Simulating Quantum Time Evolution

Before we are ready to describe the phase estimation algorithm, we consider another natural physics problem. Given a local Hamiltonian H and an initial state $|\psi_0\rangle$, solve for the dynamical evolution by producing the state

$$|\psi_t\rangle = e^{-itH}|\psi_0\rangle$$

A quantum computer is a controllable device for implementing desired unitary evolutions, so the problem of simulating quantum many-body dynamics is in some sense the simplest one it could solve.

The challenge is that a quantum computer evolves according to gates that are discrete and local:

$$|\phi_T\rangle = U_T \ldots U_1 |0^n\rangle$$

where each unitary gate U_i acts on at most k qubits. (we usually take k = 2 to make life easier for experimentalist, but any constant k > 1 is asymptotically equivalent).

This model of quantum computing contrasts with the dynamical simulation problem, since the latter involves time evolution that is continuous, and more importantly it is non-local (because H acts on the entire system).

Simulating Quantum Time Evolution

The dynamical simulation problem for local Hamiltonians was solved by Seth Lloyd in 1996. The paper is called “Universal Quantum Simulators.”

Abstract: “Feynman's 1982 conjecture, that quantum computers can be programmed to simulate any local quantum system, is shown to be correct.”

Lloyds idea is based on decomposing the time evolution into many short steps:

$$e^{itH} = \left(e^{itH/L}\right)^L = e^{itH/L} e^{itH/L} \dots e^{itH/L}$$

And then applying the **Suzuki-Trotter formula** to decompose each of the short steps $e^{itH/L}$ into local unitary gates.

Simulating Quantum Time Evolution

Before describing the Suzuki-Trotter formula, consider time evolution with a local Hamiltonian in which all of the terms pairwise commute (a “commuting Hamiltonian”),

$$H = \sum_{a=1}^{m} H_a \quad , \quad [H_a, H_b] = 0$$

The unitary evolution generated by this Hamiltonian can be decomposed because the terms commute:

$$e^{itH} = e^{it\sum_{a=1}^{m} H_a} = \prod_{a=1}^{m} e^{itH_a} = e^{itH_m} \ldots e^{itH_1}$$

Now given an arbitrary Hamiltonian $H = \sum_{a=1}^{m} H_a$ we can always write *H* as a sum of at most *m* commuting Hamiltonians, called commuting layers. Note that $m = \mathrm{poly}(n)$ for a general k-local Hamiltonian.

Simulating Quantum Time Evolution

In 1875, Sophus Lie proved that for any real or complex finite dimensional matrices A, B, we have

$$\lim_{L\to\infty}\left(e^{A/L}e^{B/L}\right)^{L}=e^{A+B}$$

This Lie product formula was generalized by Trotter in 1959 to include certain infinite dimensional linear operators.

A finite approximation to this limiting result was obtained by Masuo Suzuki in 1976, and appears in his classic papers “Relationship between *d*-Dimensional Quantal Spin Systems and (*d*+1)-Dimensional Ising Systems” , and “Monte Carlo Simulation of Quantum Spin Systems.” Suzuki writes:

$$\|\exp(\sum_{j=1}^{p}A_j)-\{e^{A_1/n}e^{A_2/n}\cdots e^{A_p/n}\}^n\|\leq\frac{2}{n}(\sum_{j=1}^{p}\|A_j\|)^2\exp\left(\frac{n+2}{n}\sum_{j=1}^{p}\|A_j\|\right),$$

Simulating Quantum Time Evolution

In practice, Suzuki's formula allows us to make the following replacement for sufficiently large Trotter number L:

$$e^{(A+B)/L} = e^{A/L}e^{B/L} + \mathcal{O}(L^{-2})$$

If we decompose the Hamiltonian into G commuting layers, $H = H^{(1)} + H^{(2)} + ... + H^{(G)}$, then

$$e^{itH/L} = e^{itH^{(1)}/L}...e^{itH^{(G)}/L} + \mathcal{O}\left(GL^{-2}\right)$$

Replacing the desired time evolution with $L \cdot t$ copies of this "Trotterized" time evolution will produce the desired state $|\psi(t)\rangle$ up to an error that scales like $\mathcal{O}(GtL^{-1})$, from which one can work out the value of L that suffices to approximate the desired state up to any reasonable error. To achieve error less than δ we need,

$$GtL^{-1} \leq \delta$$

Therefore the Trotter number should scale as $L \sim Gt\delta^{-1}$, which scales polynomially with the number of qubits as long as we only require the error in the output states to decrease polynomially.

