Quantum Information
Chapter 6. Quantum Algorithms

John Preskill
Institute for Quantum Information and Matter
California Institute of Technology

Updated November 2020

For further updates and additional chapters, see:
http://www.theory.caltech.edu/people/preskill/ph219/

Please send corrections to preskill@caltech.edu

Contents

Preface

This is the 6th chapter of my book *Quantum Information*, based on the course I have been teaching at Caltech since 1997. An early version of this chapter has been available on the course website since 1998, but this version is substantially revised and expanded.

This is a working draft of Chapter 6, which I will continue to update. See the URL on the title page for further updates and drafts of other chapters. Please send an email to preskill@caltech.edu if you notice errors.

Eventually, the complete book will be published by Cambridge University Press. I hesitate to predict the publication date — they have been far too patient with me.

6
Quantum Algorithms

6.1 Some Quantum Algorithms

While we are not yet able to show that $BPP \neq BQP$, there are three approaches that we can pursue to study the differences between the capabilities of classical and quantum computers:

(1) Nonexponential speedup. We can find quantum algorithms that are demonstrably faster than the best classical algorithm, but not *exponentially* faster. These algorithms shed no light on the conventional classification of complexity. But they do demonstrate a type of separation between tasks that classical and quantum computers can perform. Example: Grover's quantum speedup of the search of an unsorted data base.

(2) "Relativized" exponential speedup. We can consider the problem of analyzing the contents of a "quantum black box." The box performs an *a priori* unknown) unitary transformation. We can prepare an input for the box, and we can measure its output; our task is to find out what the box does. It is possible to prove that quantum black boxes (computer scientists call them oracles[1]) exist with this property: By feeding quantum superpositions to the box, we can learn what is inside with an *exponential* speedup, compared to how long it would take if we were only allowed classical inputs. A computer scientist would say that $BPP \neq BQP$ "relative to the oracle." Example: Simon's exponential quantum speedup for finding the period of a 2 to 1 function.

(3) Exponential speedup for "apparently" hard problems. We can exhibit a quantum algorithm that solves a problem in polynomial time, where the problem appears to be hard classically, so that it is strongly suspected (though not proved) that the problem is not in BPP. Example: Shor's factoring algorithm.

Deutsch's problem. We will discuss examples from all three approaches. But first, we'll warm up by recalling an example of a simple quantum algorithm that was previously discussed in §1.5: Deutsch's algorithm for distinguishing between constant and balanced functions $f : \{0,1\} \to \{0,1\}$. We are presented with a quantum black box that computes $f(x)$; that is, it enacts the two-qubit unitary transformation

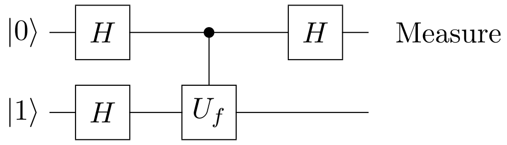
$$U_f : |x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle, \tag{6.1}$$

which flips the second qubit iff $f(\text{first qubit}) = 1$. Our assignment is to determine

[1] The term "oracle" signifies that the box responds to a query *immediately*; that is, the time it takes the box to operate is not included in the complexity analysis.

whether $f(0) = f(1)$. If we are restricted to the "classical" inputs $|0\rangle$ and $|1\rangle$, we need to access the box twice ($x = 0$ and $x = 1$) to get the answer. But if we are allowed to input a coherent superposition of these "classical" states, then once is enough.

The quantum circuit that solves the problem (discussed in §1.5) is:



Here H denotes the Hadamard transform

$$\boldsymbol{H} : |x\rangle \to \frac{1}{\sqrt{2}} \sum_y (-1)^{xy} |y\rangle, \tag{6.2}$$

or

$$\begin{aligned} \boldsymbol{H} : |0\rangle &\to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle &\to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle); \end{aligned} \tag{6.3}$$

that is, $\boldsymbol{H}$ is the 2×2 matrix

$$\boldsymbol{H} : \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}. \tag{6.4}$$

The circuit takes the input $|0\rangle|1\rangle$ to

$$\begin{aligned} |0\rangle|1\rangle \to & \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\ \to & \frac{1}{2}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)(|0\rangle - |1\rangle) \\ \to & \frac{1}{2}\Bigg[\left((-1)^{f(0)} + (-1)^{f(1)}\right)|0\rangle \\ & + \left((-1)^{f(0)} - (-1)^{f(1)}\right)|1\rangle\Bigg] \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{6.5}$$

Then when we measure the first qubit, we find the outcome $|0\rangle$ with probability one if $f(0) = f(1)$ (constant function) and the outcome $|1\rangle$ with probability one if $f(0) \neq f(1)$ (balanced function).

A quantum computer enjoys an advantage over a classical computer because it can invoke *quantum parallelism.* Because we input a superposition of $|0\rangle$ and $|1\rangle$, the output is sensitive to both the values of $f(0)$ and $f(1)$, even though we ran the box just once.

Deutsch–Jozsa problem. Now we'll consider some generalizations of Deutsch's problem. We will continue to assume that we are to analyze a quantum black box ("quantum oracle"). But in the hope of learning something about complexity, we will imagine that we have a family of black boxes, with variable input size. We are interested in how the time needed to find out what is inside the box scales with the size of the input (where "time" is measured by how many times we query the box).

In the *Deutsch–Jozsa problem*, we are presented with a quantum black box that computes a function taking n bits to 1,

$$f : \{0,1\}^n \to \{0,1\}, \tag{6.6}$$

and we have it on good authority that f is either constant ($f(x) = c$ for all x) or balanced ($f(x) = 0$ for exactly $\frac{1}{2}$ of the possible input values). We are to solve the decision problem: Is f constant or balanced?

In fact, we can solve this problem, too, accessing the box only once, using the same circuit as for Deutsch's problem (but with x expanded from one bit to n bits). We note that if we apply n Hadamard gates in parallel to n-qubits.

$$\boldsymbol{H}^{(n)} = \boldsymbol{H} \otimes \boldsymbol{H} \otimes \ldots \otimes \boldsymbol{H}, \tag{6.7}$$

then the n-qubit state transforms as

$$\boldsymbol{H}^{(n)} : |x\rangle \to \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2}} \sum_{y_i=\{0,1\}} (-1)^{x_i y_i} |y_i\rangle \right) \equiv \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle, \tag{6.8}$$

where x, y represent n-bit strings, and $x \cdot y$ denotes the *bitwise* AND (or mod 2 scalar product)

$$x \cdot y = (x_1 \wedge y_1) \oplus (x_2 \wedge y_2) \oplus \ldots \oplus (x_n \wedge y_n). \tag{6.9}$$

Acting on the input $(|0\rangle)^n|1\rangle$, the action of the circuit is

$$\begin{aligned} (|0\rangle)^n|1\rangle &\to \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &\to \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &\to \left(\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y} |y\rangle \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{6.10}$$

Now let us evaluate the sum

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y}. \tag{6.11}$$

If f is a constant function, the sum is

$$(-1)^{f(x)} \left(\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{x \cdot y} \right) = (-1)^{f(x)} \delta_{y,0}; \tag{6.12}$$

it vanishes unless $y = 0$. Hence, when we measure the n-bit register, we obtain the result $|y = 0\rangle \equiv (|0\rangle)^n$ with probability one. But if the function is balanced, then for $y = 0$, the sum becomes

$$\frac{1}{2^n} \sum_{x=0}^{2^{n-1}} (-1)^{f(x)} = 0, \tag{6.13}$$

(because half of the terms are $(+1)$ and half are (-1)). Therefore, the probability of obtaining the measurement outcome $|y = 0\rangle$ is zero.

We conclude that one query of the quantum oracle suffices to distinguish constant and balanced function with 100% confidence. The measurement result $y = 0$ means constant, any other result means balanced.

So quantum computation solves this problem neatly, but is the problem really hard classically? If we are restricted to classical input states $|x\rangle$, we can query the oracle repeatedly, choosing the input x at random (without replacement) each time. Once we obtain distinct outputs for two different queries, we have determined that the function is balanced (not constant). But if the function is in fact constant, we will not be *certain* it is constant until we have submitted $2^{n-1} + 1$ queries and have obtained the same response every time. In contrast, the quantum computation gives a definite response in only one go. So in this sense (if we demand absolute certainty) the classical calculation requires a number of queries exponential in n, while the quantum computation does not, and we might therefore claim an exponential quantum speedup.

But perhaps it is not reasonable to demand absolute certainty of the classical computation (particularly since any real quantum computer will be susceptible to errors, so that the quantum computer will also be unable to attain absolute certainty.) Suppose we are satisfied to guess balanced or constant, with a probability of success

$$P(\text{success}) > 1 - \varepsilon. \tag{6.14}$$

If the function is actually balanced, then if we make k queries, the probability of getting the same response every time is $p = 2^{-(k-1)}$. If after receiving the same response k consecutive times we guess that the function is balanced, then a quick Bayesian analysis shows that the probability that our guess is wrong is $\frac{1}{2^{k-1}+1}$ (assuming that balanced and constant are a priori equally probable). So if we guess after k queries, the probability of a wrong guess is

$$1 - P(\text{success}) = \frac{1}{2^{k-1}(2^{k-1}+1)}. \tag{6.15}$$

Therefore, we can achieve success probability $1 - \varepsilon$ for $\varepsilon^{-1} = 2^{k-1}(2^{k-1} + 1)$ or $k \sim \frac{1}{2}\log\left(\frac{1}{\varepsilon}\right)$. Since we can reach an exponentially good success probability with a polynomial number of trials, it is not really fair to say that the problem is hard.

Bernstein–Vazirani problem. Exactly the same circuit can be used to solve another variation on the Deutsch–Jozsa problem. Let's suppose that our quantum black box computes one of the functions f_a, where

$$f_a(x) = a \cdot x, \tag{6.16}$$

and a is an n-bit string. Our job is to determine a.

The quantum algorithm can solve this problem with certainty, given just one (n-qubit) quantum query. For this particular function, the quantum state in eq. (6.10) becomes

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{a\cdot x}(-1)^{x\cdot y}|y\rangle. \tag{6.17}$$

But in fact

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{a\cdot x}(-1)^{x\cdot y} = \delta_{a,y}, \tag{6.18}$$

so this state is $|a\rangle$. We can execute the circuit once and measure the n-qubit register, finding the n-bit string a with probability one.

If only classical queries are allowed, we acquire only one bit of information from each query, and it takes n queries to determine the value of a. Therefore, we have a clear separation between the quantum and classical difficulty of the problem. Even so, this example does not probe the relation of BPP to BQP, because the classical problem is not hard. The number of queries required classically is only linear in the input size, not exponential.

Simon's problem. Bernstein and Vazirani managed to formulate a variation on the above problem that *is* hard classically, and so establish for the first time a "relativized" separation between quantum and classical complexity. We will find it more instructive to consider a simpler example proposed somewhat later by Daniel Simon.

Once again we are presented with a quantum black box, and this time we are assured that the box computes a function

$$f : \{0,1\}^n \to \{0,1\}^n, \tag{6.19}$$

that is 2-to-1. Furthermore, the function has a "period" given by the n-bit string a; that is

$$f(x) = f(y) \qquad \text{iff} \qquad y = x \oplus a, \tag{6.20}$$

where here $\oplus$ denotes the bitwise XOR operation. (So a is the period if we regard x as taking values in $(Z_2)^n$ rather than Z_{2^n}.) This is all we know about f. Our job is to determine the value of a.

Classically this problem is *hard*. We need to query the oracle an exponentially large number of times to have any reasonable probability of finding a. We don't learn anything until we are fortunate enough to choose two queries x and y that happen to satisfy $x \oplus y = a$. Suppose, for example, that we choose $2^{n/4}$ queries. The number of pairs of queries is less than $(2^{n/4})^2$, and for each pair $\{x, y\}$, the probability that $x \oplus y = a$ is 2^{-n}. Therefore, the probability of successfully finding a is less than

$$2^{-n}(2^{n/4})^2 = 2^{-n/2}; \tag{6.21}$$

even with exponentially many queries, the success probability is exponentially small.

If we wish, we can frame the question as a decision problem: Either f is a 1-1 function, or it is 2-to-1 with some randomly chosen period a, each occurring with an a priori probability $\frac{1}{2}$. We are to determine whether the function is 1-to-1 or 2-to-1. Then, after $2^{n/4}$ classical queries, our probability of making a correct guess is

$$P(\text{success}) < \frac{1}{2} + \frac{1}{2^{n/2}}, \tag{6.22}$$

which does not remain bounded away from $\frac{1}{2}$ as n gets large.

But with quantum queries the problem is easy! The circuit we use is essentially the same as above, but now *both* registers are expanded to n qubits. We prepare the equally weighted superposition of all n-bit strings (by acting on $|0\rangle$ with $\boldsymbol{H}^{(n)}$), and then we query the oracle:

$$U_f : \left(\sum_{x=0}^{2^n-1} |x\rangle \right) |0\rangle \to \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle. \tag{6.23}$$

Now we measure the second register. (This step is not actually necessary, but I include it here for the sake of pedagogical clarity.) The measurement outcome is selected at random from the 2^{n-1} possible values of $f(x)$, each occurring equiprobably. Suppose the outcome is $f(x_0)$. Then because both x_0 and $x_0 \oplus a$, and only these values, are mapped by f to $f(x_0)$, we have prepared the state

$$\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle) \tag{6.24}$$

in the first register.

Now we want to extract some information about a. Clearly it would do us no good to measure the register (in the computational basis) at this point. We would obtain either the outcome x_0 or $x_0 \oplus a$, each occurring with probability $\frac{1}{2}$, but neither outcome would reveal anything about the value of a.

But suppose we apply the Hadamard transform $\boldsymbol{H}^{(n)}$ to the register before we measure:

$$\begin{aligned} \boldsymbol{H}^{(n)} :& \frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 + a\rangle) \\ &\to \frac{1}{2^{(n+1)/2}} \sum_{y=0}^{2^n-1} \left[(-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus a)\cdot y}\right] |y\rangle \\ &= \frac{1}{2^{(n-1)/2}} \sum_{a \cdot y = 0} (-1)^{x_0 \cdot y} |y\rangle. \end{aligned} \tag{6.25}$$

If $a \cdot y = 1$, then the terms in the coefficient of $|y\rangle$ interfere destructively. Hence only states $|y\rangle$ with $a \cdot y = 0$ survive in the sum over y. The measurement outcome, then, is selected at random from all possible values of y such that $a \cdot y = 0$, each occurring with probability $2^{-(n-1)}$.

We run this algorithm repeatedly, each time obtaining another value of y satisfying $y \cdot a = 0$. Once we have found n such linearly independent values $\{y_1, y_2, y_3 \ldots y_n\}$ (that is, linearly independent over $(Z_2)^n$), we can solve the equations

$$\begin{aligned} y_1 \cdot a &= 0 \\ y_2 \cdot a &= 0 \\ &\vdots \\ y_n \cdot a &= 0, \end{aligned} \tag{6.26}$$

to determine a unique value of a, and our problem is solved. It is easy to see that with $O(n)$ repetitions, we can attain a success probability that is exponentially close to 1.

