CPEN 400Q Lecture 15
Quantum phase estimation; order finding

Monday 4 March 2024

Announcements

- Quiz 6 today
- Technical assignment 3 available later this week
- Midterm checkpoint due Wednesday; meetings on Thurs/Fri

We implemented the quantum Fourier transform using a *polynomial* number of gates:

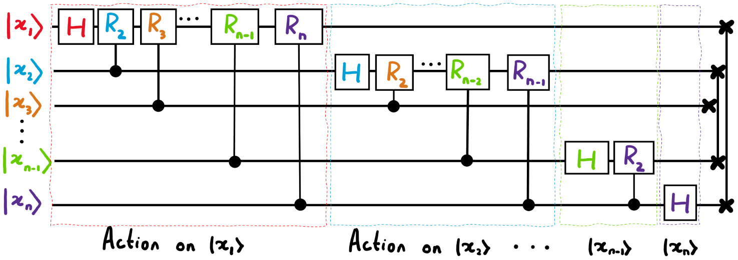


Image credit: Xanadu Quantum Codebook node F.3

We started learning about the quantum phase estimation subroutine which estimates the eigenvalues of unitary matrices.

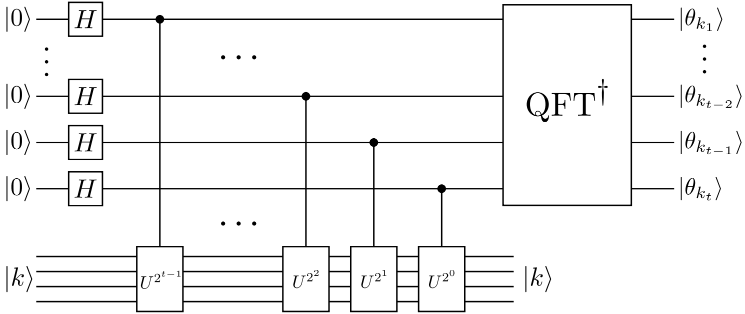
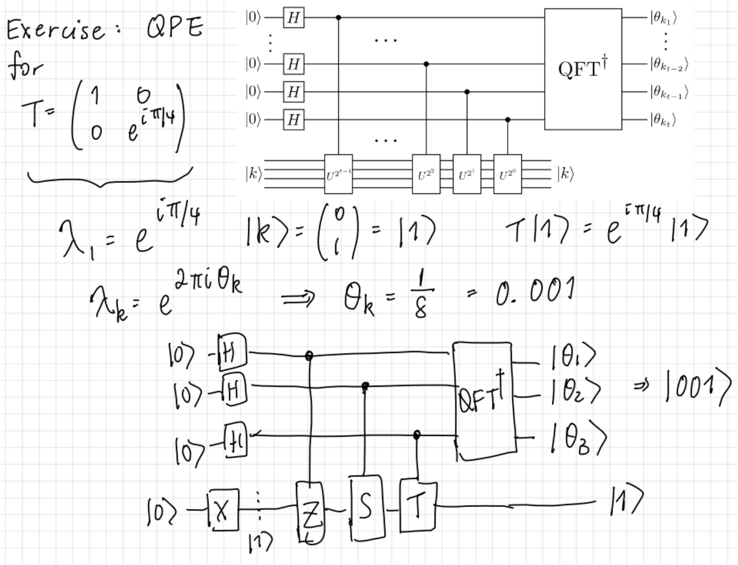


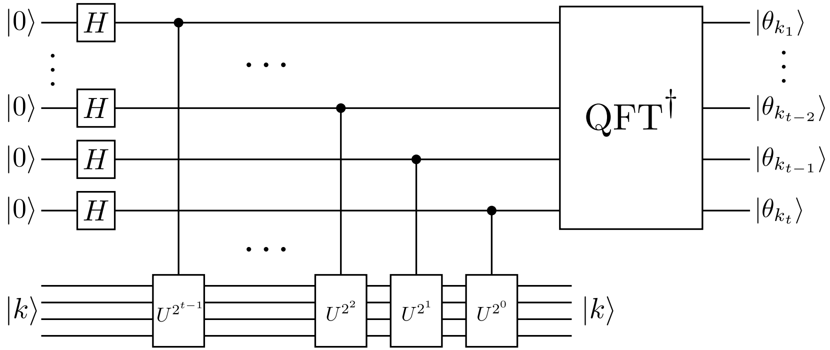
Image credit: Xanadu Quantum Codebook node P.2

We walked through an example using T.

