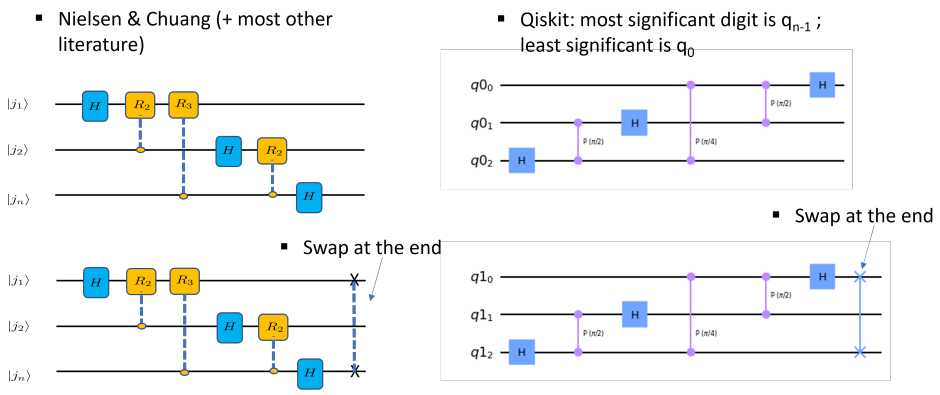


FIG. 3. Illustration of the difference between Qiskit's convention and other literature.

IV. QUANTUM PHASE ESTIMATION (QPE)

When one examines the output of the QFT, one notices that by inverting the QFT, the particular phase encoding results in the binary string $j_1 j_2 \dots j_n$. Suppose that for some problem, we were able to create n qubits in the particular state identical to the output of the QFT, then by inverting the QFT, one can read out the phase information. This is the idea of quantum phase estimation (QPE) [2–4]. How we get can the desired 'QFT' state? The first qubit (after the final swap) has the phase $e^{2\pi i 0.j_n} = e^{2\pi j * 2^{n-1}/2^n}$ and the last one has $e^{2\pi j 2^0/2^n}$.

Assume that we have a unitary U (e.g. it can be the time evolution e^{-iHt}) which has an eigenstate $|u\rangle$ with eigenvalue bing $e^{i2\pi\phi}$, where $\phi = 0.\phi_1\phi_2\dots\phi_n\dots$. Let us consider that it is exactly n-digit, i.e. $\phi = 0.\phi_1\phi_2\dots\phi_n$. Then the phase $e^{2\pi i\phi * 2^{n-1}}$ can be obtained by $U^{2^{n-1}}|u\rangle$ and the other is by $U|u\rangle$. The inverse QFT (iQFT) will give us the digits of ϕ. The phase encoding circuit is shown in Fig. 4, where we use t qubits to encode the phase. There we do not assume that ϕ has exactly t digits and it can be arbitrary. We can write the circuit in the following expression,

$$\sum_{\tau=0}^{2^t-1} |\tau\rangle\langle\tau| \otimes U^\tau = \sum_\tau |\tau\rangle\langle\tau| \otimes e^{-iH\tau},$$

which is a conditional 'evolution', whose time is controlled by the ancillary qubits.

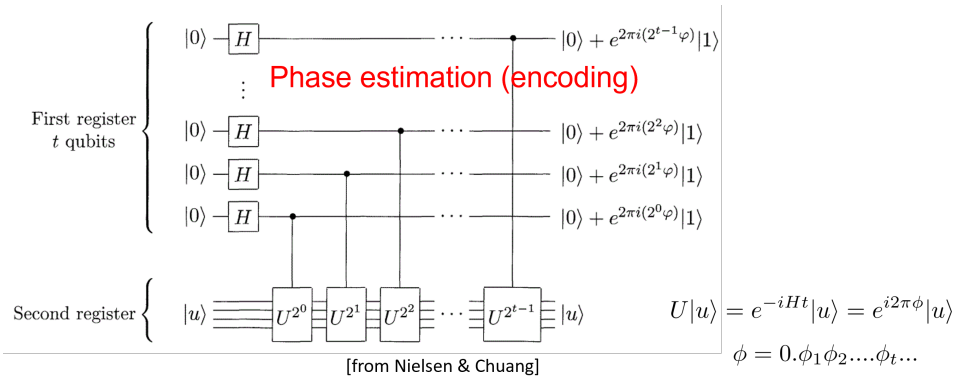


FIG. 4. Illustration of the phase encoding.