So we finally have found an example where, given a particular type of quantum oracle, we can solve a problem in polynomial time by exploiting quantum superpositions, while exponential time is required if we are limited to classical queries. As a computer scientist might put it:

There exists an oracle relative to which $BQP \neq BPP$.

Note that whenever we compare classical and quantum complexity relative to an oracle, we are considering a quantum oracle (queries and replies are states in Hilbert space), but with a preferred orthonormal basis. If we submit a classical query (an element of the preferred basis) we always receive a classical response (another basis element). The issue is whether we can achieve a significant speedup by choosing more general quantum queries.

6.2 Periodicity

So far, the one case for which we have found an exponential separation between the speed of a quantum algorithm and the speed of the corresponding classical algorithm is the case of Simon's problem. Simon's algorithm exploits quantum parallelism to speed up the search for the period of a function. Its success encourages us to seek other quantum algorithms designed for other kinds of period finding.

Simon studied periodic functions taking values in $(Z_2)^n$. For that purpose the n-bit Hadamard transform $\boldsymbol{H}^{(n)}$ was a powerful tool. If we wish instead to study periodic functions taking values in Z_{2^n}, the (discrete) Fourier transform will be a tool of comparable power.

The moral of Simon's problem is that, while finding needles in a haystack may be difficult, finding *periodically* spaced needles in a haystack can be far easier. For example, if we scatter a photon off of a periodic array of needles, the photon is likely to be scattered in one of a set of preferred directions, where the Bragg scattering condition is satisfied. These preferred directions depend on the spacing between the needles, so by scattering just one photon, we can already collect some useful information about the spacing. We should further explore the implications of this metaphor for the construction of efficient quantum algorithms.

So imagine a quantum oracle that computes a function

$$f : \{0,1\}^n \to \{0,1\}^m, \tag{6.27}$$

that has an unknown period r, where r is a positive integer satisfying

$$1 \ll r \ll 2^n. \tag{6.28}$$

That is,

$$f(x) = f(x + mr), \tag{6.29}$$

where m is any integer such that x and $x + mr$ lie in $\{0, 1, 2, \ldots, 2^n - 1\}$. We are to find the period r. Classically, this problem is *hard.* If r is, say, of order $2^{n/2}$, we will need to query the oracle of order $2^{n/4}$ times before we are likely to find two values of x that are mapped to the same value of $f(x)$, and hence learn something about r. But we will see that there is a quantum algorithm that finds r in time poly (n).

Even if we know how to compute efficiently the function $f(x)$, it may be a hard problem to determine its period. Our quantum algorithm can be applied to finding, in poly(n) time, the period of any function that we can compute in poly(n) time. Efficient period finding allows us to efficiently solve a variety of (apparently) hard problems, such as factoring an integer, or evaluating a discrete logarithm.

The key idea underlying quantum period finding is that the Fourier transform can be evaluated by an efficient quantum circuit (as discovered by Peter Shor). The quantum Fourier transform (QFT) exploits the power of quantum parallelism to achieve an exponential speedup of the well-known (classical) fast Fourier transform (FFT). Since the FFT has such a wide variety of applications, perhaps the QFT will also come into widespread use someday.

6.2.1 Finding the period

The QFT is the unitary transformation that acts on the computational basis according to

$$QFT : |x\rangle \to \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N} |y\rangle, \tag{6.30}$$

where $N = 2^n$. For now let's suppose that we can perform the QFT efficiently, and see how it enables us to extract the period of $f(x)$.

Emulating Simon's algorithm, we first query the oracle with the input $\frac{1}{\sqrt{N}} \sum_x |x\rangle$ (easily prepared by applying $\boldsymbol{H}^{(n)}$ to $|0\rangle$), and so prepare the state

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |f(x)\rangle. \tag{6.31}$$

Then we measure the output register, obtaining the result $|f(x_0)\rangle$ for some $0 \le x_0 < r$. This measurement prepares in the input register the coherent superposition of the A values of x that are mapped to $f(x_0)$:

$$\frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle, \tag{6.32}$$

where

$$N - r \le x_0 + (A-1)r < N, \tag{6.33}$$

or

$$A - 1 < \frac{N}{r} < A + 1. \tag{6.34}$$

Actually, the measurement of the output register is unnecessary. If it is omitted, the state of the input register is an incoherent superposition (summed over $x_0 \in \{0, 1, \ldots r-1\}$) of states of the form eq. (6.32). The rest of the algorithm works just as well acting on this initial state.

Now our task is to extract the value of r from the state eq. (6.32) that we have prepared. Were we to measure the input register by projecting onto the computational basis at this point, we would learn nothing about r. Instead (cf. Simon's algorithm), we should Fourier transform first and *then* measure.

By applying the QFT to the state eq. (6.32) we obtain

$$\frac{1}{\sqrt{NA}} \sum_{y=0}^{N-1} e^{2\pi i x_0 y/N} \sum_{j=0}^{A-1} e^{2\pi i j r y/N} |y\rangle. \tag{6.35}$$

If we now measure in the computational basis, the probability of obtaining the outcome y is

$$\text{Prob}(y) = \frac{A}{N} \left| \frac{1}{A} \sum_{j=0}^{A-1} e^{2\pi i j r y/N} \right|^2 . \tag{6.36}$$

This distribution strongly favors values of y such that yr/N is close to an integer. For example, if N/r happened to be an integer (and therefore equal to A), we would have

$$\mathrm{Prob}(y) = \frac{1}{r}\left|\frac{1}{A}\sum_{j=0}^{A-1} e^{2\pi i j y/A}\right| = \begin{cases} \frac{1}{r} & y = A\cdot(\text{integer}) \\ 0 & \text{otherwise.} \end{cases} \tag{6.37}$$

More generally, we may sum the geometric series

$$\sum_{j=0}^{A-1} e^{i\theta j} = \frac{e^{iA\theta}-1}{e^{i\theta}-1}, \tag{6.38}$$

where

$$\theta_y = \frac{2\pi y r(\text{mod } N)}{N}. \tag{6.39}$$

There are precisely r values of y in $\{0, 1, \ldots, N-1\}$ that satisfy

$$-\frac{r}{2} \le yr(\text{mod } N) \le \frac{r}{2}. \tag{6.40}$$

(To see this, imagine marking the multiples of r and N on a number line ranging from 0 to $rN-1$. For each multiple of N, there is a multiple of r no more than distance $r/2$ away.) For each of these values, the corresponding θ_y satisfies.

$$-\pi\frac{r}{N} \le \theta_y \le \pi\frac{r}{N}. \tag{6.41}$$

Now, since $A - 1 < \frac{N}{r}$, for these values of θ_y all of the terms in the sum over j in eq. (6.38) lie in the same half-plane, so that the terms interfere constructively and the sum is substantial.

We know that

$$|1 - e^{i\theta}| \le |\theta|, \tag{6.42}$$

because the straight-line distance from the origin is less than the arc length along the circle, and for $A|\theta| \le \pi$, we know that

$$|1 - e^{iA\theta}| \ge \frac{2A|\theta|}{\pi}, \tag{6.43}$$

because we can see (either graphically or by evaluating its derivative) that this distance is a convex function. We actually have $A < \frac{N}{r} + 1$, and hence $A\theta_y < \pi\left(1 + \frac{r}{N}\right)$, but by applying the above bound to

$$\left|\frac{e^{i(A-1)\theta}-1}{e^{i\theta}-1} + e^{i(A-1)\theta}\right| \ge \left|\frac{e^{i(A-1)\theta}-1}{e^{i\theta}-1}\right| - 1, \tag{6.44}$$

we can still conclude that

$$\left|\frac{e^{iA\theta}-1}{e^{i\theta}-1}\right| \ge \frac{2(A-1)|\theta|}{\pi|\theta|} - 1 = \frac{2}{\pi}A - \left(1 + \frac{2}{\pi}\right). \tag{6.45}$$

Ignoring a possible correction of order $2/A$, then, we find

$$\mathrm{Prob}(y) \ge \left(\frac{4}{\pi^2}\right)\frac{1}{r}, \tag{6.46}$$

for each of the r values of y that satisfy eq. (6.40). Therefore, with a probability of at least $4/\pi^2$, the measured value of y will satisfy

$$k\frac{N}{r} - \frac{1}{2} \leq y \leq k\frac{N}{r} + \frac{1}{2}, \tag{6.47}$$

or

$$\frac{k}{r} - \frac{1}{2N} \leq \frac{y}{N} \leq \frac{k}{r} + \frac{1}{2N}, \tag{6.48}$$

where k is an integer chosen from $\{0, 1, \ldots, r-1\}$. The output of the computation is reasonable likely to be within distance $1/2$ of an integer multiple of N/r.

Suppose that we know that $r < M \ll N$. Thus N/r is a rational number with a denominator less than M. Two distinct rational numbers, each with denominator less than M, can be no closer together than $1/M^2$, since $\frac{a}{b} - \frac{c}{d} = \frac{ad-bc}{bd}$. If the measurement outcome y satisfies eq. (6.47), then there is a *unique* value of k/r (with $r < M$) determined by y/N, provided that $N \geq M^2$. This value of k/r can be efficiently extracted from the measured y/N, by the continued fraction method.

Now, with probability exceeding $4/\pi^2$, we have found a value of k/r where k is selected (roughly equiprobably) from $\{0, 1, 2, \ldots, r-1\}$. It is reasonably likely that k and r are relatively prime (have no common factor), so that we have succeeded in finding r. With a query of the oracle, we may check whether $f(x) = f(x+r)$. But if $\text{GCD}(k, r) \neq 1$, we have found only a factor (r_1) of r.

If we did not succeed, we could test some nearby values of y (the measured value might have been close to the range $-r/2 \leq yr(\text{mod } N) \leq r/2$ without actually lying inside), or we could try a few multiples of r (the value of $\text{GCD}(k, r)$, if not 1, is probably not large). That failing, we resort to a repetition of the quantum circuit, this time (with probability at least $4/\pi^2$) obtaining a value k'/r. Now k', too, may have a common factor with r, in which case our procedure again determines a factor (r_2) of r. But it is reasonably likely that $\text{GCD}(k, k') = 1$, in which case $r = \text{LCM}(r_1, r_2)$. Indeed, we can estimate the probability that randomly selected k and k' are relatively prime as follows: Since a prime number p divides a fraction $1/p$ of all numbers, the probability that p divides both k and k' is $1/p^2$. And k and k' are coprime if and only if there is no prime p that divides both. Therefore,

$$\text{Prob}(k, k' \text{ coprime}) = \prod_{\text{prime } p} \left(1 - \frac{1}{p^2}\right) = \frac{1}{\zeta(2)} = \frac{6}{\pi^2} \simeq .607 \tag{6.49}$$

(where $\zeta(z)$ denotes the Riemann zeta function). Therefore, we are likely to succeed in finding the period r after some constant number (independent of N) of repetitions of the algorithm.

6.2.2 From FFT to QFT

Now let's consider the implementation of the quantum Fourier transform. The Fourier transform

$$\sum_x f(x)|x\rangle \rightarrow \sum_y \left(\frac{1}{\sqrt{N}} \sum_x e^{2\pi i x y/N} f(x)\right) |y\rangle, \tag{6.50}$$

is multiplication by an $N \times N$ unitary matrix, where the (x, y) matrix element is $(e^{2\pi i/N})^{xy}$. Naively, this transform requires $O(N^2)$ elementary operations. But there is a well-known and very useful (classical) procedure that reduces the number of operations to $O(N\log N)$. Assuming $N=2^n$, we express x and y as binary expansions

$$\begin{aligned} x &= x_{n-1}\cdot 2^{n-1}+x_{n-2}\cdot 2^{n-2}+\ldots+x_1\cdot 2+x_0 \\ y &= y_{n-1}\cdot 2^{n-1}+y_{n-2}\cdot 2^{n-2}+\ldots+y_1\cdot 2+y_0. \end{aligned} \tag{6.51}$$

In the product of x and y, we may discard any terms containing n or more powers of 2, as these make no contribution to $e^{2\pi ixy}/2^n$. Hence

$$\begin{aligned} \frac{xy}{2^n} &\equiv y_{n-1}(.x_0)+y_{n-2}(.x_1x_0)+y_{n-3}(.x_2x_1x_0)+\ldots \\ &\quad + y_1(.x_{n-2}x_{n-3}\ldots x_0)+y_0(.x_{n-1}x_{n-2}\ldots x_0), \end{aligned} \tag{6.52}$$

where the factors in parentheses are binary expansions; *e.g.*,

$$.x_2x_1x_0=\frac{x_2}{2}+\frac{x_1}{2^2}+\frac{x_0}{2^3}. \tag{6.53}$$

We can now evaluate

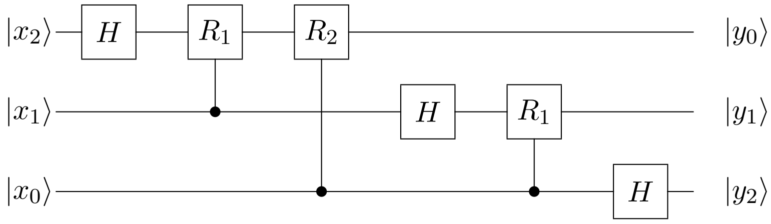
$$\tilde{f}(x)=\frac{1}{\sqrt{N}}\sum_y e^{2\pi ixy/N}f(y), \tag{6.54}$$

for each of the N values of x. But the sum over y factors into n sums over $y_k=0,1$, which can be done sequentially in a time of order n.

With quantum parallelism, we can do far better. From eq. (6.52) we obtain

$$\begin{aligned} QFT:&|x\rangle \to \frac{1}{\sqrt{N}}\sum_y e^{2\pi ixy/N}|y\rangle \\ &=\frac{1}{\sqrt{2^n}}\left(|0\rangle+e^{2\pi i(.x_0)}|1\rangle\right)\left(|0\rangle+e^{2\pi i(.x_1x_0)}|1\rangle\right) \\ &\ldots\left(|0\rangle+e^{2\pi i(.x_{n-1}x_{n-2}\ldots x_0)}|1\rangle\right). \end{aligned} \tag{6.55}$$

The QFT takes each computational basis state to an *unentangled* state of n qubits; thus we anticipate that it can be efficiently implemented. Indeed, let's consider the case $n=3$. We can readily see that the circuit



does the job (but note that the order of the bits has been reversed in the output). Each Hadamard gate acts as

$$\boldsymbol{H}:|x_k\rangle\to\frac{1}{\sqrt{2}}\left(|0\rangle+e^{2\pi i(.x_k)}|1\rangle\right). \tag{6.56}$$

The other contributions to the relative phase of $|0\rangle$ and $|1\rangle$ in the kth qubit are provided by the two-qubit conditional rotations, where

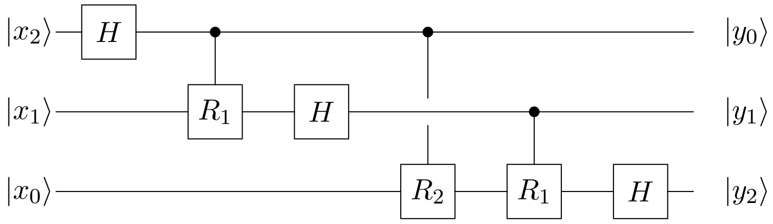
$$\boldsymbol{R}_d=\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2^d} \end{pmatrix}, \tag{6.57}$$

and $d = (k - j)$ is the “distance” between the qubits.