Simulating Quantum Time Evolution

In 2003, Lloyd's result for k-local Hamiltonians was generalized by Aharanov and Ta-Shma to sparse Hamiltonians, which are D x D Hermitian matrices with at most poly(log D) non-zero efficiently computable entries per row.

There the result is that the unitary evolution generated by such sparse Hamiltonians can be simulated on a quantum computer in time poly(log D). All local Hamiltonians are sparse, but not all sparse Hamiltonians are local.

Simulating quantum dynamics is such an important task, that subsequent improvements have focused on optimizing the run-time to be nearly linear in t, and achieving exponentially small error in polynomial time.

The ability to simulate time evolution for arbitrary local Hamiltonians is one of the main reason for considering them (instead of restricting to more "physical" classes of Hamiltonians).

Even if H is k-local for k = 1000, and your quantum computer only allows nearest-neighbor unitary gates in 1-dimension, it can eventually simulate the time evolution of a physical system governed by H.

Quantum Fourier Transform

Before we can describe the phase estimation algorithm that lets us estimate the energy of a state with respect to a local Hamiltonian, we need an important quantum subroutine called the quantum Fourier transform (QFT).

For this discussion, we think of the computational basis as labeled by integers $|0\rangle, |1\rangle, ..., |N-1\rangle$, where $N = 2^n$.

The (classical) Fourier transform maps a complex vector $(x_0, ..., x_{N-1})$ to $(y_0, ..., y_{N-1})$, with

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk} \quad , \quad \omega_N = e^{-2\pi i j k / N}$$

The Quantum Fourier transform applies the same transformation to the components of a quantum state. Like any unitary transformation, it is described by its action on the computational basis vectors:

$$F_N |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle \quad , \quad j = 0, ..., N-1$$

Quantum Fourier Transform

Similarly, the inverse Fourier transform maps a complex vector $(x_0, ..., x_{N-1})$ to $(y_0, ..., y_{N-1})$, with

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{-jk} \quad , \quad \omega_N = e^{-2\pi i jk/N}$$

The inverse QFT is then given by: $F_N^{-1}|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{-jk}|k\rangle \quad , \quad j = 0, ..., N-1$

$$F_N^{-1} F_N |j\rangle = F_N^{-1} \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle \right) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} \left(F_N^{-1} |k\rangle \right) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} \left(\frac{1}{\sqrt{N}} \sum_{k'=0}^{N-1} \omega_N^{-kk'} |k'\rangle \right)$$

$$= \frac{1}{N} \sum_{k,k'} e^{\frac{2\pi i k}{N}(j-k')} |k'\rangle = \sum_{k'} \delta_{j,k'} |k'\rangle = |j\rangle$$

Quantum Fourier Transform

The QFT takes our original quantum state that stores the vector $(x_0, ..., x_{N-1})$ in the computational basis, and re-expresses it in an alternative basis of frequency components.

$$F_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad F_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

Classically, a naive implementation of the QFT takes time $\mathcal{O}(N^2)$, and much ado is made about the "fast Fourier transform" implementation in time $\mathcal{O}(N \log N)$. The quantum Fourier transform can be done exponentially faster, it can be decomposed into $\mathrm{poly}(\log N)$ elementary quantum gates.

The tradeoff with the QFT is that we only obtain the vector of frequency components as a quantum state. Therefore we could sample it, and learn a few dominant frequencies, but we can't just read through all N components like we would in a classical setting.

Quantum Fourier Transform

Starting from the vector $(x_0, ..., x_{N-1})$ as a quantum state, $|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$, we apply the QFT which takes

$$F_N|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle \quad , \quad j = 0, ..., N-1$$

Applying this linearly, $F_N|\psi\rangle = \sum_{k=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk} \right) |k\rangle$

And so the components of $F_N|\psi\rangle$ (the k-th of which is given by the sum in parentheses) are indeed the Fourier components of the vector $(x_0, ..., x_{N-1})$, as claimed.

The QFT is a powerful subroutine, allowing us to Fourier transform exponentially large vectors. But for it to be useful we need applications that only rely on the ability to sample these coefficients. Two such applications were found in the 1990s: Shor's quantum factoring algorithm, and Kitaev's phase estimation algorithm.