If ϕ does not have exactly t bits, then the iQFT can give us wrong results. The phase state after iQFT becomes

$$\text{iQFT}: |\phi\rangle = \frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 0.\phi_t\phi_{t+1}\cdots}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.\phi_{t-1}\phi_t\cdots}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.\phi_1\phi_2\cdots}|1\rangle\right) \to |\tilde{\phi}\rangle = \sum_j \underbrace{\left[\frac{1}{2^t}\sum_k e^{2\pi i k(\phi - j/2^t)}\right]}_{\alpha_j} |j\rangle.$$

The probability of measuring j is $p_j = |\alpha_j|^2$. Naively, it should be peaked around the t-bit approximation of ϕ. This is illustrated in Fig. 5, where we have $\phi = 0.00011000110000101110011_2$ in binary or equally 0.096723759008708_{10} in

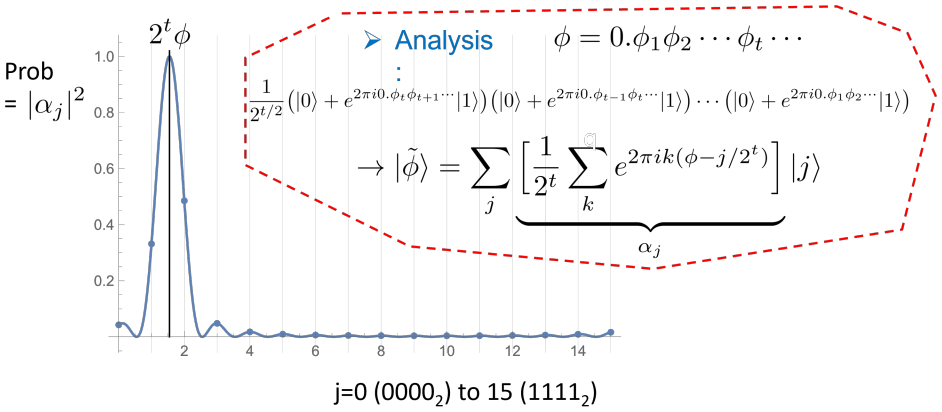


FIG. 5. Illustration of the probability in getting measurement outcomes and the analysis.

demical and $t = 4$. Thus $2^4\phi = 1.100011000010111001_2$, two of the best probable outcomes are (a) 0001_2 (prob= 0.331695) and (b) 0010_2 (prob= 0.48531).

We will not perform the detailed analysis here but refer the readers to the book by Nielsen and Chuang [5]. If one sets the probability of success to be $\geq 1 - \epsilon$ then with t bits of encoding, the measured phase bits are only accurate to the first n_q bits, where

$$n_q = t - \log_2\left(2 + \frac{1}{2\epsilon}\right).$$

Example: QPE on S gate. Let us consider a simple example of S gate,

$$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix},$$

which has two eigenvalues of the form $e^{i2\pi\phi}$, with $\phi = 0$ and $\phi = 1/4 = 0.01_2$ and eigenstates $|0\rangle$ and $|1\rangle$, respectively. Given that the phases can be represented exactly with $t = 2$ qubits and the S gate acts on a single qubit, we will consider a total of 3 qubits. This example is illustrated in Fig. 6.

The $|0\rangle$ eigenstate is associated with a trivial phase and the $|1\rangle$ is associated with a nontrivial phase; the $|1\rangle$ is prepared by applying X to $|0\rangle$. In order to prepare the phase state, we need to implement controlled version of S gate and its power (square in this case): $C - S$ and $C - S^2 = CZ$. The final part of the circuit is the iQFT that reads out the bits of the phase (over 2π to be more precise). Since in this case a two-qubit iQFT is needed, the only two-qubit gate is $C_{R_2^{-1}} = C - S^{-1}$. The single qubits needed are all Hadamard gates H. We can run this on Qiskit Qasm simulators or real devices; we recommend the readers to try themselves.