In the case $n = 3$, the QFT is constructed from three $\boldsymbol{H}$ gates and three controlled-$\boldsymbol{R}$ gates. For general n, the obvious generalization of this circuit requires n $\boldsymbol{H}$ gates and $\binom{n}{2} = \frac{1}{2}n(n-1)$ controlled R's. A two qubit gate is applied to each pair of qubits, again with controlled relative phase $\pi/2^d$, where d is the “distance” between the qubits. Thus the circuit family that implements QFT has a size of order $(\log N)^2$.

We can reduce the circuit complexity to linear in $\log N$ if we are willing to settle for an implementation of fixed accuracy, because the two-qubit gates acting on distantly separated qubits contribute only exponentially small phases. If we drop the gates acting on pairs with distance greater than m, than each term in eq. (6.52) is replaced by an approximation to m bits of accuracy; the total error in $xy/2^n$ is certainly no worse than $n2^{-m}$, so we can achieve accuracy ε in $xy/2^n$ with $m \geq \log n/\varepsilon$. If we retain only the gates acting on qubit pairs with distance m or less, then the circuit size is $mn \sim n \log n/\varepsilon$.

In fact, if we are going to measure in the computational basis immediately after implementing the QFT (or its inverse), a further simplification is possible – no two-qubit gates are needed at all! We first remark that the controlled – $\boldsymbol{R}_d$ gate acts symmetrically on the two qubits – it acts trivially on $|00\rangle$, $|01\rangle$, and $|10\rangle$, and modifies the phase of $|11\rangle$ by $e^{i\theta_d}$. Thus, we can interchange the “control” and “target” bits without modifying the gate. With this change, our circuit for the 3-qubit QFT can be redrawn as:



Once we have measured $|y_0\rangle$, we *know* the value of the control bit in the controlled-$\boldsymbol{R}_1$ gate that acted on the first two qubits. Therefore, we will obtain the same probability distribution of measurement outcomes if, instead of applying controlled-$\boldsymbol{R}_1$ and then measuring, we instead measure y_0 first, and then apply $(\boldsymbol{R}_1)^{y_0}$ to the next qubit, conditioned on the outcome of the measurement of the first qubit. Similarly, we can replace the controlled-$\boldsymbol{R}_1$ and controlled-$\boldsymbol{R}_2$ gates acting on the third qubit by the single qubit rotation

$$(\boldsymbol{R}_2)^{y_0}(\boldsymbol{R}_1)^{y_1}, \qquad (6.58)$$

(that is, a rotation with relative phase $\pi(.y_1y_0)$) *after* the values of y_1 and y_0 have been measured.

Altogether then, if we are going to measure after performing the QFT, only n Hadamard gates and $n-1$ single-qubit rotations are needed to implement it. The QFT is remarkably simple!

6.3 Factoring

6.3.1 Factoring as period finding

What does the factoring problem (finding the prime factors of a large composite positive integer) have to do with periodicity? There is a well-known (randomized) reduction of factoring to determining the period of a function. Although this reduction is not directly related to quantum computing, we will discuss it here for completeness, and because the prospect of using a quantum computer as a factoring engine has generated so much excitement.

Suppose we want to find a factor of the n-bit number N. Select pseudo-randomly $a < N$, and compute the greatest common divisor $\mathrm{GCD}(a, N)$, which can be done efficiently (in a time of order $(\log N)^3$) using the Euclidean algorithm. If $\mathrm{GCD}(a, N) \neq 1$ then the GCD is a nontrivial factor of N, and we are done. So suppose $\mathrm{GCD}(a, N) = 1$.

[**Aside: The Euclidean algorithm.** To compute $\mathrm{GCD}(N_1, N_2)$ (for $N_1 > N_2$) first divide N_1 by N_2 obtaining remainder R_1. Then divide N_2 by R_1, obtaining remainder R_2. Divide R_1 by R_2, *etc.* until the remainder is 0. The last nonzero remainder is $R = \mathrm{GCD}(N_1, N_2)$. To see that the algorithm works, just note that (1) R divides all previous remainders and hence also N_1 and N_2, and (2) *any* number that divides N_1 and N_2 will also divide all remainders, including R. A number that divides both N_1 and N_2, and also is divided by any number that divides both N_1 and N_2 must be $\mathrm{GCD}(N_1, N_2)$. To see how long the Euclidean algorithm takes, note that

$$R_j = qR_{j+1} + R_{j+2}, \tag{6.59}$$

where $q \geq 1$ and $R_{j+2} < R_{j+1}$; therefore $R_{j+2} < \frac{1}{2}R_j$. Two divisions reduce the remainder by at least a factor of 2, so no more than $2\ \log N_1$ divisions are required, with each division using $O((\log N)^2)$ elementary operations; the total number of operations is $O((\log N)^3)$.]

The numbers $a < N$ coprime to N (having no common factor with N) form a finite group under multiplication mod N. [Why? We need to establish that each element a has an inverse. But for given $a < N$ coprime to N, each ab (mod N) is distinct, as b ranges over all $b < N$ coprime to N. (If N divides $ab - ab'$, it must divide $b - b'$.) Therefore, for some b, we must have $ab \equiv 1$ (mod N); hence the inverse of a exists.] Each element a of this finite group has a finite *order* r, the smallest positive integer such that

$$a^r \equiv 1 \ (\mathrm{mod}\ N). \tag{6.60}$$

The order of a mod N is the period of the function

$$f_{N,a}(x) = a^x \ (\mathrm{mod}\ N). \tag{6.61}$$

We know there is an efficient quantum algorithm that can find the period of a function; therefore, if we can compute $f_{N,a}$ efficiently, we can find the order of a efficiently.

Computing $f_{N,a}$ may look difficult at first, since the exponent x can be very large. But if $x < 2^m$ and we express x as a binary expansion

$$x = x_{m-1} \cdot 2^{m-1} + x_{m-2} \cdot 2^{m-2} + \ldots + x_0, \tag{6.62}$$

we have

$$a^x(\mathrm{mod}\ N) = (a^{2^{m-1}})^{x_{m-1}}(a^{2^{m-2}})^{x_{m-2}} \ldots (a)^{x_0} \ (\mathrm{mod}\ N). \tag{6.63}$$

Each a^{2^j} has a large exponent, but can be computed efficiently by a *classical* computer, using repeated squaring

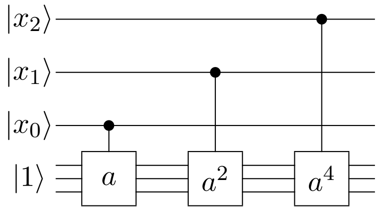
$$a^{2^j} \ \ (\mathrm{mod}\ N) = (a^{2^{j-1}})^2 \ \ (\mathrm{mod}\ N). \tag{6.64}$$

So only $m-1$ (classical) mod N multiplications are needed to assemble a table of all a^{2^j}'s.

The computation of $a^x (\text{mod } N)$ is carried out by executing a routine:

```
INPUT 1
For $j = 0$ to $m-1$, if $x_j = 1$, MULTIPLY $a^{2^j}$.
```

This routine requires at most m mod N multiplications, each requiring of order $(\log N)^2$ elementary operations. (Using tricks for performing efficient multiplication of very large numbers, the number of elementary operations can be reduced to $O(\log N \log\log N \log\log\log N)$; thus, asymptotically for large N, a circuit family with size $O(\log^2 N \log\log N \log\log\log N)$ can compute $f_{N,a}$.) Since $r < N$, we will have a reasonable chance of success at extracting the period if we choose $m \sim 2\log N$. Hence, the computation of $f_{N,a}$ can be carried out by a circuit family of size $O((\log N)^3)$. Schematically, the circuit has the structure:



Multiplication by a^{2^j} is performed if the control qubit x_j has the value 1.

Suppose we have found the period r of a mod N. Then *if* r is even, we have

$$N \text{ divides } \left(a^{\frac{r}{2}}+1\right)\left(a^{\frac{r}{2}}-1\right). \tag{6.65}$$

We know that N does not divide $a^{r/2}-1$; if it did, the order of a would be $\le r/2$. Thus, *if* it is also the case that N does not divide $a^{r/2}+1$, or

$$a^{r/2} \ne -1 \ (\text{mod } N), \tag{6.66}$$

then N must have a nontrivial common factor with each of $a^{r/2} \pm 1$. Therefore, $\text{GCD}(N, a^{r/2}+1) \ne 1$ is a factor (that we can find efficiently by a classical computation), and we are done.

We see that, once we have found r, we succeed in factoring N *unless* either (1) r is odd or (2) r is even and $a^{r/2} \equiv -1$ (mod N). How likely is success?

Let's suppose that N is a product of two prime factors $p_1 \ne p_2$,

$$N = p_1 p_2 \tag{6.67}$$

(this is actually the least favorable case). For each $a < p_1 p_2$, there exist unique $a_1 < p_1$ and $a_2 < p_2$ such that

$$\begin{aligned} a &\equiv a_1 \quad (\text{mod } p_1) \\ a &\equiv a_2 \quad (\text{mod } p_2). \end{aligned} \tag{6.68}$$

Choosing a random $a < N$ is, therefore, equivalent to choosing random $a, < p_1$ and $a_2 < p_2$.

[**Aside**: We're using the **Chinese Remainder Theorem.** The a solving eq. (6.68) is unique because if a and b are both solutions, then both p_1 and p_2 must divide $a-b$. The solution exists because every $a < p_1p_2$ solves eq. (6.68) for *some* a_1 and a_2. Since there are exactly p_1p_2 ways to choose a_1 and a_2, and exactly p_1p_2 ways to choose a, uniqueness implies that there is an a corresponding to each pair a_1, a_2.]

Now let r_1 denote the order of a_1 mod p_1 and r_2 denote the order of a_2 mod p_2. The Chinese remainder theorem tells us that $a^r \equiv 1 \pmod{p_1p_2}$ is equivalent to

$$\begin{aligned} a_1^r &\equiv 1 \pmod{p_1} \\ a_2^r &\equiv 1 \pmod{p_2}. \end{aligned} \tag{6.69}$$

Therefore $r = \mathrm{LCM}(r_1, r_2)$. If r_1 and r_2 are both odd, then so is r, and we lose.

But if *either* r_1 or r_2 is even, then so is r, and we are still in the game. If

$$\begin{aligned} a^{r/2} &\equiv -1 \pmod{p_1} \\ a^{r/2} &\equiv -1 \pmod{p_2}. \end{aligned} \tag{6.70}$$

Then we have $a^{r/2} \equiv -1 \pmod{p_1p_2}$ and we still lose. But if either

$$\begin{aligned} a^{r/2} &\equiv -1 \pmod{p_1} \\ a^{r/2} &\equiv 1 \pmod{p_2}, \end{aligned} \tag{6.71}$$

or

$$\begin{aligned} a^{r/2} &\equiv 1 \pmod{p_1} \\ a^{r/2} &\equiv -1 \pmod{p_2}, \end{aligned} \tag{6.72}$$

then $a^{r/2} \not\equiv -1 (\mathrm{mod}\ p_1p_2)$ and we win. (Of course, $a^{r/2} \equiv 1 \pmod{p_1}$ and $a^{r/2} \equiv 1 \pmod{p_2}$ is not possible, for that would imply $a^{r/2} \equiv 1 \pmod{p_1p_2}$, and r could not be the order of a.)

Suppose that

$$\begin{aligned} r_1 &= 2^{c_1} \cdot \mathrm{odd} \\ r_2 &= 2^{c_2} \cdot \mathrm{odd}, \end{aligned} \tag{6.73}$$

where $c_1 > c_2$. Then $r = \mathrm{LCM}(r_1, r_2) = 2r_2 \cdot$ integer, so that $a^{r/2} \equiv 1 \pmod{p_2}$ and eq. (6.71) is satisfied – we win! Similarly $c_2 > c_1$ implies eq. (6.72) – again we win. But for $c_1 = c_2$, $r = r_1 \cdot (\mathrm{odd}) = r_2 \cdot (\mathrm{odd}')$ so that eq. (6.70) is satisfied – in that case we lose.

Okay – it comes down to: for $c_1 = c_2$ we lose, for $c_1 \neq c_2$ we win. How likely is $c_1 \neq c_2$?

It helps to know that the multiplicative group mod p is cyclic – it contains a primitive element of order $p-1$, so that all elements are powers of the primitive element. [Why? The integers mod p are a finite *field*. If the group were not cyclic, the maximum order of the elements would be $q < p-1$, so that $x^q \equiv 1 \pmod{p}$ would have $p-1$ solutions. But that can't be: in a finite field there are no more than q qth roots of unity.]

Suppose that $p-1 = 2^k \cdot s$, where s is odd, and consider the orders of all the elements of the cyclic group of order $p-1$. For brevity, we'll discuss only the case $k=1$, which is the least favorable case for us. Then if b is a primitive (order $2s$) element, the even powers of b have odd order, and the odd powers of b have order $2\cdot$ (odd). In this case, then, $r = 2^{c}\cdot$ (odd) where $c \in \{0, 1\}$, each occuring equiprobably. Therefore, if p_1 and p_2 are both of this (unfavorable) type, and a_1, a_2 are chosen randomly, the probability that $c_1 \neq c_2$ is $\frac{1}{2}$. Hence, once we have found r, our probability of successfully finding a factor is at least $\frac{1}{2}$, if N is a product of two distinct primes. If N has more than two distinct prime factors, our odds are even better. The method fails if N is a prime power, $N = p^{\alpha}$, but prime powers can be efficiently factored by other methods.

6.3.2 RSA

Does anyone care whether factoring is easy or hard? Well, yes, some people do.

The presumed difficulty of factoring is the basis of the security of the widely used RSA (for Rivest, Shamir, and Adleman) scheme for public key cryptography, which you may have used yourself if you have ever sent your credit card number over the internet.

The idea behind public key cryptography is to avoid the need to exchange a secret key (which might be intercepted and copied) between the parties that want to communicate. The enciphering key is public knowledge. But using the enciphering key to infer the deciphering key involves a prohibitively difficult computation. Therefore, Bob can send the enciphering key to Alice and everyone else, but only Bob will be able to decode the message that Alice (or anyone else) encodes using the key. Encoding is a "one-way function" that is easy to compute but very hard to invert.