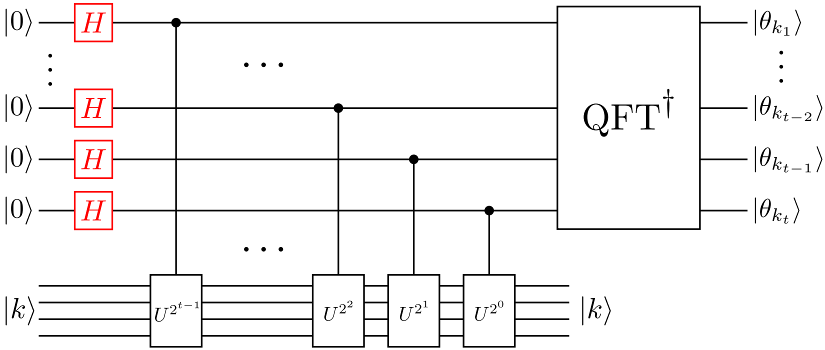


Learning outcomes

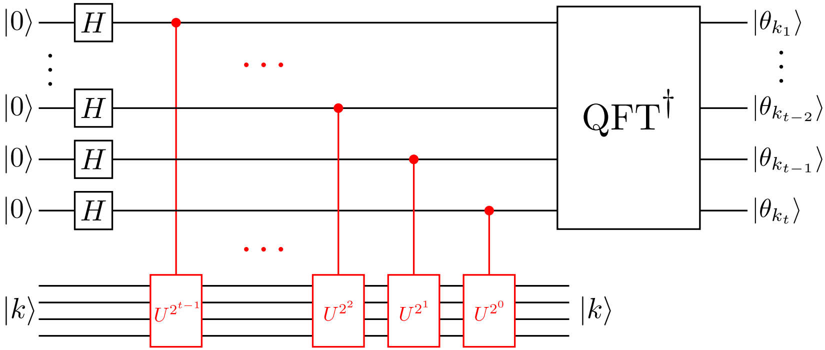
- Outline the steps of the quantum phase estimation (QPE) subroutine
- Use QPE to implement the order finding algorithm
- Implement Shor's algorithm in PennyLane



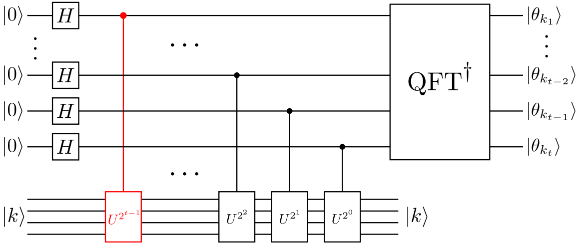
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
H
H
H
H
$|k\rangle$
$U^{2^{t-1}}$
U^{2^2}
U^{2^1}
U^{2^0}
$|k\rangle$
$\mathrm{QFT}^\dagger$
$|\theta_{k_1}\rangle$
$|\theta_{k_{t-2}}\rangle$
$|\theta_{k_{t-1}}\rangle$
$|\theta_{k_t}\rangle$