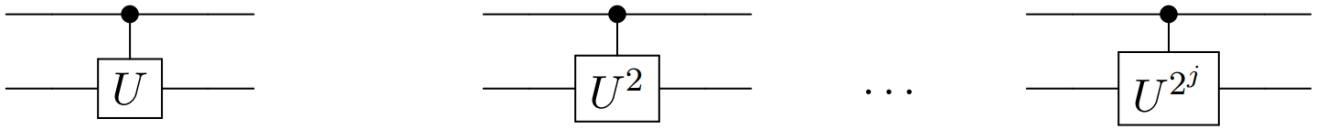
Quantum Phase Estimation

If U is a unitary operator, then all of its eigenvalues lie on the unit circle:

$$U|\psi\rangle = p|\psi\rangle \implies p = e^{2\pi i\theta}\ ,\ 0 \leq \theta < 1$$

Given U and the state $|\psi\rangle$, the goal of QPE is to determine p, or equivalently θ. This is equivalent to estimating the eigenvalues of the Hermitian operator H that generates U, so if we can do QPE we can find energies of H.

Besides the QFT, the main capability we need is to apply integer powers of U in a controlled manner, so we assume for some range of j we can implement:



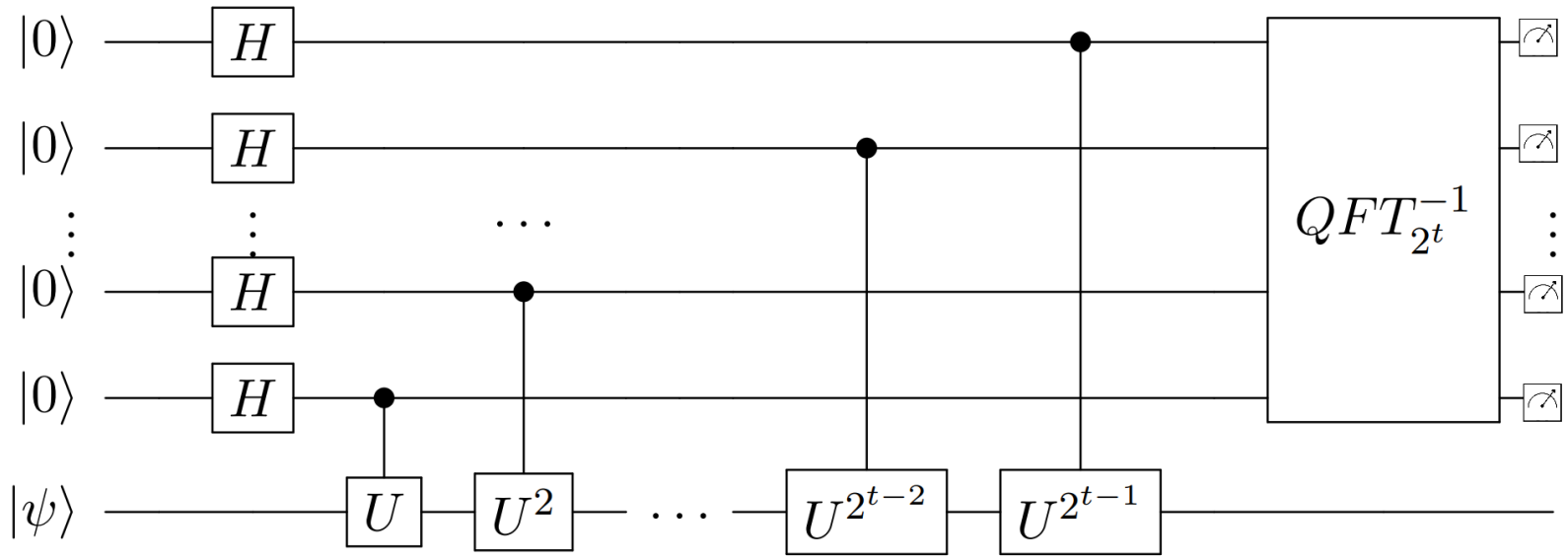
In this general symbol for a controlled unitary, the solid dot on the upper line indicates the control bit. If the control bit is 1, the unitary U is applied to the target register, and otherwise nothing happens e.g.