(Of course, Alice and Bob could have avoided the need to exchange the public key if they had decided on a private key in their previous clandestine meeting. For example, they could have agreed to use a long random string as a one-time pad for encoding and decoding. But perhaps Alice and Bob never anticipated that they would someday need to communicate privately. Or perhaps they did agree in advance to use a one-time pad, but they have now used up their private key, and they are loath to reuse it for fear that an eavesdropper might then be able to break their code. Now they are too far apart to safely exchange a new private key; public key cryptography appears to be their most secure option.)

To construct the public key Bob chooses two large prime numbers p and q. But he does not publicly reveal their values. Instead he computes the product

$$N = pq. \tag{6.74}$$

Since Bob knows the prime factorization of N, he also knows the value of the Euler function $\varphi(N)$ – the number of number less than N that are coprime with N. In the case of a product of two primes it is

$$\varphi(N) = N - p - q + 1 = (p - 1)(q - 1), \tag{6.75}$$

(only multiples of p and q share a factor with N). It is easy to find $\varphi(N)$ if you know the prime factorization of N, but it is hard if you know only N.

Bob then pseudo-randomly selects $e < \varphi(N)$ that is coprime with $\varphi(N)$. He reveals to Alice (and anyone else who is listening) the value of N and e, but nothing else.

Alice converts her message to ASCII, a number $a < N$. She encodes the message by computing

$$b = f(a) = a^{e} (\text{mod } N), \tag{6.76}$$

which she can do quickly by repeated squaring. How does Bob decode the message?

Suppose that a is coprime to N (which is overwhelmingly likely if p and q are very large – anyway Alice can check before she encodes). Then

$$a^{\varphi(N)} \equiv 1 \pmod N \tag{6.77}$$

(Euler's theorem). This is so because the numbers less than N and coprime to N form a group (of order $\varphi(N)$) under mod N multiplication. The order of any group element must divide the order of the group (the powers of a form a subgroup). Since $\text{GCD}(e, \varphi(N) = 1$, we know that e has a multiplicative inverse $d = e^{-1} \bmod \varphi(N)$:

$$ed \equiv 1 \pmod{\varphi(N)}. \tag{6.78}$$

The value of d is Bob's closely guarded secret; he uses it to decode by computing:

$$\begin{aligned} f^{-1}(b) &= b^d \pmod N \\ &= a^{ed} \pmod N \\ &= a \cdot (a^{\varphi(N)})^{\text{integer}} \pmod N \\ &= a \pmod N. \end{aligned} \tag{6.79}$$

[**Aside:** How does Bob compute $d = e^{-1}$? The multiplicative inverse is a byproduct of carrying out the Euclidean algorithm to compute $\text{GCD}(e, \varphi(N)) = 1$. Tracing the chain of remainders from the bottom up, starting with $R_n = 1$:

$$\begin{aligned} 1 = R_n &= R_{n-2} - q_{n-1}R_{n-1} \\ R_{n-1} &= R_{n-3} - q_{n-2}R_{n-2} \\ R_{n-2} &= R_{n-4} - q_{n-3}R_{n-3} \\ &\mathit{etc.}\ldots \end{aligned} \tag{6.80}$$

(where the q_j's are the quotients), so that

$$\begin{aligned} 1 &= (1 + q_{n-1}q_{n-2})R_{n-2} - q_{n-1}R_{n-3} \\ 1 &= (-q_{n-1} - q_{n-3}(1 + q_{n-1}q_{n-2}))R_{n-3} \\ &\quad + (1 + q_{n-1}q_{n-2})R_{n-4}, \\ &\mathit{etc.}\ldots. \end{aligned} \tag{6.81}$$

Continuing, we can express 1 as a linear combination of any two successive remainders; eventually we work our way up to

$$1 = d \cdot e + q \cdot \varphi(N), \tag{6.82}$$

and identify d as $e^{-1} \pmod{\varphi(N)}$.]

Of course, if Eve has a superfast factoring engine, the RSA scheme is insecure. She factors N, finds $\varphi(N)$, and quickly computes d. In fact, she does not really need to factor N; it is sufficient to compute the order modulo N of the encoded message $a^e \pmod N$. Since e is coprime with $\varphi(N)$, the order of $a^e \pmod N$ is the same as the order of a (both elements generate the same *orbit*, or cyclic subgroup). Once the order $\text{Ord}(a)$ is known, Eve computes $\tilde{d}$ such that

$$\tilde{d}e \equiv 1 \pmod{\text{Ord}(a)} \tag{6.83}$$

so that

$$(a^e)^{\tilde{d}} \equiv a \cdot (a^{\text{Ord}(a)})^{\text{integer}} \pmod N \equiv a \pmod N, \tag{6.84}$$

and Eve can decipher the message. If our only concern is to defeat RSA, we run the Shor algorithm to find $r = \text{Ord}(a^e)$, and we needn't worry about whether we can use r to extract a factor of N or not.

How important are such prospective cryptographic applications of quantum computing? When fast quantum computers are readily available, concerned parties can stop using RSA, or can use longer keys to stay a step ahead of contemporary technology. However, people with secrets sometimes want their messages to remain confidential for a while (30 years?). They may not be satisfied by longer keys if they are not confident about the pace of future technological advances.

And if they shun RSA, what will they use instead? Not so many suitable one-way functions are known, and others besides RSA are (or may be) vulnerable to a quantum attack. So there really is a lot at stake. If fast large scale quantum computers become available, the cryptographic implications may be far reaching.

But while quantum theory taketh away, quantum theory also giveth; quantum computers may compromise public key schemes, but also offer an alternative: secure quantum key distribution, as discussed in Chapter 4.

6.4 Phase Estimation

There is an alternative way to view the factoring algorithm (due to Kitaev) that deepens our insight into how it works: we can factor because we can measure efficiently and accurately the eigenvalue of a certain unitary operator.

Consider $a < N$ coprime to N, let x take values in $\{0, 1, 2, \ldots, N-1\}$, and let $\boldsymbol{U}_a$ denote the unitary operator

$$\boldsymbol{U}_a : |x\rangle \to |ax \pmod N\rangle. \tag{6.85}$$

This operator is unitary (a permutation of the computational basis) because multiplication by a mod N is invertible.

If the order of a mod N is r, then

$$\boldsymbol{U}_a^r = \mathbf{1}. \tag{6.86}$$

It follows that all eigenvalues of $\boldsymbol{U}_a$ are rth roots of unity:

$$\lambda_k = e^{2\pi i k/r}, \quad k \in \{0, 1, 2, \ldots, r-1\}. \tag{6.87}$$

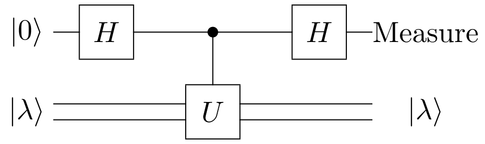
The corresponding eigenstates are

$$|\lambda_k\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i k j/r} |a^j x_0 (\text{mod } N)\rangle; \tag{6.88}$$

associated with each orbit of length r generated by multiplication by a, there are r mutually orthogonal eigenstates.

$\boldsymbol{U}_a$ is not hermitian, but its *phase* (the Hermitian operator that generates $\boldsymbol{U}_a$) is an observable quantity. Suppose that we can perform a measurement that projects onto the basis of $\boldsymbol{U}_a$ eigenstates, and determines a value λ_k selected equiprobably from the possible eigenvalues. Hence the measurement determines a value of k/r, as does Shor's procedure, and we can proceed to factor N with a reasonably high success probability. But how do we measure the eigenvalues of a unitary operator?

Suppose that we can execute the unitary $\boldsymbol{U}$ conditioned on a control bit, and consider the circuit:



Here $|\lambda\rangle$ denotes an eigenstate of $\boldsymbol{U}$ with eigenvalue λ ($\boldsymbol{U}|\lambda\rangle = \lambda|\lambda\rangle$). Then the action of the circuit on the control bit is

$$|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \to \frac{1}{\sqrt{2}}(|0\rangle + \lambda|1\rangle)$$

$$\to \frac{1}{2}(1+\lambda)|0\rangle + \frac{1}{2}(1-\lambda)|1\rangle. \tag{6.89}$$

Then the outcome of the measurement of the control qubit has probability distribution

$$\begin{aligned} \text{Prob}(0) &= \left|\frac{1}{2}(1+\lambda)\right|^2 = \cos^2(\pi\phi) \\ \text{Prob}(1) &= \left|\frac{1}{2}(1-\lambda)\right)|^2 = \sin^2(\pi\phi), \end{aligned} \tag{6.90}$$

where $\lambda = e^{2\pi i\phi}$.

As we have discussed previously (for example in connection with Deutsch's problem), this procedure distinguishes with certainty between the eigenvalues $\lambda = 1$ ($\phi = 0$) and $\lambda = -1$ ($\phi = 1/2$). But other possible values of λ can also be distinguished, albeit with less statistical confidence. For example, suppose the state on which $\boldsymbol{U}$ acts is a superposition of $\boldsymbol{U}$ eigenstates

$$\alpha_1|\lambda_1\rangle + \alpha_2|\lambda_2\rangle. \tag{6.91}$$

And suppose we execute the above circuit n times, with n distinct control bits. We thus prepare the state

$$\begin{aligned} &\alpha_1|\lambda_1\rangle\left(\frac{1+\lambda_1}{2}|0\rangle + \frac{1-\lambda_1}{2}|1\rangle\right)^{\otimes n} \\ +&\alpha_2|\lambda_2\rangle\left(\frac{1+\lambda_2}{2}|0\rangle + \frac{1-\lambda_2}{2}|1\rangle\right)^{\otimes n}. \end{aligned} \tag{6.92}$$

If $\lambda_1 \neq \lambda_2$, the overlap between the two states of the n control bits is exponentially small for large n; by measuring the control bits, we can perform the orthogonal projection onto the $\{|\lambda_1\rangle, |\lambda_2\rangle\}$ basis, at least to an excellent approximation.

If we use enough control bits, we have a large enough sample to measure Prob (0)= $\frac{1}{2}(1+\cos 2\pi\phi)$ with reasonable statistical confidence. By executing a controlled-($i\boldsymbol{U}$), we can also measure $\frac{1}{2}(1+\sin 2\pi\phi)$ which suffices to determine ϕ modulo an integer.

However, in the factoring algorithm, we need to measure the phase of $e^{2\pi ik/r}$ to exponential accuracy, which seems to require an exponential number of trials. Suppose, though, that we can efficiently compute high powers of $\boldsymbol{U}$ (as is the case for $\boldsymbol{U}_a$) such as

$$\boldsymbol{U}^{2^j}. \tag{6.93}$$

By applying the above procedure to measurement of $\boldsymbol{U}^{2^j}$, we determine

$$\exp(2\pi i 2^j\phi), \tag{6.94}$$

where $e^{2\pi i\phi}$ is an eigenvalue of $\boldsymbol{U}$. Hence, measuring $\boldsymbol{U}^{2^j}$ to one bit of accuracy is equivalent to measuring the jth bit of the eigenvalue of $\boldsymbol{U}$.

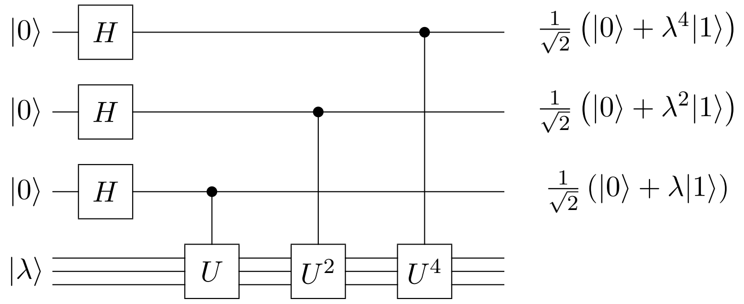
We can use this phase estimation procedure for order finding, and hence factorization. We invert eq. (6.88) to obtain

$$|x_0\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|\lambda_k\rangle; \tag{6.95}$$

each computational basis state (for $x_0 \neq 0$) is an equally weighted superposition of r eigenstates of $\boldsymbol{U}_a$.

Measuring the eigenvalue, we obtain $\lambda_k = e^{2\pi i k/r}$, with k selected from $\{0, 1\ldots, r-1\}$ equiprobably. If $r < 2^n$, we measure to $2n$ bits of precision to determine k/r. In principle, we can carry out this procedure in a computer that stores fewer qubits than we would need to evaluate the QFT, because we can attack just one bit of k/r at a time.

But it is instructive to imagine that we incorporate the QFT into this phase estimation procedure. Suppose the circuit



$|0\rangle$
H
$\frac{1}{\sqrt{2}}\left(|0\rangle + \lambda^4|1\rangle\right)$
$|0\rangle$
H
$\frac{1}{\sqrt{2}}\left(|0\rangle + \lambda^2|1\rangle\right)$
$|0\rangle$
H
$\frac{1}{\sqrt{2}}\left(|0\rangle + \lambda|1\rangle\right)$
$|\lambda\rangle$
U
U^2
U^4

acts on the eigenstate $|\lambda\rangle$ of the unitary transformation $\boldsymbol{U}$. The conditional $\boldsymbol{U}$ prepares $\frac{1}{\sqrt{2}}(|0\rangle + \lambda|1\rangle)$, the conditional $\boldsymbol{U}^2$ prepares $\frac{1}{\sqrt{2}}(|0\rangle + \lambda^2|1\rangle)$, the conditional $\boldsymbol{U}^4$ prepares $\frac{1}{\sqrt{2}}(|0\rangle + \lambda^4|1\rangle)$, and so on. We could perform a Hadamard and measure each of these qubits to sample the probability distribution governed by the jth bit of ϕ, where $\lambda = e^{2\pi i\phi}$. But a more efficient method is to note that the state prepared by the circuit is

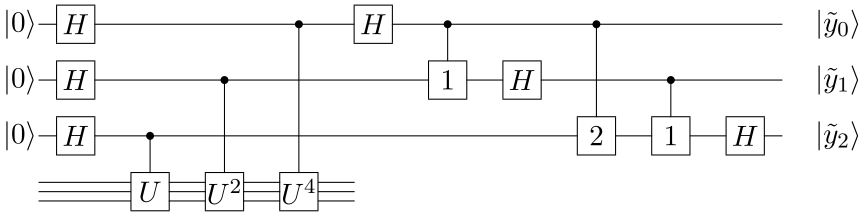
$$\frac{1}{\sqrt{2^m}}\sum_{y=0}^{2^m-1} e^{2\pi i\phi y}|y\rangle. \tag{6.96}$$

A better way to learn the value of ϕ is to perform the QFT$^{(m)}$, not the Hadamard $\boldsymbol{H}^{(m)}$, before we measure.

Considering the case $m = 3$ for clarity, the circuit that prepares and then Fourier analyzes the state

$$\frac{1}{\sqrt{8}}\sum_{y=0}^{7} e^{2\pi i\phi y}|y\rangle \tag{6.97}$$

is



This circuit very nearly carries out our strategy for phase estimation outlined above, but with a significant modification. Before we execute the final Hadamard transformation and measurement of $\tilde{y}_1$ and $\tilde{y}_2$, some conditional phase rotations are performed. It is those phase rotations that distinguish the QFT$^{(3)}$ from Hadamard transform $\boldsymbol{H}^{(3)}$, and they strongly enhance the reliability with which we can extract the value of ϕ.