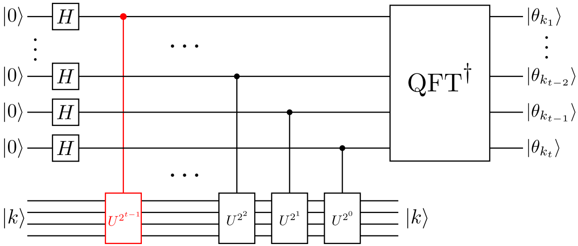

$|0\rangle$
H
$|0\rangle$
H
$|0\rangle$
H
$|0\rangle$
H
$|k\rangle$
$U^{2^{t-1}}$
U^{2^2}
U^{2^1}
U^{2^0}
$|k\rangle$
$\mathrm{QFT}^\dagger$
$|\theta_{k_1}\rangle$
$|\theta_{k_{t-2}}\rangle$
$|\theta_{k_{t-1}}\rangle$
$|\theta_{k_t}\rangle$

Quantum phase estimation: step 1



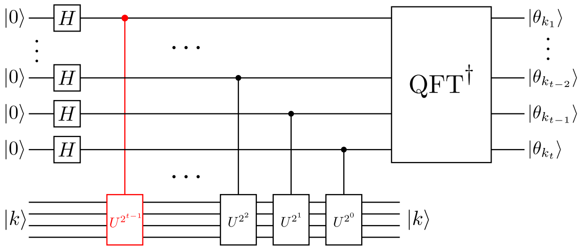
Quantum phase estimation: step 2





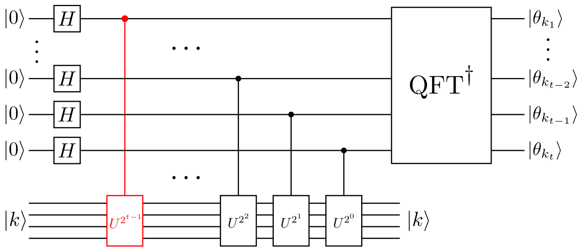
Use phase kickback

Quantum phase estimation: step 2

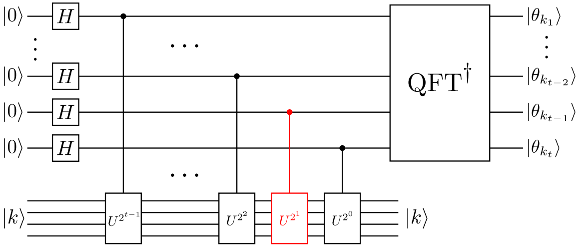


What is happening in the exponent?

Quantum phase estimation: step 2

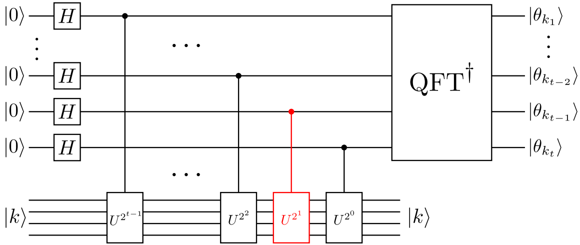


Quantum phase estimation: step 2

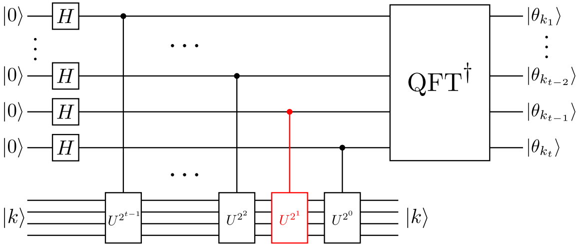


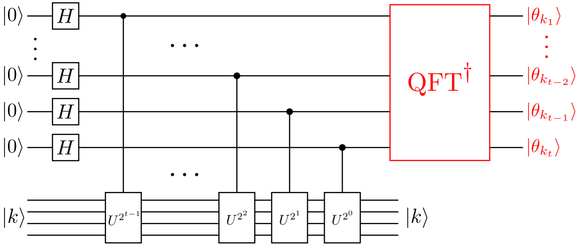
Check second-last qubit (ignore the others)

Quantum phase estimation: step 2

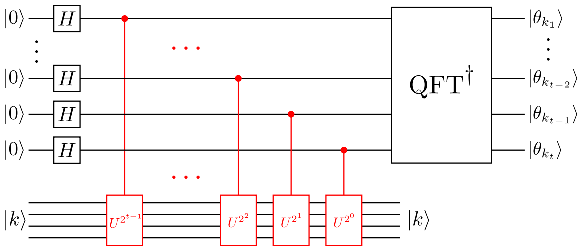


Again check the exponent...