We display an example code for the $|1\rangle$ eigenstate case:

```
q = QuantumRegister(3)
c = ClassicalRegister(2)
qc = QuantumCircuit(q, c)

# initialize |+> and |+>, |1>
qc.h(q[0])
qc.h(q[1])
qc.x(q[2])

# circuit to generate the phase (of S operator) on qubit 1
qc.cu1(np.pi/2,q[1],q[2])

# circuit to generate the phase (of S operator) on qubit 0
qc.cz(q[0],q[2])

qc.barrier()
#  inverse Quantum Fourier Transform is the Hadamard gate
qc.h(q[0])
```

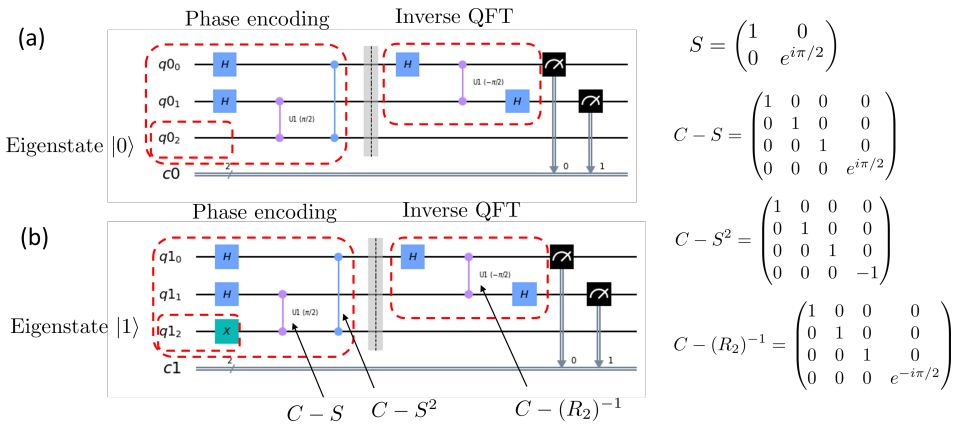


FIG. 6. Illustration of the QPE for the S gate.

```
qc.cu1(-np.pi/2,q[0],q[1])
qc.h(q[1])

#  measure qubit 0 & 1
qc.measure([0,1],[0,1])
```

Given the importance of QFT, it is already implemented in Qiskit and you import it in Python as follows,

```
 from qiskit.circuit.library import QFT
 from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
qr_data=QuantumRegister(3)
qc=QuantumCircuit(qr_data)

qft=QFT(num_qubits=3, approximation_degree=0, do_swaps=False, inverse=False, insert_barriers=False,
    name='Qft').decompose()
qc.append(qft,qr_data)
qc.decompose().draw('mpl')
```

The above code is used to generate one of the circuits (top right panel) in Fig. 3.

V. SEMICLASSICAL QFT

If QFT or iQFT is the last step, then controlled phase gates can be replaced by 0/1 measurement followed by classical controlled phase gates, which was proposed by Niu and Griffiths, PRL '96 [6]. This can be easily seen from Fig. 1, and let us omit the swap and consider the iQFT by reversing (and changing all the signs of the phase angles) to read out j_n, j_{n-1}, etc. But one can use the classically controlled gates to replace all controlled phase gates. It can be as follows. One first measures qubit n and obtains a value j_n and all the reversed controlled phase gates controlled by this qubit become classically controlled phase gates (depending on whether the outcome is 1 or not). Then one measures $(n-1)$-th qubit to read out j_{n-1} and replaces all controlled-phase gates by the classically controlled ones. By continuing this until qubit 1, we can read out j_1.

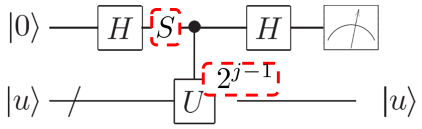
VI. APPLICATION: APPROXIMATE PROJECTION TO EIGENSTATES

When we discuss the QPE, we assume that the input register is an eigenstate $|u\rangle$, but in general we may not have the eigenstate and have a superposition, instead. The QPE (via iQFT) can approximately project the system to some eigenstate: $|\psi\rangle = \sum_i a_i|u_i\rangle$, i.e. project to approximately $|u_i\rangle$ with probability approximately $|a_i|^2$. We use 'approximately' because the phase in general is not exactly represented by the finite number of qubits and QPE is done approximately.