$$|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U$$

Quantum Phase Estimation

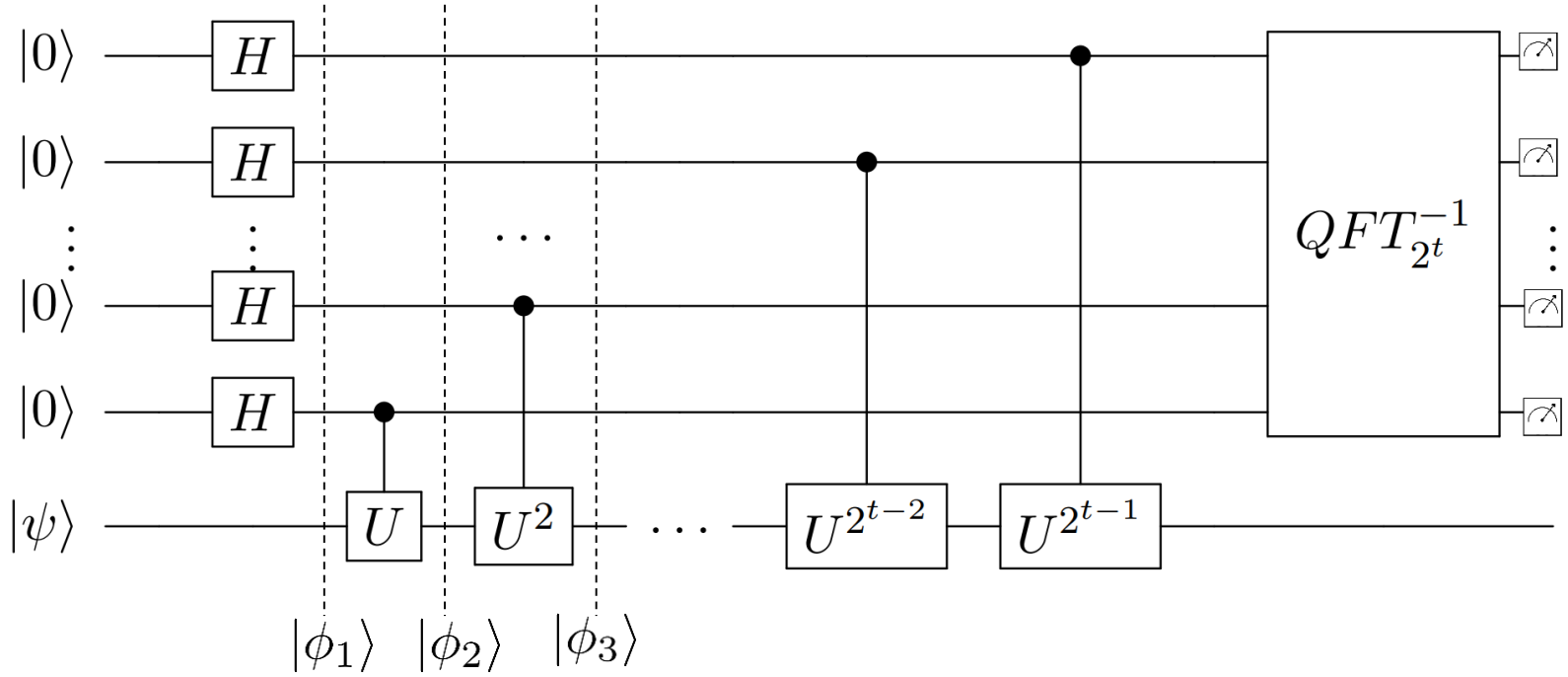
Since $0 \leq \theta < 1$ we write $\theta = 0.\theta_1\theta_2 \ldots \theta_t$ for the first t bits of θ expressed in binary. It simplifies the analysis if we assume this expression is exact, but in general we are approximating the angle by its first t bits.

Note that this assumed binary expansion is also equivalent to: $\theta = \frac{\theta_1}{2} + \frac{\theta_2}{4} + \ldots + \frac{\theta_t}{2^t}$