|0⟩
H
|0⟩
H
|0⟩
H
|0⟩
H
|k⟩
$U^{2^{t-1}}$
U^{2^2}
U^{2^1}
U^{2^0}
|k⟩
QFT†
$|\theta_{k_1}\rangle$
$|\theta_{k_{t-2}}\rangle$
$|\theta_{k_{t-1}}\rangle$
$|\theta_{k_t}\rangle$



Can show in the same way for the last qubit (ignore others)

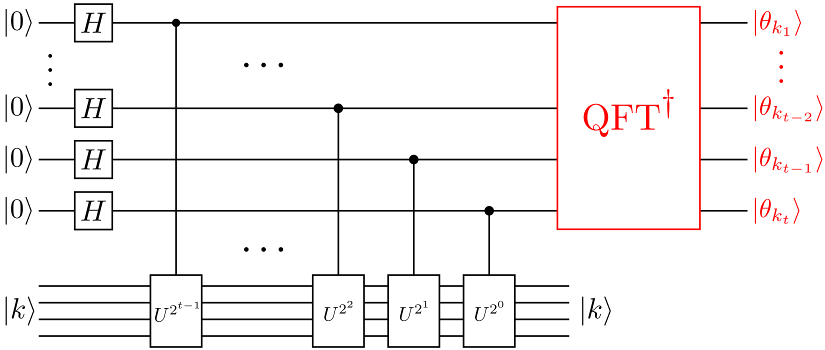


After step 2, we have the state

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.\theta_{k_t}}|1\rangle) \cdots \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.\theta_{k_2}\cdots\theta_{k_t}}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.\theta_{k_1}\cdots\theta_{k_t}}|1\rangle)|k\rangle$$

Should look familiar!

Measure to learn the bits of θ_k.



Let's implement it.

Reminder: where are we going?

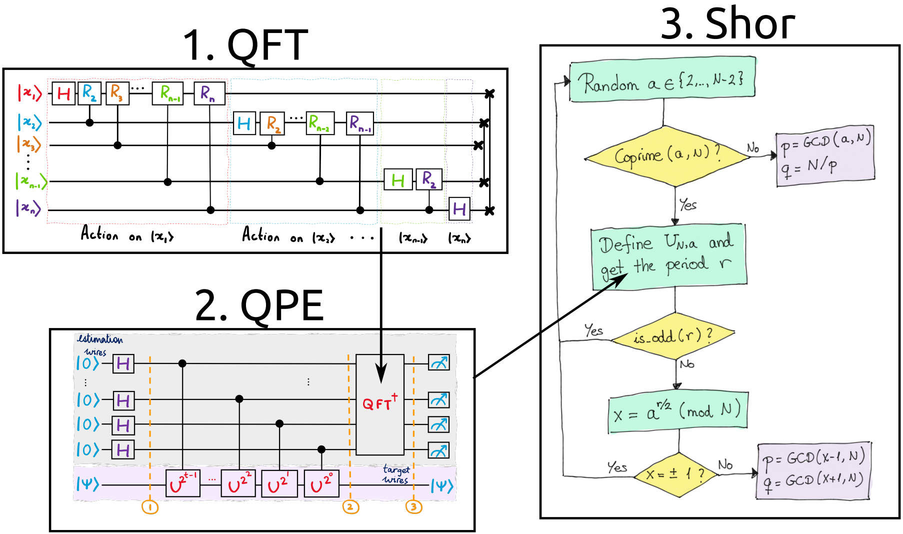


Image credit: Xanadu Quantum Codebook nodes F.3, P.3, S.4

Order finding on a quantum computer

Suppose we have a function

over the integers modulo N.

If there exists $r \in \mathbb{Z}$ s.t.

$f(x)$ is periodic with period r.