We can understand better what the conditional rotations are doing if we suppose that $\phi = k/8$, for $k \in \{0, 1, 2 \ldots, 7\}$; in that case, we know that the Fourier transform will generate the output $\tilde{y} = k$ with probability one. We may express k as the binary expansion

$$k = k_2 k_1 k_0 \equiv k_2 \cdot 4 + k_1 \cdot 2 + k_0. \tag{6.98}$$

In fact, the circuit for the least significant bit $\tilde{y}_0$ of the Fourier transform is precisely Kitaev's measurement circuit applied to the unitary $\boldsymbol{U}^4$, whose eigenvalue is

$$(e^{2\pi i\phi})^4 = e^{i\pi k} = e^{i\pi k_0} = \pm 1. \tag{6.99}$$

The measurement circuit distinguishes eigenvalues ±1 perfectly, so that $\tilde{y}_0 = k_0$.

The circuit for the next bit $\tilde{y}_1$ is almost the measurement circuit for $\boldsymbol{U}^2$, with eigenvalue

$$(e^{2\pi i\phi})^2 = e^{i\pi k/2} = e^{i\pi(k_1 \cdot k_0)}. \tag{6.100}$$

Except that the conditional phase rotation has been inserted, which multiplies the phase by $\exp[i\pi(\cdot k_0)]$, resulting in $e^{i\pi k_1}$. Again, applying a Hadamard followed by measurement, we obtain the outcome $\tilde{y}_1 = k_1$ with certainty. Similarly, the circuit for $\tilde{y}_2$ measures the eigenvalue

$$e^{2\pi i\phi} = e^{i\pi k/4} = e^{i\pi(k_2 \cdot k_1 k_0)}, \tag{6.101}$$

except that the conditional rotation removes $e^{i\pi(\cdot k_1 k_0)}$, so that the outcome is $\tilde{y}_2 = k_2$ with certainty.

Thus, the QFT implements the phase estimation routine with maximal cleverness. We measure the less significant bits of ϕ first, and we exploit the information gained in the measurements to improve the reliability of our estimate of the more significant bits. Keeping this interpretation in mind, you will find it easy to remember the circuit for the QFT$^{(n)}$!

6.5 Hidden Subgroup Problem

Simon's problem and period finding are two black-box problems for which quantum computers provide exponential speedups. What else can quantum computers do? These two problems have a similar structure, and it is useful to recognize this common ground, because it suggests further generalizations.

More specifically, Simon's problem and period finding are both special cases of a problem that is naturally formulated in group-theoretic language: the Hidden Subgroup Problem (HSP). This is a black-box problem where we may regard the input to the function f to be an element of a group G which is mapped into a finite set X, where X may be chosen to be the set of m-bit strings:

$$f : G \to X = \{0,1\}^m. \tag{6.102}$$

The group G may be either finite or infinite, but we ordinarily assume it is finitely generated, that is, each element of G can be expressed as a product of a finite set of generating elements, where these generating elements may be used any number of times in the product, and in any order.

We are promised that the function f is constant and distinct on the cosets of a subgroup $H \subseteq G$. This means that

$$f(g_1) = f(g_2) \ \text{ iff } \ g_2^{-1}g_1 \in H \tag{6.103}$$

(that is, $g_1 = g_2 h$ for some $h \in H$). The problem is to find H — to list a set of elements of G that generate H. We may take the input size for the HSP to be an upper bound on $\log(|G/H|)$, the number of cosets (which is finite because X is finite).

The promise may restrict the hidden subgroup further by specifying additional properties of H. For example, in the case of Simon's problem,

$$G = \mathbb{Z}_2^n, \quad H = \mathbb{Z}_2 = \{0, a\}. \tag{6.104}$$

The group $\mathbb{Z}_2$ is the set $\{0,1\}$, where the group operation is addition modulo 2. A product group $G_1 \times G_2$ is defined as the group of pairs of elements

$$G_1 \times G_2 = \{(g_1, g_2) | g_1 \in G_1, g_2 \in G_2\}, \tag{6.105}$$

where the group operations are performed in parallel:

$$(g_1, g_2) \circ (g_1', g_2') = (g_1 g_1', g_2 g_2') \tag{6.106}$$

Thus, the elements of $\mathbb{Z}_2^n$ (the product of n $\mathbb{Z}_2$s) are n-bit strings of bits, where the group operation is bitwise XOR:

$$(x_{n-1}, .., x_0) \circ (y_{n-1}, .., y_0) = (x_{n-1} \oplus y_{n-1}, .., x_0 \oplus y_0) \tag{6.107}$$

Each element is its own inverse (i.e. is order 2).

The promise in Simon's problem is:

$$f(x) = f(y) \ \text{ iff } \ x \in \{0, a\} = H = \mathbb{Z}_2. \tag{6.108}$$

Here $\{0, a\}$ is isomorphic to $\mathbb{Z}_2$. The problem is to determine how this $\mathbb{Z}_2$ is embedded in $G = \mathbb{Z}_2^n$ — i.e., to find its generator a. The number of possible embeddings is exponential in n. The number of cosets (and so the number of possible outputs in the set X) is 2^{n-1}, and its log is the input size.

Another example is period finding, for which

$$G = \mathbb{Z} \ \text{ and } \ H = r\mathbb{Z} = \{rk, k \in \mathbb{Z}\}. \tag{6.109}$$

The group operation is addition, and we are promised that

$$f(x) = f(y) \ \text{ iff } \ x - y = r \cdot \text{integer} \in H \tag{6.110}$$

The problem is to find the generator of H, namely the period r. The number of cosets of H is $|G/H| = r$, and an upper bound on its log is the input size.

Classically, the HSP has query complexity $\Omega(\sqrt{|G/H|})$; we need to query this many times in order to get the same output in response to two different queries with reasonable probability. This is exponential in the input size — the problem is hard classically.

But for any finitely generated abelian group, the problem is easy quantumly! It can be solved (with high success probability) using $O(\text{polylog}|G/H|)$ queries, and $O(\text{polylog}|G/H|)$ additional computational steps.

Before we explain the algorithm, let's discuss another application:

6.5.1 Discrete Log Problem

Recall that if q is prime, then the group $\mathbb{Z}_q^*$ (multiplication mod q with elements $\{1, 2, ..., q-1\}$) is cyclic. This means that $\mathbb{Z}_q^*$ is generated by a single element a; Thus

$$\mathbb{Z}_q^* = \{a, a^2, a^3, .., a^{q-1} = e\}. \tag{6.111}$$

Therefore any element $x \in \mathbb{Z}_q^*$ can be expressed in a unique way as the modular exponential

$$x = a^y \pmod{q} \quad \text{where } y \in \{0, 1, 2, .., q-2\}. \tag{6.112}$$

The discrete log (mod q) with base a is the inverse of this function':

$$x = a^y \ (\text{mod } q) \ \leftrightarrow \ y = \text{dlog}_{q,a}(x) \tag{6.113}$$

A discrete log can be defined this way for any cyclic group and any generating element of the group.

Example: $q = 7$, $\mathbb{Z}_q^* = \{1, 2, ..., 6\}$, $a = 5$ is the generator:

$y =$	0	1	2	3	4	5
$x = 5^y$ (mod 7)	1	5	4	6	2	3

.

The inverse function is

$x =$	1	2	3	4	5	6
$y = \text{dlog}_{7,5}(x) =$	0	4	5	2	1	3

.

The modular exponential is easy to compute classically (by repeated squaring), but the discrete log seems to be hard to compute — the modular exponential is a candidate one-way function. It is hard to invert because a^x seems to jump about in $\mathbb{Z}_q^*$ haphazardly as x varies (for at least some values of q).

There are applications of this one-way function in cryptography; for example:

6.5.2 Diffie-Hellman key exchange

This protocol's security rests on the presumed hardness of computing the discrete logarithm. The objective is for Alice and Bob to generate a shared secret key that is not known by their adversary Eve.

- A prime number q and a generating element $a \in \mathbb{Z}_q^*$ are publicly announced.
- Alice generates a random element $x \in \mathbb{Z}_q^*$ and keeps it secret. Bob generates a random element $y \in \mathbb{Z}_q^*$ and keeps it secret.
- Alice computes and announces a^x (mod q). Bob computes and announces a^y (mod q).

- Alice computes $(a^y)^x = a^{xy} \pmod q$. Bob computes $(a^x)^y = a^{xy} \pmod q$. This is their final shared key.

Alice and Bob can both compute the key because the modular exponential can be evaluated efficiently. The protocol is expected to be secure because even when a^x and a^y (but not x or y) are known, it is hard to compute a^{xy}. Of course, if Eve can compute the discrete log, she could break the protocol. E.g. knowing a^x and a^y she would be able to compute x and then compute $(a^y)^x$.

But a quantum computer can evalutate a discrete log by solving a HSP! Here is how. We would like to find

$$r = \mathrm{dlog}_{q,a}(x) \tag{6.114}$$

where the value of r is such that $x = a^r \pmod q$. We consider the function

$$f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}_q^*, \quad f(y_1, y_2) = a^{y_1} x^{-y_2} \pmod q. \tag{6.115}$$

When does f map two different inputs to the same output?

$$f(y_1, y_2) = a^{y_1 - r y_2} \pmod q = f(z_1, z_2) = a^{z_1 - r z_2} \pmod q \tag{6.116}$$

iff $(y_1 - z_1) - r(y_2 - z_2) = 0 \pmod{q-1}$. This means that we may think of the input to f as an element of the additive group $G = \mathbb{Z} \times \mathbb{Z}$ where f is constant and distinct on the cosets of

$$H = \{(y_1, y_2) | y_1 = r y_2 \pmod{q-1}\}. \tag{6.117}$$

H is generated by the elements $(r, 1), (q-1, 0)$ so if we find generators, we determine r.

6.5.3 Finding abelian hidden subgroups

Primal lattice and dual lattice

For a HSP problem with finitely generated abelian $\tilde{G}$, we may consider without loss of generality a corresponding problem with $G = \mathbb{Z}^n$, since there is a homomorphism mapping G onto $\tilde{G}$. In general, it is useful that we can give geometrical interpretations to G and H. G is the n-dimensional hypercubic lattice, containing all ordered n-tuples of integers. The subgroup H can be regarded as a sublattice of $\mathbb{Z}^n$. This sublattice is spanned by a set of n linearly independent vectors $\{v_1, v_2, ..v_n\}$, each an element of $\mathbb{Z}^n$ (i.e. with integer entries). A general element x of H is a linear combination of the generating vectors

$$x = \sum_{a=1}^{n} \alpha_a v_a. \tag{6.118}$$

We may construct an $n \times n$ generator matrix for the lattice H whose rows are the generating vectors:

$$M = \begin{pmatrix} v_1 \\ v_2 \\ . \\ v_n \end{pmatrix} \quad \text{and} \quad H = \{x = \alpha M, \ \alpha \in \mathbb{Z}^n\}, \tag{6.119}$$

where α is the row vector $\alpha = (\alpha_1, \alpha_2, .., \alpha_n)$. For fixed H, the generator matrix is not unique. We may make the replacement $M \mapsto RM$ where R is an invertible integral matrix with $\det R = \pm 1$ (so that R^{-1} is also integral). Both M and RM are generators of the same lattice.

The quotient space G/H may be called the unit cell of the lattice. It contains all the distinct ways to shift the lattice H by an element of G. We may say that $|G/H|$ is the volume of the unit cell, the number of points it contains. Note that

$$|G/H| = \det M \tag{6.120}$$

(the linear transformation M inflates the cube $\{0,1\}^n$ to a region of volume $\det M$).

Corresponding to the integral lattice H is its dual lattice, denoted $H^\perp$. The elements of $H^\perp$ are points in $\mathbb{R}^n$ that are orthogonal to all the vectors in H, modulo integers:

$$H^\perp = \{k \in \mathbb{R}^n \text{ such that } k \cdot x \in \mathbb{Z} \text{ for all } x \in H\}. \tag{6.121}$$

Equivalently, $\exp(2\pi i k \cdot x) = 1$ for $k \in H^\perp$ and $x \in H$. $H^\perp$ is also a lattice (i.e. its elements are the span of a set of generating vectors with integer coefficients), but the components are not necessarily integer (although they are rational numbers). If $H^\perp$ is generated by vectors $u_1, u_2, ..u_n$ then its generating matrix is

$$M^\perp = \begin{pmatrix} u_1 \\ u_2 \\ \cdot \\ u_n \end{pmatrix} \quad \text{and} \quad H^\perp = \{k = \beta M^\perp,\ \beta \in \mathbb{Z}^n\} \tag{6.122}$$

We can choose the basis for the dual lattice such that $u_a v_b^T = \delta_{ab}$, in which case $M^\perp M^T = I$. That means that, once we have found $M^\perp$, an easy computation determines M (matrix inversion of transpose of $M^\perp$). In the quantum algorithm for the abelian HSP, the quantum computation determines the generators of $H^\perp$ (i.e. the matrix $M^\perp$) and then finding the generators of H is easy (by matrix inversion).

The efficient solution to the problem makes use "Fourier sampling" — after evaluating the function f on a coherent superposition of values of G, we perform (an approximation to) the Fourier transform over the group G, and then measure. This procedure enables us to sample nearly uniformly from $H^\perp$. Only a modest number of samples are needed to determine $H^\perp$, and hence H, which solves the problem.

For example, in the case of period finding, we have $G = \mathbb{Z}$, $H = r\mathbb{Z} = \{x = r\alpha, \alpha \in \mathbb{Z}\}$ and $H^\perp = \{k = \beta/r, \beta \in \mathbb{Z}\}$. In the quantum algorithm, we are promised that $r \le R$; thus we can sample from $H^\perp$ using the Fourier transform for the finite group $\mathbb{Z}_N$ rather than the infinite group $\mathbb{Z}$, where $N \ge R^2$. Fourier sampling then provides sufficient accuracy to determine an element β/r of $H^\perp$ with high success probability. After a few samples we can determine $1/r$, the generator of $H^\perp$, and hence r, the generator of H. We want to extend this idea from subgroups of $\mathbb{Z}$ to subgroups of $\mathbb{Z}^n$.

So, to solve the general abelian HSP, we Fourier transform over $\mathbb{Z}_N^n$ instead of $\mathbb{Z}^n$,for some sufficiently large N. And to keep the discussion simple at first, let's suppose that H is actually a subgroup of the finite group $\mathbb{Z}_N^n$, rather than of $\mathbb{Z}^n$.