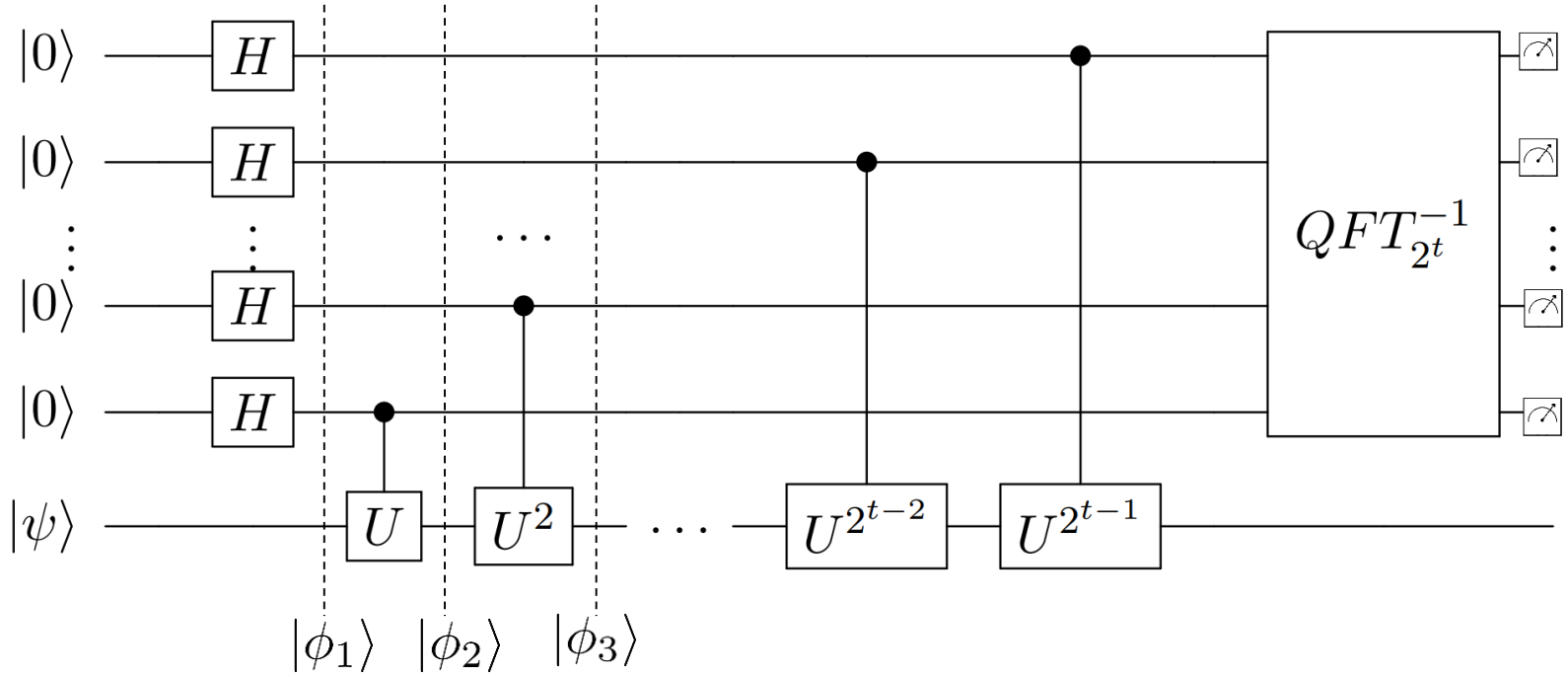
This is the entire phase estimation circuit. The measurements at the end read of the corresponding bits in the binary expansion of θ.

Quantum Phase Estimation



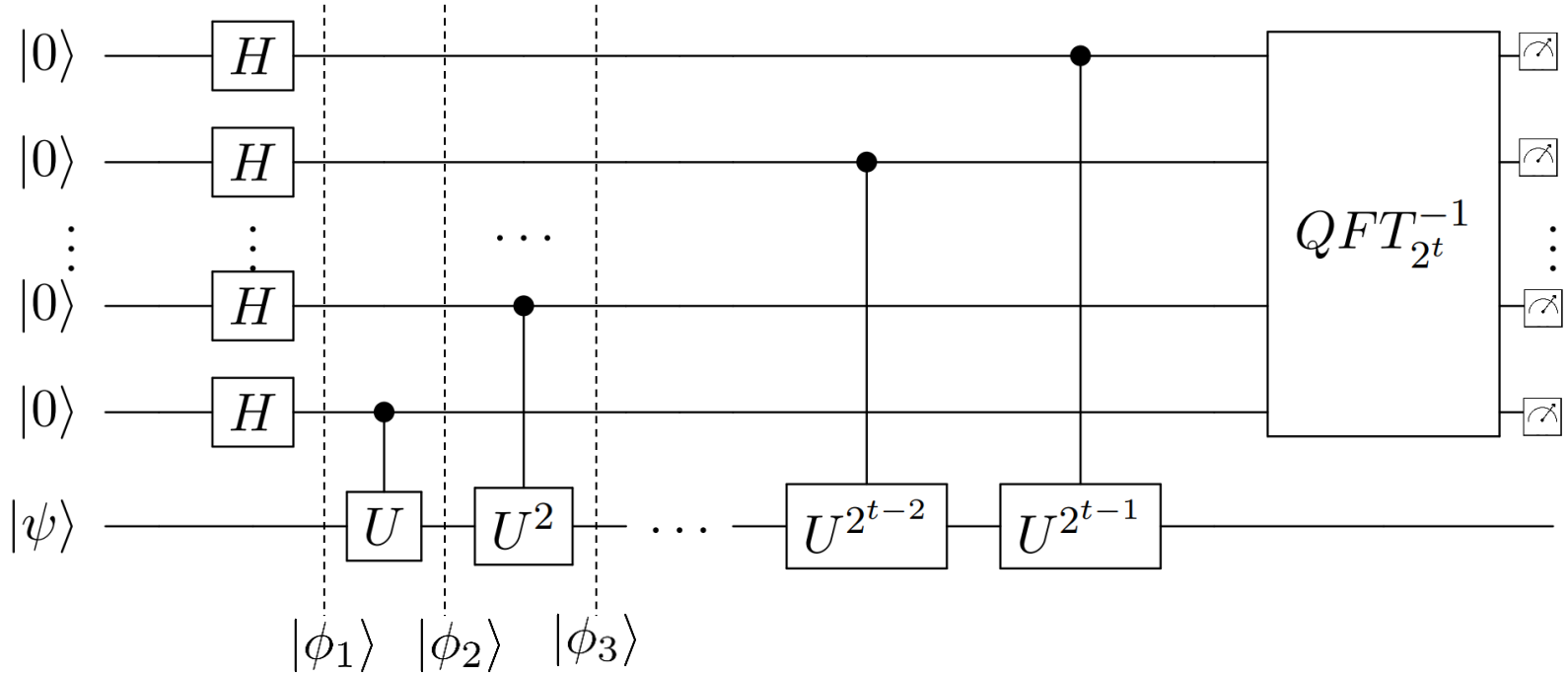
$$|\phi_1\rangle : (H|0\rangle)^{\otimes t} \otimes |\psi\rangle = \frac{1}{2^{t/2}} (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \otimes |\psi\rangle$$