Order finding on a quantum computer

Suppose

The *order* of a is the smallest m such that

Note that this is also the period:

Order finding on a quantum computer

Exercise: find the order of $a = 5$ for $N = 7$.

Order finding on a quantum computer

More formally, define

Define a unitary operation that performs

If m is the order of a, and we apply $U_{N,a}$ m times,

So m is also the order of $U_{N,a}$! We can find it efficiently using a quantum computer.

Order finding on a quantum computer

Let U be an operator and $|\phi\rangle$ any state. How do we find the minimum r such that

QPE does the trick if we set things up in a clever way:

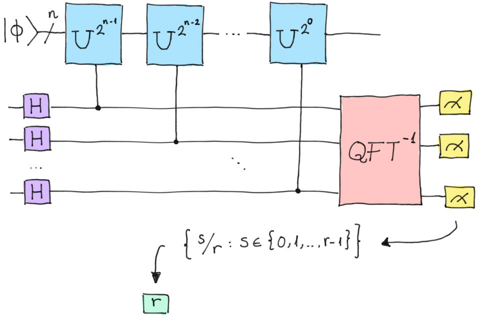


Image credit: Xanadu Quantum Codebook node S.3

Consider the state

If we apply U to this:

Now consider the state

If we apply U to this:

This generalizes to $|\Psi_s\rangle$

It has eigenvalue

Idea: if we can create *any* one of these $|\Psi_s\rangle$, we could run QPE and get an estimate for s/r, and then recover r.

Problem: to construct any $|\Psi_s\rangle$, we would need to know r in advance!

Solution: construct the uniform superposition of all of them.

But what does this equal?

The superposition of all $|\Psi_s\rangle$ is just our original state $|\phi\rangle$!

$$|\Psi\rangle = \frac{1}{\sqrt{r}} \cdot \left(|\Psi_0\rangle + |\Psi_1\rangle + \cdots + |\Psi_{r-1}\rangle \right) = \frac{1}{\sqrt{r}} \cdot \begin{array}{l} \frac{1}{\sqrt{r}}\left(|\phi\rangle + 1 \cdot U|\phi\rangle + \cdots + 1 \cdot U^{r-1}|\phi\rangle\right) \\ + \frac{1}{\sqrt{r}}\left(|\phi\rangle + e^{\frac{-2\pi i}{r}} U|\phi\rangle + \cdots + e^{\frac{-2\pi i (r-1)}{r}} U^{r-1}|\phi\rangle\right) \\ \quad \vdots \\ + \frac{1}{\sqrt{r}}\left(|\phi\rangle + e^{\frac{-2\pi i (r-1)}{r}} U|\phi\rangle + \cdots + e^{\frac{-2\pi i (r-1)^2}{r}} U^{r-1}|\phi\rangle\right) \end{array}$$

r 0 .. 0 .. 0

$$= \frac{1}{\sqrt{r}} \cdot \frac{1}{\sqrt{r}} \cdot r\,|\phi\rangle = \boxed{|\phi\rangle}$$

If we run QPE, the output will be s/r for one of these states.