VII. KITAEV ITERATIVE QPE AND OTHER ITERATIVE SCHEMES

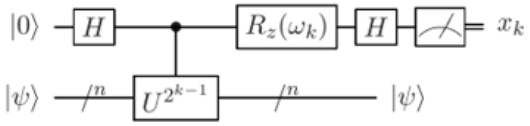
Kitaev iterative QPE. Kitaev describes in his textbook an approach of QPE using one ancillary qubit at a time (in sequence). The idea of the diagram is shown here

(Kitaev's algorithm) Consider the quantum circuit



where we also insert the S in the red-dashed box and the unitary U can be raised to a power of 2^{j-1}. Without S, this Hadamard-like test gives $p_0 - p_1 = \cos(2\pi\phi 2^{j-1})$ and with S, it gives $\sin(2\pi\phi 2^{j-1})$. Note that if $U_a|y\rangle = |ay \bmod N\rangle$, then $(U_a)^p = U_{a^p} = U_{(a^p \bmod N)}$, so its exponential can be simplified. Basically, by knowing the cosine and sine of $0.\phi_j\phi_{j+1}\dots$ (for $j = 1$ to n) precisely enough (by repeating experiments) then we can dedude ϕ to certain accuracy.

Other iterative QPE. Another iterative one is given by the following picture,



In some sense, this iterative quantum phase estimation [Dobsicek, Johansson, Shumeiko, Wendin, PRA '07] can be regarded as based on the semi-classical inverse QFT. One proceeds by reading out the most significant digit (with certain probability of success) ϕ_n (from the phase $0.\phi_n$) and using this result to read out ϕ_{n-1} from $0.\phi_{n-1}\phi_n$ by first cancelling out ϕ_n before reading out ϕ_{n-1} by the Hadamard gate. One continues this until ϕ_0 is read out.

VIII. APPLICATION: SHOR'S FACTORING ALGORITHM

In this section, we will discuss Shor's factoring algorithm. To explain it we find it useful to first the period finding and the particular problem is the modular exponentiation.

Period finding in the modular exponentiation. Given positive integers x and N ($x < N$) with no common factors, the period finding problem is find the least integer r such that

$$x^r = 1 \,(\mathrm{mod}\, N).$$

We will define an unitary operator that implements the modular exponentiation,

$$\begin{cases} U|y\rangle \equiv |xy \,(\mathrm{mod}\, N)\rangle, \text{ for } y \in \{0,1\}^L, \text{ and } y < N; \\ U|y\rangle \equiv |y\rangle, \quad \text{for } N \le y < 2^L. \end{cases}$$

It can be verified by direct substituion that the eigenstates of U are of the following form (labeled by the interger $s \in [0, r-1]$)

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi i s k}{r}} |x^k \bmod N\rangle, \quad U|u_s\rangle = e^{\frac{2\pi i s}{r}} |u_s\rangle.$$

From a previous section of the QPE, we can estimate the phase, or in particular s/r to certain accuracy (using a sufficient number of qubits to encode the fraction).

However, we have a problem here, as we cannot create the eigenstate. Luckily, we can create a certain superposition of all the eigenstates,

$$|\text{`}1'\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle,$$

which is the binary representation of $1 = 00\ldots01_2$ that can be easily prepared. Then the QPE procedure outputs randomly an approximation of s_i/r, with different s_i's each time we carry out the QPE. We leave the details of how to employ the 'continued fraction' method to find s_i to the book by Nielsen & Chuang [5].

We aim to achieve the accuracy to $2L+1$ bits so that $2^{-2L-1} < 1/(2r^2)$ and thus need to take

$$t = 2L + 1 + \log_2\Big(2 + \frac{1}{2\epsilon}\Big).$$

With the above accuracy we can deduce a 'r' and check whether it's a correct answer (repeat if necessary). Note that there is an improved error analysis by Chapell et al. [7].

Shor's factoring algorithm. Can we factorize the following number?

$$\begin{aligned} N = \ & 18070820886874048059516561644059055662781025167694013491701270214 \\ & 50056662540244048387341127590812303371781887966563182013214880557. \end{aligned}$$

It is known that primality test is polynomial time, so it is relatively easy to check that the above number is not a prime number. But to factorize it using current classical computers will take almost an exponential time in the number digits. However, Shor's quantum algorithm takes in principle a time cubic in the number of digits, making factorization a relatively simple task for quantum computers.