Quantum Phase Estimation



$$|\phi_2\rangle : \frac{1}{2^{t/2}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + e^{i2\pi\theta}|1\rangle) \otimes |\psi\rangle$$

Quantum Phase Estimation



$$|\phi_t\rangle : \frac{1}{2^{t/2}}(|0\rangle+e^{i2\pi\theta 2^{t-1}}|1\rangle)\otimes(|0\rangle+e^{i2\pi\theta 2^{t-2}}|1\rangle)\otimes\cdots\otimes(|0\rangle+e^{i2\pi\theta}|1\rangle)\otimes|\psi\rangle = \frac{1}{2^{t/2}}\sum_{x=0}^{2^{t/2}-1} e^{i2\pi\theta x}|x\rangle|\psi\rangle$$

Quantum Phase Estimation

Therefore just prior to the inverse Fourier transform we have the state

$$|\phi_t\rangle = \left(2^{-t/2} \sum_{x=0}^{2^{t/2}-1} e^{i2\pi\theta x} |x\rangle \right) \otimes |\psi\rangle$$

Applying the inverse QFT to the first register yields

$$\left(2^{-t} \sum_{x=0}^{2^{t/2}-1} \sum_{k=0}^{2^{t/2}-1} e^{-\frac{2\pi i x}{2^t}(k-a)} |k\rangle \right) \quad , \quad a = 2^t\theta$$

By our simplifying assumption, there is some k which is exactly equal to a, and so the same calculation we used to practice inverting the QFT implies that measurement of this register returns the string with k = a. If a is only approximately equal to a, the cancellations are not perfect but still most of the amplitude on the k that is closest to a.

Quantum Phase Estimation

Now consider what happens if we apply phase estimation to a state that is not an eigenstate of U (or of H).

Any state can be decomposed as a superposition of eigenstates of H. Therefore we can apply the QPE algorithm to this superposition, and measuring any particular eigenvalue will collapse the state onto the corresponding eigenvector.

This leads to one of the most promising applications for quantum computers. We can use classical methods to design approximate ground states, and then run phase estimation to collapse into the true ground state. This will work with high probability if our initial state has a high overlap with the true ground state.

This algorithm was proposed in the context of Quantum Chemistry by Alan Aspuru-Guzik in 2005. Although the idea is fairly immediate, it was a major milestone in finding practical applications for quantum computation.

Notice that the main ingredient we need in order to perform phase estimation is to apply the unitary e^{itH}, which is done by simulating time evolution of a many-body Hamiltonian. Therefore it would not be too brash to declare quantum simulation and quantum phase estimation as the most important quantum algorithms (the ones we are most certain will give useful exponential speedups over classical computers).