H-invariant coset state and Fourier sampling from the dual lattice

As in the period finding algorithm we query the black box with

$$\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle \quad \text{to obtain} \quad \frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle \otimes |f(x)\rangle, \tag{6.123}$$

where f is constant and distinct on the cosets of $H \subseteq G$. Were we to measure the output register, obtaining $f(x_0)$, we would prepare in the input register the uniform superposition of elements in the same coset as x_0; which is

$$|H, x_0\rangle = \frac{1}{\sqrt{|H|}} \sum_{x \in H} |x + x_0\rangle. \tag{6.124}$$

This "coset state" has an important property: it is H-invariant. We may consider the unitary transformation U_y associated with an element $y \in G$ whose action is

$$U_y |H, x_0\rangle = \frac{1}{\sqrt{|H|}} \sum_{x \in H} |x + x_0 + y\rangle, \tag{6.125}$$

and we may reparamaterize the sum over x replacing $x \mapsto x' - y$, thus obtaining:

$$\frac{1}{\sqrt{|H|}} \sum_{x' \in H} |x' + x_0\rangle = |H, x_0\rangle. \tag{6.126}$$

To appreciate the significance of H-invariance, note that if $|\psi\rangle$ obeys $U|\psi\rangle = |\psi\rangle$, then $\tilde{U}|\tilde{\psi}\rangle = VUV^{-1}|\tilde{\psi}\rangle = |\tilde{\psi}\rangle$ where $|\tilde{\psi}\rangle = V|\psi\rangle$. Now apply this identity to $U = U_y$ where V is the Fourier transform

$$V : |x\rangle \mapsto \frac{1}{\sqrt{|G|}} \sum_{k \in G^\perp} e^{2\pi i k \cdot x/N} |k\rangle, \tag{6.127}$$

$$V^{-1} : |k\rangle \mapsto \frac{1}{\sqrt{|G|}} \sum_{x \in G} e^{-2\pi i k \cdot x/N} |x\rangle. \tag{6.128}$$

We then find $\tilde{U}_y |k\rangle = e^{-2\pi i k \cdot y/N} |k\rangle$, as follows:

$$\begin{aligned}
|k\rangle &\xrightarrow{V^{-1}} \frac{1}{\sqrt{|G|}} \sum_{x \in G^\perp} e^{-2\pi i k \cdot x/N} |x\rangle && (6.129)\\
&\xrightarrow{U_y} \frac{1}{\sqrt{|G|}} \sum_{x \in G^\perp} e^{-2\pi i k \cdot x/N} |x + y\rangle && (6.130)\\
&= e^{2\pi i k \cdot y/N} \frac{1}{\sqrt{|G|}} \sum_{x' \in G^\perp} e^{-2\pi i k \cdot x'/N} |x'\rangle && (6.131)\\
&\xrightarrow{V} e^{2\pi i k \cdot y/N} |k\rangle. && (6.132)
\end{aligned}$$

Therefore, the state $|k\rangle$ is an eigenstate of $\tilde{U}_y$ with eigenvalue 1 iff $k \cdot y/N =$ integer — or for $y \in H$ iff $k/N \in H^\perp$. Thus, if a state is H-invariant, then in the Fourier basis its expansion contains the state $|k\rangle$ with a non-zero coefficient only if $k/N \in H^\perp$.

More explicitly, we compute

$$\begin{aligned}
V^{-1} : |H, x_0\rangle &= \frac{1}{\sqrt{|H|}} \sum_{x \in H} |x + x_0\rangle && (6.133)\\
&\mapsto \frac{1}{\sqrt{|H||G|}} \sum_{x \in H} \sum_{k \in G^\perp} e^{2\pi i k \cdot (x + x_0)/N} |k\rangle. && (6.134)\\
& && (6.135)
\end{aligned}$$

Because of H-invariance, only $k/N \in H^\perp$ survives in the sum over $G^\perp$, and for such k,

$e^{2\pi i(k\cdot x)/N} = 1$ so we obtain

$$\frac{1}{\sqrt{|H^\perp|}} \sum_{k\in H^\perp} e^{2\pi i k\cdot x_0/N} |k\rangle \tag{6.136}$$

Therefore if we Fourier sample — i.e. Fourier transform and then measure — the probability distribution that governs the outcome is the uniform distribution on $H^\perp$. Once we have sampled from $H^\perp$ enough times, with high probability a generating set for $H^\perp$ will be found.

Query complexity

How many samples are enough (assuming now that G is finite — e.g. $G = \mathbb{Z}_N^n$ — and $H \subseteq G$)? Suppose K is a group (either abelian or not), and m elements of K are chosen uniformly at random. If these m elements do not generate K, then they must be contained in some maximal proper subgroup $S \subset K$. (Proper means S is smaller than K, and maximal means we cannot add another element of K to S without generating all of K.) Any proper subgroup has order $|S| \le |K|/2$, because the order of the subgroup must divide the order of K, and the probability that all m elements are in S is

$$\text{Prob(all } m \text{ in } S) = \left(\frac{|S|}{|K|}\right)^m ; \tag{6.137}$$

therefore the probability that the m elements generate K is

$$\text{Prob(}m \text{ elements generate } K) \ge 1 - \sum_{S\in\max} \left(\frac{|S|}{|K|}\right)^m \ge 1 - (\#\ \max)2^{-m}, \tag{6.138}$$

where the sum is over maximal proper subgroups $\{S\}$, and where $(\#\ \max)$ denotes the total number of maximal proper subgroups.

If K is abelian, we can count the maximal proper subgroups. S is a sublattice of K and if S is a maximal proper subgroup, then its dual lattice $S^\perp$ contains a vector not in $K^\perp$. There is only one such (linearly independent) vector if S is maximal, for if there were two then we could remove one, obtaining a smaller $S^\perp$ and hence a larger proper subgroup S. Any nontrivial vector not in $K^\perp$ determines such a subgroup, so there are $|G/K^\perp| - 1$ choices (where e.g. $G = \mathbb{Z}_N^n$), and therefore

$$\text{Prob(}m \text{ elements generate } S) \ge 1 - 2^{-m}|G/K^\perp|. \tag{6.139}$$

In the case of the hidden subgroup problem where $H \subseteq G = \mathbb{Z}_N^n$, we are sampling $K = H^\perp$ and $|G/K^\perp|$ becomes $|G/H|$, the number of cosets. To have constant success probability, then, we choose m such that e.g.

$$2^{-m}|G/K^\perp| < \frac{1}{2}, \tag{6.140}$$

or $m - 1 > \log |G/H|$ (compare this with the conclusion for Simon's problem). Since $|G/H| < N^n$; it suffices if $m = O(n \log N)$.

How large should N be? For period finding with $r \le R$, choosing $N \ge R^2$ provided adequate precision for finding r. For an integral lattice with generator matrix M, its inverse matrix (transpose of $M^\perp$) has entries that can be expressed as integer$/\det M$, where $\det M = |G/H|$, the number of cosets. In the formulation of the HSP, we are provided with an upper bound $|G/H| \le R$, and N needs to be large enough to point to a unique rational number with denominator $\le R$ with reasonable success probability.

In our discussion of period finding ($H^\perp = \mathbb{Z}/r$) we noted that Fourier sampling yields a rational number y/N close to integer$/r$ with high probability:

$$\sum_k \text{Prob}\left(\left|\frac{y}{N} - \frac{k}{r}\right| \le \frac{1}{2N}\right) \ge \frac{4}{\pi^2}, \tag{6.141}$$

so that choosing $N \ge R^2$ was good enough to determine a rational number with denominator $< R$. If we fix the desired accuracy δ, then the part of the distribution lying outside the peaks decreases as N increases (an exercise):

$$\text{Prob}\left(\forall k \left|\frac{y}{N} - \frac{k}{r}\right| > \delta\right) \le \frac{1}{N\delta}, \tag{6.142}$$

The peak of the Fourier transform sharpens with increasing N, so that the prob of lying outside all peaks with half width δ scales like $1/N$.

When we sample $H^\perp$, we find an n-component vector, where each component should be determined to accuracy $1/R^2$ (where $|G/H| < R$). The probability of success in finding all n-components to this accuracy is

$$\text{Prob(success)} \ge \left(1 - \frac{1}{N\delta}\right)^n \tag{6.143}$$

and the probability of being successful in each of m consecutive samplings is

$$\text{Prob(success } m \text{ times)} \ge \left(1 - \frac{1}{N\delta}\right)^{nm}. \tag{6.144}$$

For $\delta = 1/R^2$, the success probability is a constant for

$$\frac{mnR^2}{N} < \text{constant} \quad \text{or} \quad N = O(mnR^2). \tag{6.145}$$

Since $m = O(n \log N)$ samples are sufficient to find generators of $H^\perp$, we conclude it suffices to choose N to be

$$N = O(n^2R^2 \log N) = O(n^2R^2 \log(nR)) \tag{6.146}$$

This is good enough to determine $H^\perp$ in $m = O(n \log N) = O(n \log(nR))$ queries, and the generators of H are found by inverting the matrix $M^\perp$ that generates $H^\perp$.

The algorithm is efficient: both the number of queries and the number of steps in the quantum Fourier transform are polylog in (the upper bound on) the number of cosets $|G/H|$.

6.6 Quantum Searching

For the hardest instances of NP-hard problems, no better method is known than exhaustive search for a solution. For example, suppose that for some efficiently computable Boolean function

$$f : \{0,1\}^n \longrightarrow \{0,1\}; \tag{6.147}$$

we wish to determine whether there is an x such that $f(x) = 1$. We could search for a solution by trying all of the $N = 2^n$ possible values of the input x, but that might require time $O(N\text{polylog}N)$ — assuming we can evaluate f in time $O(\text{polylog}N)$. That's very slow, yet if f has no structure that we know how to exploit, we might not know how to do better.

We can model this situation in the black box setting. Suppose we are promised that the function evaluated by the box has the form

$$f_w(x) = \begin{cases} 0 & x \neq w, \\ 1 & x = w, \end{cases} \tag{6.148}$$

where $x \in \{0, 1, 2.., N-1\}$ for some unknown w. Our task is to find w, the *marked string.* Classically, we'll need to query the box more than $N/2$ times to find w with success probability above $1/2$. This is a black-box version of an NP-hard problem, where there is a unique witness accepted by a circuit, but the problem has no structure, so there is no better option than exhaustive searching.

Now we ask, can exhaustive search be done faster on a quantum computer? The answer is yes, using *Grover's algorithm.* With quantum queries, we can find the marked string using $O(\sqrt{N})$ queries. Thus we can solve NP-hard problems by exhaustive search in time $O(\sqrt{N}\text{polylog}N)$.

We say that Grover's algorithm achieves a *quadratic speedup* relative to exhaustive search on a classical computer. Though the speedup is only quadratic rather than exponential, Grover's algorithm is interesting because of its broad applicability. And it is rather remarkable: in effect we can interrogate N potential witnesses by asking $O(\sqrt{N})$ questions.

In the quantum setting, the black box applies the unitary

$$U_w : |x\rangle \otimes |y\rangle \mapsto |x\rangle \otimes |y \oplus f_w(x)\rangle, \tag{6.149}$$

where $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$. By the standard trick, U_w becomes a *phase oracle*:

$$U_w : |x\rangle \otimes |-\rangle \mapsto (-1)^{f_w(x)}|x\rangle \otimes |-\rangle \tag{6.150}$$

where

$$(-1)^{f_w(x)} = \begin{cases} 1 & x \neq w, \\ -1 & x = w. \end{cases} \tag{6.151}$$

Ignoring the output register (which is unaffected by U_w), we can express U_w acting on input as

$$U_w = I - 2|w\rangle\langle w|. \tag{6.152}$$

We can express a general n-qubit state $|\psi\rangle$ as

$$U_w : |\psi\rangle = a|w\rangle + b|\psi^\perp\rangle \mapsto -a|w\rangle + b|\psi^\perp\rangle, \tag{6.153}$$

where $\langle w|\psi^\perp\rangle = 0$. That is, we resolve $|\psi\rangle$ into a component along $|w\rangle$ and a component in the hyperplane orthogonal to $|w\rangle$. U_w induces a reflection of the vector $|\psi\rangle$ about this hyperplane.

The first step in Grover's algorithm is to prepare the uniform superposition of all values of x:

$$|s\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N-1} |x\rangle; \tag{6.154}$$

this state $|s\rangle$ has overlap with the marked string $|w\rangle$

$$\langle w|s\rangle = \frac{1}{\sqrt{N}}. \tag{6.155}$$

The next step is to apply the *Grover iteration* many times in succession, where each iteration enhances the overlap of the quantum superposition with the marked state $|w\rangle$, while suppressing the amplitude for each $|x\rangle$ with $x \neq w$. This iteration is

$$U_{\text{Grover}} = U_s U_w \tag{6.156}$$

where U_w is the query and

$$U_s = 2|s\rangle\langle s| - I \tag{6.157}$$

reflects a vector about the axis determined by $|s\rangle$. Note that U_s is easy to construct as a quantum circuit. It can be expressed as

$$U_s = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} \tag{6.158}$$

since $H^{\otimes n}$ maps $|s\rangle$ to $|0\rangle$, where H is the single qubit Hadamard gate. Furthermore a multiply controlled Z gate can be formed from a multiply controlled-not gate $\Lambda^{n-1}(X)$ where the target qubit has a Hadamard before and after the gate, and we know that $\Lambda^{n-1}(X)$ can be constructed from $O(n)$ Toffoli gates. Finally, we can conjugate by $X^{\otimes n}$ so the phase gate is triggered by $|00..0\rangle$ rather than $|11..1\rangle$. Thus U_s is realized by a circuit of size $O(\log N)$.

What does U_{Grover} do? It preserves the plane spanned by $|s\rangle$ and $|w\rangle$, so we may confine our attention to that plane. First, U_w reflects $|s\rangle$ about the axis $|w^{\perp}\rangle$ (the vector $\perp$ to $|w\rangle$ in the span of $|s\rangle$ and $|w\rangle$). Then U_s reflects $U_w|s\rangle$ about the axis $|s\rangle$. The net effect of U_{Grover}, then, is a counterclockwise rotation in the plane by the angle 2θ, where θ is initial angle between $|s\rangle$ and $|w\rangle$. Each time we repeat the Grover iteration the state vector rotates further in the counterclockwise direction by 2θ.

The initial angle θ between $|s\rangle$ and $|w^{\perp}\rangle$ is given by $\sin\theta = \langle w|s\rangle = 1/\sqrt{N}$. For $N \gg 1$, then, we have $\theta = 1/\sqrt{N} + O(1/N^{3/2})$. If we repeat the Grover iteration T times, then the vector is rotated away from the $|w^{\perp}\rangle$ axis by $(2T+1)\theta$.

We may choose T such that $(2T+1)\theta = \pi/2 + \delta$ where $|\delta| \leq \theta/2 \approx \frac{1}{2\sqrt{N}}$. Then if we measure in the computational basis, we find the outcome $|w\rangle$ with probability $\text{Prob}(w) = \cos^2\delta \geq 1 - \delta^2 \geq 1 - \frac{1}{4N}$. Thus we find $|w\rangle$ with success probability close to 1 using $T \approx \frac{\pi}{4\theta} \approx \frac{\pi}{4}\sqrt{N}$ Grover iterations, making use of T quantum queries to the black box. This is Grover's quadratic speedup.