Let us now explain this algorithm. We can define a function that we discuss earlier,

$$F_{x,N}(a) := x^a \bmod N,$$

and we can find the period r using the QPE so that $x^r \equiv 1$. With this we can explain the method behind Shor's algorithm. An important fact to notice is that if $gcd(x, N) = 1$ and that the period r of $F_{x,N}(a)$ is even, we have

$$(x^{r/2}+1)(x^{r/2}-1) = x^r - 1 = 0 \,(\mathrm{mod}\, N).$$

This means that $(x^{r/2}+1)(x^{r/2}-1)$ is divisible by N and either of $(x^{r/2} \pm 1)$ has a nonzero common factor with N.

Example. For $N = 15$, we can choose $x \in X = \{2, 4, 7, 8, 11, 13, 14\}$ and the corresponding periods are $r \in R = \{4, 2, 4, 4, 2, 4, 2\}$. If we take $x = 11$, $z = x^{r/2} = 11$, then we find that $gcd(11+1, 15) = 3$ and $gcd(11-1, 15) = 5$, thus we have the factorization $15 = 3 \times 5$.

Shor's algorithm is thus using quantum computers to find the period r. We will not analyze the success probability (see Nielsen & Chuang); it suffices to note that it is finite and by repeating the procedure a few times, one can find a factor with a high probability.

By the way the two factors of the above number are

$$\begin{aligned} a &= 39685999459597454290161126162883786067576449112810064832555157243 \\ b &= 45534498646735972188403686897274408864356301263205069600999044599. \end{aligned}$$

You can easily check that these are two factors by multiplying them together to see if the result agrees with the above number, i.e. whether $N = a \times b$. This also illustrates that factoring is a problem in NP, where it may not be easy to find the answers, but given the answers, it is easy to check.

Update. In 2023, Oded Regev proposed another quantum algorithm to perform factoring [8], which turns out to be faster than Shor's algorithm, i.e., with gate complexity $O(n^{3/2})$ (running $\sqrt{n}+4$ times), where n is the number of bits of the integer.

IX. APPLICATION: DISCRETE LOGARITHM

The goal of the discrete logarithm is: for two positive integers a and b we need to find the positive integer s such that

$$b = a^s \bmod N.$$

We can define a two-argument function,

$$f(x_1, x_2) = a^{sx_1+x_2} \bmod N.$$

It is obvious that the function has the following (periodic) property,

$$f(x_1 + q, x_2 - q\,s) = f(x_1, x_2).$$

This means that the problem can be solved efficiently by a quantum algorithm [9] using both the phase estimation/order or period finding that we have learned above. Its double-argument Fourier transform (in range $[0, r-1]$) is

$$|\hat{f}(q_1, q_2)\rangle = \frac{1}{r\sqrt{r}} \sum_{x_1=0}^{r-1} \sum_{x_2=0}^{r-1} e^{-2\pi i(q_1 x_1 + q_2 x_2)/r} |f(x_1, x_2)\rangle.$$

We thus have

$$|\hat{f}(q_1, q_2)\rangle \overset{\text{requires}}{\underset{\text{proof}}{=\!=}} \begin{cases} \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i q_2 j/r} |f(0, j)\rangle, & \text{if } q_1 - s q_2 \text{ multiple of } r \\ 0 \text{ otherwise} \end{cases}$$

We can use two quantum registers, both with enough number of qubits, to perform phase encoding and then iQFT to read out the two periods sl/r and l/r to deduce s.

X. APPLICATION: QUANTUM LINEAR SYSTEM ALGORITHM (A.K.A. HHL ALGORITHM)

The idea is to the solve following linear equation by a quantum computer,

$$A\vec{x} = b, \quad \vec{x} = A^{-1} b,$$

which was considered by Harrow, Hassidim, Lloyd in a PRL paper in 2008 (hence the name HHL algorithm) [10]. There have also been a few improvements on the original scheme.