Image credit: Xanadu Quantum Codebook node S.3

Shor's algorithm

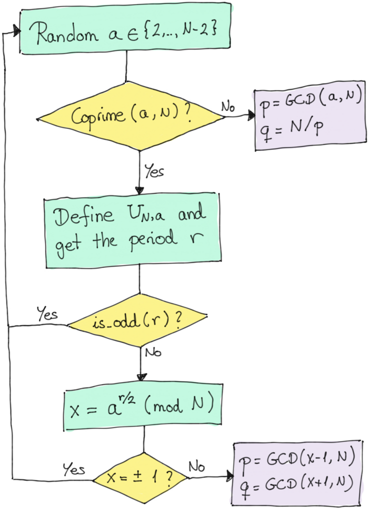


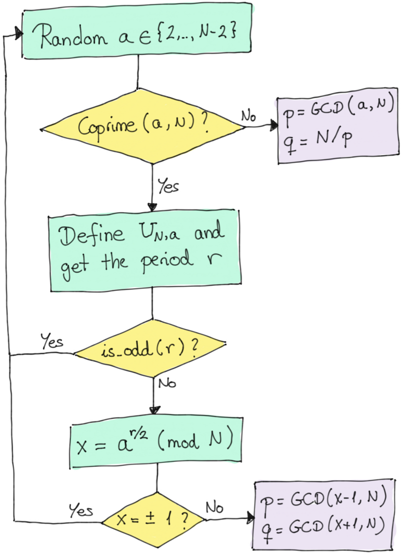
Image credit: Xanadu Quantum Codebook node S.4

Shor's algorithm is used to factor some number N into

where p and q are prime.

A quantum computer runs order finding, and the result is used to obtain p and q.

The rest of the algorithm is classical.



Non-trivial square roots

Idea: find a *non-trivial square root* of N, i.e., some $x \neq \pm 1$ s.t.

If we find such an x, then we know

This means that

for some integer k.

Non-trivial square roots

If

then $x - 1$ is a multiple of one of p or q, and $x + 1$ is a multiple of the other. If

we can compute the values of p and q by finding their *gcd* with N:

But... how do we find such an x?

Non-trivial square roots and factoring

It's actually okay to find any *even* power of x for which this holds:

We can use order finding to find such an r, and it is an even number, then we can find an x and factor N.

Shor's algorithm

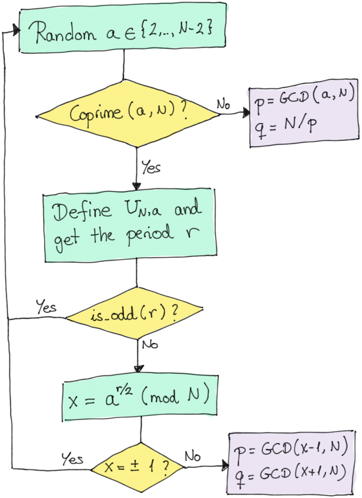
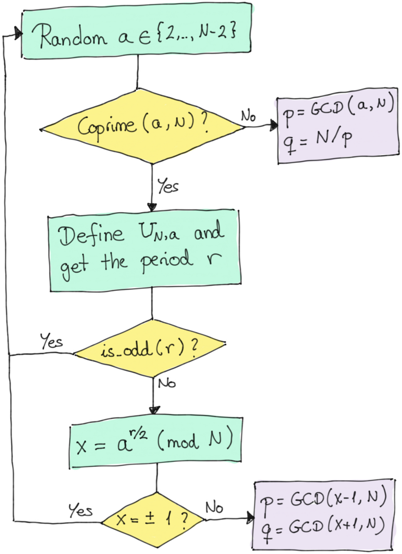


Image credit: Xanadu Quantum Codebook node S.4

Is this really efficient?

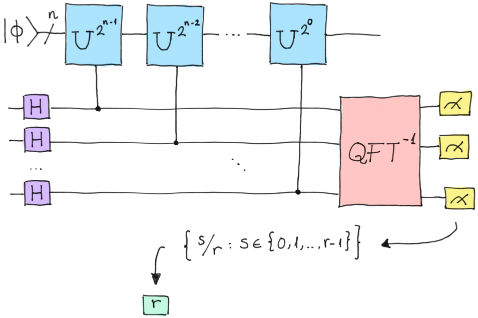
GCD: polynomial w/Euclid's algorithm

Modular exponentiation: can use exponentiation by squaring, other methods to reduce number of operations and memory required



Is this really efficient?

Quantum part: let $L = \lceil \log_2 N \rceil$.



QFT: polynomial in number of qubits $O(L^2)$

Controlled-U gates: implemented using something called *modular exponentiation* in $O(L^3)$ gates.

Next time

Content:

- Hands-on with quantum key distribution

Action items:

1. Midterm checkpoint submission

Recommended reading:

- Codebook modules F, P, and S
- Nielsen & Chuang 5.3, Appendix A.5