Suppose now that there are r marked states, where r is known. Classically, with each query the probability of finding a solution (w_i such that $f(w_i) = 1$) is r/N, so we need $O(N/r)$ queries to find a solution with constant success probability. Quantumly, the uniform superposition of the marked states

$$|\text{marked}\rangle = \frac{1}{\sqrt{r}} \sum_{i=1}^{r} |w_i\rangle \tag{6.159}$$

has overlap with $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$

$$\langle \text{marked}|s\rangle = \sqrt{\frac{r}{N}} = \sin\theta \tag{6.160}$$

and the Grover iteration again rotates by 2θ in the plane spanned by $|s\rangle$ and $|\text{marked}\rangle$ (because the query reflects about the axis perpendicular to $|\text{marked}\rangle$).

As above, then, for $N/r \gg 1$, we achieve success probability $\text{Prob} = 1 - O(r/N)$

in $T \approx \pi/4\sqrt{N/r}$ queries. Again, the speedup is quadratic: The number of quantum queries needed to find a solution is

$$\#\text{quantum queries} = O(\sqrt{\#\text{classical queries}}). \tag{6.161}$$

What if r is *not* known a priori? As a function of the number of queries the success probability oscillates, where the period of the oscillation is $T \approx \pi/2\sqrt{N/r}$. If we choose T uniformly at random in the interval $T = \{0, 1, 2, .., T_{\max} \approx \pi/4\sqrt{N}\}$, then if there is a solution ($r > 0$), a solution will be found with Prob $\geq 1/2 + O(1/N)$. If we repeat m times, sampling a different random value of T each time, we will find a solution apart from a small failure probability $\approx 2^{-m}$. Therefore, we can use Grover's algorithm to solve a decision problem in NP with high success probability, in

$$\text{time} = O(\sqrt{N}\text{polylog}N), \tag{6.162}$$

since we can compute the circuit that evaluates $f(x)$ in (classical or quantum) time $O(\text{polylog}N)$.

6.6.1 Generalized Search

In some cases, the problem may have structure that can be exploited to search faster for a solution. In that case, some strings are better candidates than others to be solutions, and so we ought to be able to search more efficiently by spending more time testing likely solutions instead of less likely ones.

For the case of Grover's exhaustive search of a function without any apparent structure, we started the algorithm by preparing

$$|s\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle, \tag{6.163}$$

which has overlap $\sin\theta = 1/\sqrt{N}$ with the solution $|w\rangle$, and hence would yield the uniform distribution on x if we measured in the computational basis. For a function with structure we may be able to construct an efficiently computable unitary U such that $U|0\rangle = \sin\theta|w\rangle + \cos\theta|\psi^{\perp}\rangle$ where $\sin^2\theta > 1/N$. In that case, we can conduct Grover's algorithm with $H^{\otimes n}$ replaced by U, and U_s replaced by

$$\tilde{U}_s = U(2|0\rangle\langle 0| - I)U^{\dagger}. \tag{6.164}$$

Thus $\tilde{U}_s$ reflects in the axis $U|0\rangle$ rather than $|s\rangle$. The analysis of the algorithm is the same as before, and in the case where there exists a unique solution,we can find it with high probability in $T \approx \frac{\pi}{4\theta} < \frac{\pi}{4}\sqrt{N}$ queries.

Specifically, because of the structure of the function, we might be able to exclude all except $M < N$ inputs as potential solutions. Then, classically, we could find the solution in $O(M)$ queries, while quantumly only $O(\sqrt{M})$ queries suffice, if we can construct U such that

$$U|0\rangle = \frac{1}{\sqrt{M}}\sum_{i=1}^{M}|x_i\rangle, \tag{6.165}$$

the uniform superposition of the candidate solutions.

For example, suppose that classical search for a solution can be accelerated by a *classical heuristic* — that is, a function g that takes a randomly generated *seed* r in a set R to a trial solution:

$$g : r \mapsto g(r) \quad \text{where } r \in R. \tag{6.166}$$

The heuristic is useful if trial solutions generated by the heuristic are more likely to be accepted than trial solutions chosen uniformly at random

$$\left\langle \frac{\#\text{ of soln. in } g(R)}{|R|} \right\rangle > \left\langle \frac{\text{total }\#\text{ of soln.}}{N} \right\rangle, \tag{6.167}$$

where the bracket $\langle\cdot\rangle$ indicates the expectation value evaluated for a probability distribution on black-box functions. Then the number of classical queries to find a solution, using the heuristic, with constant success probability is

$$T_{\text{classical}} = O\left(\left\langle \frac{|R|}{\#\text{ of soln. in } g(R)} \right\rangle\right). \tag{6.168}$$

To exploit the heuristic in quantum searching, we apply Grover's algorithm to searching in the space of seeds instead of the full search space. The heuristic is realized as an efficiently computable unitary:

$$|r\rangle \otimes |0\rangle \mapsto |r\rangle \otimes |g(r)\rangle. \tag{6.169}$$

We can query the box with $|g(r)\rangle$ and then run the evaluation of g backwards to erase garbage:

$$\begin{aligned} |r\rangle \otimes |0\rangle \otimes |y\rangle \mapsto |r\rangle \otimes |g(r)\rangle \otimes |y\rangle \mapsto |r\rangle \otimes |g(r)\rangle \otimes |y \oplus f(g(r))\rangle \\ \mapsto |r\rangle \otimes |0\rangle \otimes |y \oplus f(g(r))\rangle \end{aligned} \tag{6.170}$$

This composite oracle can be consulted to search R for a state marked by the function $f \circ g$ (i.e. for a state marked by f in $g(R)$, the range of g). The number of quantum queries used is

$$T_{\text{quantum}} = O\left(\left\langle \sqrt{\frac{|R|}{\#\text{ of soln. in } g(R)}} \right\rangle\right) \tag{6.171}$$

(for each black box there is a quadratic speed up). Furthermore, the square root function is *concave*: $\langle\sqrt{F}\rangle = \sum_i p_a\sqrt{p_a} \leq \sqrt{\sum_i p_a F_a} = \sqrt{\langle F\rangle}$ where $\{p_a\}$ is a probability distribution, and here F_a represents the classical query complexity for a black box function labeled by a. Thus,

$$T_{\text{quantum}} \leq O\left(\sqrt{\left\langle \frac{|R|}{\#\text{ of soln. in } g(R)} \right\rangle}\right) = O(\sqrt{T_{\text{classical}}}). \tag{6.172}$$

The speedup is quadratic, as for unstructured search.

6.7 The Grover Algorithm Is Optimal

Grover's quadratic quantum speedup for exhaustive search is already interesting and potentially important, but surely with more cleverness we can do better, can't we? No, it turns out that we can't. Grover's algorithm provides the fastest possible quantum search of an unsorted database, if "time" is measured according to the number of queries of the oracle.

Considering the case of a single marked state $|\omega\rangle$, let $\boldsymbol{U}(\omega, T)$ denote a quantum circuit that calls the oracle T times. We place *no* restriction on the circuit aside from specifying the number of queries; in particular, we place no limit on the number of quantum gates. This circuit is applied to an initial state $|\psi(0)\rangle$, producing a final state

$$|\psi_\omega(t)\rangle = \boldsymbol{U}(\omega, T)|\psi(0)\rangle. \tag{6.173}$$

Now we are to perform a measurement designed to distinguish among the N possible values of ω. If we are to be able to perfectly distinguish among the possible values, the states $|\psi_\omega(t)\rangle$ must all be mutually orthogonal, and if we are to distinguish correctly with high probability, they must be nearly orthogonal.

Now, if the states $\{|\psi_\omega\rangle$ are an orthonormal basis, then, for any fixed normalized vector $|\varphi\rangle$,

$$\sum_{\omega=0}^{N-1} \parallel |\psi_\omega\rangle - |\varphi\rangle \parallel^2 \geq 2N - 2\sqrt{N}. \tag{6.174}$$

(The sum is minimized if $|\varphi\rangle$ is the equally weighted superposition of all the basis elements, $|\varphi\rangle = \frac{1}{\sqrt{N}} \sum_\omega |\psi_\omega\rangle$, as you can show by invoking a Lagrange multiplier to perform the constrained extremization.) Our strategy will be to choose the state $|\varphi\rangle$ suitably so that we can use this inequality to learn something about the number T of oracle calls.

Our circuit with T queries builds a unitary transformation

$$\boldsymbol{U}(\omega, T) = \boldsymbol{U}_\omega \boldsymbol{U}_T \boldsymbol{U}_\omega \boldsymbol{U}_{T-1} \dots \boldsymbol{U}_\omega \boldsymbol{U}_1, \tag{6.175}$$

where $\boldsymbol{U}_\omega$ is the oracle transformation, and the $\boldsymbol{U}_t$'s are arbitrary non-oracle transformations. For our state $|\varphi(T)\rangle$ we will choose the result of applying $\boldsymbol{U}(\omega, T)$ to $|\psi(0)\rangle$, except with each $\boldsymbol{U}_\omega$ replaced by $\boldsymbol{I}$; that is, the same circuit, but with all queries submitted to the "empty oracle." Hence,

$$|\varphi(T)\rangle = \boldsymbol{U}_T \boldsymbol{U}_{T-1} \dots \boldsymbol{U}_2 \boldsymbol{U}_1 |\psi(0)\rangle, \tag{6.176}$$

while

$$|\psi_\omega(T)\rangle = \boldsymbol{U}_\omega \boldsymbol{U}_T \boldsymbol{U}_\omega \boldsymbol{U}_{T-1} \dots \boldsymbol{U}_\omega \boldsymbol{U}_1 |\psi(0)\rangle. \tag{6.177}$$

To compare $|\varphi(T)\rangle$ and $|\psi_\omega(T)\rangle$, we appeal to our previous analysis of the effect of errors on the accuracy of a circuit, regarding the ω oracle as an "erroneous" implementation of the empty oracle. The norm of the error vector in the t-th step is

$$\begin{aligned} \parallel |E(\omega, t)\rangle \parallel &= \parallel (\boldsymbol{U}_\omega - \boldsymbol{I})|\varphi(t)\rangle \parallel \\ &= 2|\langle\omega|\varphi(t)\rangle|, \end{aligned} \tag{6.178}$$

since $\boldsymbol{U}_\omega = \boldsymbol{I} - 2|\omega\rangle\langle\omega|$. After T queries we have

$$\parallel |\psi_\omega(T)\rangle - |\varphi(T)\rangle \parallel \leq 2 \sum_{t=1}^{T} |\langle\omega|\varphi(t)\rangle|. \tag{6.179}$$

From the identity

$$\left(\sum_{t=1}^{T} c_t\right)^2 + \frac{1}{2}\sum_{s,t=1}^{T}(c_s - c_t)^2$$
$$= \sum_{s,t=1}^{T}\left(c_t c_s + \frac{1}{2}c_s^2 - c_t c_s + \frac{1}{2}c_t^2\right) = T\sum_{t=1}^{T} c_t^2, \tag{6.180}$$

we obtain the inequality

$$\left(\sum_{t=1}^{T} c_t\right)^2 \leq T\sum_{t=1}^{T} c_t^2, \tag{6.181}$$

which applied to eq. (6.179) yields

$$\| \, |\psi_\omega(T)\rangle - |\varphi(T)\rangle \, \|^2 \leq 4T\left(\sum_{t=1}^{T} |\langle\omega|\varphi(t)\rangle|^2\right). \tag{6.182}$$

Summing over ω we find

$$\sum_{\omega} \| \, |\psi_\omega(T)\rangle - |\varphi(T)\rangle \, \|^2 \leq 4T\sum_{t=1}^{T}\langle\varphi(t)|\varphi(t)\rangle \ = \ 4T^2. \tag{6.183}$$

Invoking eq. (6.174) we conclude that

$$4T^2 \geq 2N - 2\sqrt{N}, \tag{6.184}$$

if the states $|\psi_\omega(T)\rangle$ are mutually orthogonal. We have, therefore, found that any quantum algorithm that can distinguish all the possible values of the marked state must query the oracle T times where

$$T \geq \sqrt{\frac{N}{2}}, \tag{6.185}$$

(ignoring the small correction as $N \to \infty$). This lower bound on the number of queries applies if we are to identify the marked state with a success probability that approaches one for asymptotically large N. A simple extension of argument shows that

$$T \geq \sqrt{\frac{N}{2}}\left(1 - \sqrt{\varepsilon}\right)^{1/2}, \tag{6.186}$$

if instead we settle for success probabilty at least $1 - \varepsilon$ for each possible choice of the marked string ω.

Grover's algorithm finds ω in $\frac{\pi}{4}\sqrt{N}$ queries, which exceeds our lower bound by only about 11%. In fact, by a more careful argument, we can derive a tighter lower bound on the number of queries that asymptotically matches what Grover's algorithm achieves. Furthermore, we can show that Grover's circuit attains the best success probability among all circuits with T oracle calls in the case where $T < \frac{\pi}{4}\sqrt{N}$.

One feels a twinge of disappointment (as well as a surge of admiration for Grover) at the realization that the quantum search algorithm cannot be improved in the black box setting. What are the implications for quantum complexity?

For many optimization problems in the NP class, there is no better method known than exhaustive search of all the possible solutions. By exploiting quantum parallelism, we can achieve a quadratic speedup of exhaustive search. Now we have learned that the quadratic speedup is the best possible if we rely on the power of sheer quantum parallelism — that is, if we don't design our quantum algorithm to exploit the specific structure of the problem we wish to solve.

The optimality of the Grover algorithm might be construed as evidence that BQP $\not\supseteq$ NP. At least, if it turns out that NP $\subseteq$ BQP and P $\neq$ NP, then the NP problems must share a deeply hidden structure (for which there is currently no evidence) that is well-matched to the peculiar capabilities of quantum circuits.

Even the quadratic speedup may prove useful for a variety of NP-complete optimization problems. But a quadratic speedup, unlike an exponential one, does not really move the frontier between solvability and intractability. Quantum computers may someday outperform classical computers in performing exhaustive search, but only if the clock speed of quantum devices does not lag too far behind that of their classical counterparts.

6.8 Using quantum computers to simulate quantum physics

Sadly, we now know some things that quantum computers cannot do. But happily, there are also some important things that they *can* do.

- They can solve the hidden subgroup problem for finitely generated abelian groups, with an exponential speedup (in the oracle model) relative to classical algorithms.
- The can speed up, quadratically, exhaustive search for solutions to combinatorial problems.

What *else* can they do?

6.8.1 Simulating time evolution of local Hamiltonians

An important application for quantum computers is simulating the time evolution of quantum systems. The simulation is efficient if the Hamiltonian H is *local.*

For a system of n qubits, we say that H is *k-local* if

$$H = \sum_a H_a, \tag{6.187}$$

where each term H_a acts non-trivially on at most k qubits — i.e. $H_a = \tilde{H}_a \otimes I^{n-k}$, and $\tilde{H}_a$ acts on some set of at most k qubits. (Of course, we may use a similar definition for a system of d-dimensional subsystems for constant $d > 2$, rather than qubits.) We say that H is local if it is k-local for some constant k.