For simplicity, we assume A is an $N \times N$ Hermitian matrix and

$$A|u_j\rangle = \lambda_j |u_j\rangle.$$

The idea is to obtain the inverse, for each eigenspace (with nonzero eigenvalue $\lambda_j \neq 0$), we want λ_j^{-1}. We encode the vector b as

$$|b\rangle = \sum_{i=1}^{N} b_i |i\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle.$$

First, we prepare a state

$$|\Psi_0\rangle = \frac{1}{\sqrt{T}} \sum_{\tau=0}^{T} |\tau\rangle \otimes |b\rangle = |\text{all } \tau\rangle \otimes \sum_{j=1}^{N} \beta_j |u_j\rangle.$$

Similar to what we have discussed in the phase encoding above, we define

$$\text{c} - \text{A} \equiv \sum_{\tau} |\tau\rangle\langle\tau| \otimes e^{iAt_0\tau}.$$

We apply this to $|\Psi_0\rangle$,

$$|\Psi_0\rangle \to |\Psi_1\rangle = \frac{1}{\sqrt{T}} \sum_{\tau=0}^{T} |\tau\rangle \otimes \sum_{j=1}^{N} \beta_j e^{i\lambda_j t_0 \tau} |u_j\rangle.$$

Then we apply the inverse QFT to first register for phase estimation,

$$|\Psi_1\rangle \to |\Psi_2\rangle = \sum_{j} \beta_j |\tilde{\lambda_j}\rangle \otimes |u_j\rangle.$$

Next, how to apply the inverse of A? specifically we do we divide in the above each term by λ_j ? If we could, then

$$\sum_j \beta_j/\lambda_j|\tilde{\lambda_j}\rangle \otimes |u_j\rangle \xrightarrow{\text{undo QPE}} |\text{all } \tau\rangle \otimes \sum_j \beta_j/\lambda_j|u_j\rangle?$$

Unfortunately, the application of inverse cannot be done with unit probability. We attach an ancillary qubit in $|0\rangle$, then apply a U gate controlled by the first register:

$$\sum_j \beta_j|\tilde{\lambda_j}\rangle \otimes |u_j\rangle \otimes |0\rangle \to \sum_j \beta_j|\tilde{\lambda_j}\rangle \otimes |u_j\rangle \otimes \left(\sqrt{1-\frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\right).$$

The we undo the QPE and arrive at

$$|\text{all } \tau\rangle \otimes \sum_j \beta_j|u_j\rangle \otimes \left(\sqrt{1-\frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\right).$$

The inversion is successful only when the ancilla measurement gives 1. Note that in the above, the constant C needs to be chosen so that $\lambda_j < C$ for all j. In the case of success, we obtain

$$|\text{all } \tau\rangle \otimes \sum_j \beta_j \frac{C}{\lambda_j}|u_j\rangle \otimes |1\rangle = |\text{all } \tau\rangle \otimes A^{-1}|b\rangle \otimes |1\rangle.$$

Note that despite that the solution $|\vec{x}\rangle$ is obtained in a quantum, a key question is whether such a form of solution leads to any practical speedup? There have been discussions on this and further improvements to the HHL algorithm.

A. Variational approach for quantum linear systems

to add later ...

XI. CONCLUDING REMARKS

In this unit, we have discussed quantum Fourier Transform, quantum phase estimation, Shor's factoring algorithm, and quantum linear system (such as the HHL algorithm).

Learning outcomes: It is a good time to check whether you have achieved the following Learning Outcomes: After this Unit, You'll be able to understand and apply one of the most important functions: Quantum Fourier Transform and algorithms: Quantum Phase Estimation.

Suggested reading: N&C chap 5, 6.3; KLM chapter 7, 8.4; Qb chap 3.8, 3.9, 3.11.

[1] D. Coppersmith, An approximate fourier transform useful in quantum factoring, arXiv preprint quant-ph/0201067 (2002).
[2] A. Y. Kitaev, Quantum measurements and the abelian stabilizer problem, arXiv preprint quant-ph/9511026 (1995).
[3] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Quantum algorithms revisited, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 339 (1998).
[4] D. S. Abrams and S. Lloyd, Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors, Phys. Rev. Lett. **83**, 5162 (1999).
[5] M. A. Nielsen and I. Chuang, Quantum computation and quantum information (2002).
[6] R. B. Griffiths and C.-S. Niu, Semiclassical fourier transform for quantum computation, Physical Review Letters **76**, 3228 (1996).
[7] J. M. Chappell, M. A. Lohe, L. Von Smekal, A. Iqbal, and D. Abbott, A precise error bound for quantum phase estimation, Plos one **6**, e19663 (2011).
[8] O. Regev, An efficient quantum factoring algorithm, arXiv preprint arXiv:2308.06572 (2023).
[9] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th annual symposium on foundations of computer science* (Ieee, 1994) pp. 124–134.
[10] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. **103**, 150502 (2009).