There is a stronger notion of locality we sometimes use, which can be called *geometrical locality* or *spatial locality.* A k-local Hamiltonian is geometrically local in D dimensions if the qubits can be arranged in (flat) D-dimensional space with a bounded number of qubits per unit volume, and the k qubits upon which H_a acts non-trivially are all contained in a ball of constant radius. In this sense there are no *long-range* interactions among the qubits. H is geometrically local if it is geometrically k-local in D dimensions for some constant D and k.

If we write $H = \sum_a H_a$ where there is a unique H_a for each set of k qubits, then the expansion of a k-local H contains at most $\binom{n}{k} = O(n^k)$ terms, and the expansion of a geometrically local H contains $O(n)$ terms (each of the n qubits is contained in a constant number of interacting sets). Let us also assume that each H_a is bounded

$$||H_a||_\infty \leq h \quad \text{for all } a, \text{ where } h \text{ is a constant.} \tag{6.188}$$

Physicists are interested in geometrically local Hamiltonians because they seem to provide an accurate description of Nature. Therefore, it is noteworthy that quantum circuits can simulate quantum evolution governed by a local Hamiltonian efficiently: evolution of n qubits for time t can be simulated to constant accuracy using a circuit whose size is polynomial in n and t.

We can formulate the problem this way: suppose we are given an initial quantum state $|\psi(0)\rangle$, or a classical description of a quantum circuit that prepares the state. Our goal is to construct

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle \tag{6.189}$$

where $U(t)$ satisfies $\frac{d}{dt}U(t) = -iH(t)U(t)$ and the boundary condition $U(0) = I$. (Thus $U(t) = e^{-iHt}$ in the case where H is time independent). We will settle for computing $|\psi(t)\rangle$ to accuracy δ, i.e. constructing $\tilde{\psi}(t)\rangle$ where

$$|||\tilde{\psi}(t)\rangle - |\psi(t)\rangle|| < \delta. \tag{6.190}$$

Depending on the situation, we might be satisfied if δ is a sufficiently small constant, or we might impose the stricter requirement that the error is smaller than some specified power of the size n of the system. To relate this simulation task to a task that can be described classically, suppose the goal is to sample from the probability distribution

$$\langle\psi(t)|\Pi_a|\psi(t)\rangle \tag{6.191}$$

where Π_a projects onto an eigenstate with eigenvalue a of an observable A that can be measured efficiently by a quantum computer. Classicaly this task is believed to be hard at least in some cases, because the unitary matrix $U(t)$ is exponentially large ($2^n \times 2^n$). But quantumly we can do the simulation efficiently if H is a local Hamiltonian.

To simulate continuous time evolution on a classical or quantum computer, we choose a small step size Δ, and approximate evolution for time t by a sequence of t/Δ steps. (If H is actually time dependent, assume Δ is small enough that the change of H during a time interval of width Δ can be neglected.) We wish to attain accuracy

$$||\tilde{U}(t) - U(t)||_\infty < \delta, \tag{6.192}$$

where $\tilde{U}$ is the simulated unitary and U is the ideal unitary. Hence the error per time step should be less than $\delta\Delta/t$.

Suppose $H = \sum_a H_a$ is a sum of M k-local terms, and let's consider the geometrically local case, where $M = O(n)$. We will show below that a single time step can be simulated by a product of M local "gates" (unitary transformations that act on a constant number of qubits) where each such "gate" has an error $O(\Delta^2 h^2)$. Therefore the simulation of evolution for time t uses all together Mt/Δ gates where we require

$$\frac{Mt}{\Delta}\Delta^2 h^2 \approx \delta \quad \Longrightarrow \quad \Delta = O\left(\frac{\delta}{h^2 Mt}\right). \tag{6.193}$$

Therefore the total number of gates is

$$L = O\left(\frac{h^2(Mt)^2}{\delta}\right). \tag{6.194}$$

Furthermore each "gate" can be simulated to accuracy $O(\Delta^2 h^2)$ with a universal gate set using polylog $\left(\frac{1}{\Delta^2 h^2}\right)$ = polylog $\left(\frac{h^2(Mt)^2}{\delta^2}\right)$ gates, according to the Solovay-Kitaev theorem. We conclude that the simulation can be done with a quantum circuit of size

$$L = O\left(\frac{h^2(Mt)^2}{\delta}\text{polylog}\left(\frac{h^2(Mt)^2}{\delta^2}\right)\right). \tag{6.195}$$

In the case where H is geometrically local, $M = O(n) = O(V)$, where V is the spatial volume of the system. Since h is a constant, we find that the cost of simulating time evolution with fixed accuracy scales like

$$L = O(\Omega^2 \text{ polylog } \Omega), \tag{6.196}$$

where $\Omega = Vt$ is the simulated volume of spacetime.

Now we need to explain how to simulate a single time step. We'll use the idea that $\exp\left(\sum_a A_a\right)$ can be approximated by $\prod_a \exp\left(A_a\right)$ if $||A|| \ll 1$. To check the accuracy we expand the exponentials:

$$\begin{aligned}
&\exp\left(\sum_a A_a\right) - \prod_a \exp\left(A_a\right) \qquad (6.197)\\
&= \left(1 + \sum_a A_a + \frac{1}{2}\sum_{a,b} A_a A_b + \dots\right) - \prod_a \left(1 + A_a + \frac{1}{2}A_a^2 + \dots\right)\\
&= \left(1 + \sum_a A_a + \frac{1}{2}\sum_{a,b} A_a A_b + \dots\right) - \left(1 + \sum_a A_a + \sum_a \frac{1}{2}A_a^2 + \sum_{a<b} A_a A_b + \dots\right)\\
&= \frac{1}{2}\left(\sum_{a<b} A_a A_b + \sum_{a<b} A_b A_a\right) - \sum_{a<b} A_a A_b + \dots\\
&= -\frac{1}{2}\sum_{a<b}[A_a, A_b] + \dots
\end{aligned}$$

(where $+\dots$ denotes terms higher order in A_a). Writing $H = \sum_a H_a$, then, we find that

$$e^{-iH\Delta} - \prod_a e^{-iH_a\Delta} = \frac{1}{2}\Delta^2 \sum_{a<b}[H_a, H_b] + \text{h.o.} \tag{6.198}$$

Now, how many non-vanishing commutators $\{[H_a, H_b]\}$ can occur in this sum? Let's suppose the Hamiltonian is geometrically local, in which case there are $O(n)$ terms in H, and each term fails to commute with a constant number of terms. So, there are $O(n) = O(M)$ non-vanishing commutators. We conclude that (in the geometrically local case)

$$\left\|e^{-iH\Delta} - \prod_a e^{-iH_a\Delta}\right\| = O(M\Delta^2 h^2). \tag{6.199}$$

Since $\Pi_a e^{-iH_a\Delta}$ is a product of M "gates," we have verified that the accuracy per gate is $O(\Delta^2 h^2)$ (Note that terms arising from the higher-order terms in the expansion of the exponential are of order $M\Delta^3 h^3$, and therefore systematically supressed by another factor of $\Delta h = O(\delta/hMt) = O((\delta/L)^{1/2})$.)

So far we have shown that, for a geometrically local H that is a sum of bounded terms, evolution in a spacetime volume Ω can be achieved with a quantum circuit of size

$$L = O(\Omega^2 \text{ polylog } \Omega), \tag{6.200}$$

The resources needed for the simulation scale like the *square* of the simulated volume (up to a log factor). Can this be improved?

An improved approximation to $\exp\left(\sum_a A_a\right)$ is the subject of an exercise. Instead of $\prod_a e^{A_a}$ we use $\prod_{a\rightarrow} e^{\frac{1}{2}A_a}\prod_{a\leftarrow} e^{\frac{1}{2}A_a}$, where $\prod_{a\rightarrow}$ denotes the product in ascending order, and $\prod_{a\leftarrow}$ denotes the product in descending order. The exercise shows that, for geometrically local H,

$$||e^{-iH\Delta} - \prod_{a\rightarrow} e^{\frac{1}{2}H_a\Delta}\prod_{a\leftarrow} e^{\frac{1}{2}H_a\Delta}|| = O(M\Delta^3h^3), \tag{6.201}$$

i.e. the error per gate is $O(h^3\Delta^3)$ instead of $O(h^2\Delta^2)$. For an accurate simulation, we choose

$$\frac{Mt}{\Delta}(\Delta^3h^3) \approx \delta \implies \Delta \approx \left(\frac{\delta}{h^3Mt}\right)^{1/2}; \tag{6.202}$$

the number of gates is

$$\frac{Mt}{\Delta} \approx \frac{h^{3/2}(Mt)^{3/2}}{\delta^{1/2}}, \tag{6.203}$$

and the Solovay-Kitaev blowup factor is polylog $\left(\frac{1}{\Delta^3h^3}\right)$ = polylog $\left(\frac{h^{3/2}(Mt)^{3/2}}{\delta^{3/2}}\right)$ = polylog $\left(\frac{hMt}{\delta}\right)$. We conclude that spacetime volume Ω can be simulated with a circuit of size

$$L = O(\Omega^{3/2} \text{ polylog } \Omega). \tag{6.204}$$

The improved approximation in each time step increases the circuit size by only a factor of 2.

The accuracy of this *Trotter-Suzuki approximation* to $e^{-iH\Delta}$ can be improved further, so that

$$||e^{-iH\Delta} - \text{approximation}|| = O\left(c_pM(h\Delta)^{p+1}\right) \tag{6.205}$$

where p is any power, the constant c_p depends on p, and the improved approximation increases the number of gates by a factor that depends on p. With this approximation, we choose

$$\frac{Mt}{\Delta}(h\Delta)^{p+1} \approx \delta \implies \Delta \approx \frac{\delta^{1/p}}{h^{(p+1)/p}(Mt)^{1/p}}, \tag{6.206}$$

so that the number of gates is

$$\frac{Mt}{\Delta} \approx \frac{h^{(p+1)/p}(Mt)^{(p+1)/p}}{\delta^{1/p}}. \tag{6.207}$$

Including the Solovay-Kitaev log factor, we can do the simulation with a circuit of size

$$L = O(\Omega^{1+\epsilon} \text{ polylog } \Omega), \tag{6.208}$$

where $\epsilon = 1/p$. In fact, though, the factor c_p grows exponentially with $p = 1/\epsilon$; this happens because each time we increase the value of p by 1, we may the price of increasing the circuit size by a constant factor. But anyway, as we systematically improve the approximation, the circuit size comes closer and closer to scaling linearly with the simulated volume, though it never quite makes it. Somehow, Nature manages to simulate herself with "resources" scaling like Ω (and without any error!), but this approximation method using a universal quantum computer does not do quite as well. There are other simulation methods that more nearly match the scaling Nature achieves.

Whether the quantum circuit model can simulate Nature efficiently is an important issue, because it addresses whether this model is the right one for studying what problems can be solved with reasonable resources by computers that are in principle physically realizable. We believe that the classical Turing machine model is not the right model, because it seems to be incapable of efficiently simulating general quantum systems, even ones for which the Hamiltonian is geometrically local. The quantum circuit model is presumably stronger, but is it really strong enough?

Particle physicists model fundamental physics using quantum field theory (QFT). Some predictions of QFT can be computed classically, and the success of such predictions provides the evidence that QFT is a good model. But we don't know how to efficiently simulate on a classical computer the real-time evolution of, for example, nuclear matter governed by quantum chromodynamics. Can we do such simulations efficiently on a quantum computer?

The question is subtle because the number of qubits per unit volume is formally infinite in QFT (there are degrees of freedom at arbitrarily small distances) and although the Hamiltonian is local, the terms in the Hamiltonian do not necessarily have bounded norm. On the other hand, in a physical process of interest, we can usually assume that the energy density per unit volume is bounded above. In that case, we expect that the very-short-distance degrees of freedom are not so relevant, so that we can attain reasonable accuracy by approximating continuous space with a finite density of lattice points per unit volume, and that furthermore the terms in the local Hamiltonian can be well approximated by operators with bounded norm. A reasonable expectation is that the complexity of a simulation of the dynamics scales polynomially in $\Omega\rho_{\max} = (Vt\rho_{\max}) = E_{\max}t$ where $\rho_{\max}$ is the maximal energy per unit volume, and $E_{\max}$ is an upper bound on the total energy. But there is no rigorous theorem establishing such scaling. Though physicists have a pretty good grasp of the properties of QFT (as indicated by the agreement between theory and experimental data), rigorous mathematical results pertaining to QFT are still quite technical and incomplete, particulary in three spatial dimensions (where we live).

Our understanding is even less satisfactory for physical processes in which gravitational effects are important. Can the quantum circuit model simulate quantum gravity efficiently? If so, quantum computers may turn out to be a powerful tool for deepening our understanding of quantum gravity. If not, then we still have not found the computational model that fully captures the computational power that is potentially realizable in Nature.

6.8.2 Estimating energy eigenvalues and preparing energy eigenstates

We have argued that a quantum computer can efficiently simulate the time evolution of a quantum system with a local Hamiltonian; i.e., it can solve the time-dependent Schroedinger equation. Another thing that physicists and chemists want to do is solve the time-independent Schroedinger equation; i.e., compute the energy eigenvalues of a Hamiltonian. For example, chemists say that estimating the ground state energy of a molecule to "chemical accuracy" (about one part in a million) is valuable for predicting the properties of chemical reactions. In general, finding the energy eigenvalues of a local Hamiltonian seems to be a hard problem classically because we need to diagonalize a $2^n \times 2^n$ matrix for a system of n qubits. In some cases this may be easy, for example if the matrix is very sparse and has a simple structure, but in some physically relevant cases efficient classical algorithms are not known.

Sometimes, if we express H in a "natural" basis we find that all the off-diagonal terms in the matrix are non-positive, i.e.

$$H = cI - h \tag{6.209}$$

where I is the identity and h has only non-negative entries. In that case, the ground state $|\psi_0\rangle$ of H (the eigenvector with the lowest eigenvalue) can be expressed as $|\psi_0\rangle = \sum_i c_i|i\rangle$ where $|i\rangle$ is a basis element and all c_i can be chosen non-negative. That is because $|\psi_0\rangle$ maximizes

$$\langle\psi_0|h|\psi_0\rangle = \sum c_i^* c_j h_{ij} \tag{6.210}$$

and so it is optimal to choose $c_i^* c_j \geq 0$ for all i and j. When all the c_i are nonnegative, there are *quantum Monte Carlo* sampling algorithms that can find the $\{c_i\}$ efficiently and accurately in practice. But if the off-diagonal terms in H have both $+$ and $-$ signs, then sampling algorithms might not work well because there can be delicate cancellations between positive and negative terms contributing to $\langle\psi_0|h|\psi_0\rangle$. Computational physicists call this the *sign problem.*

But on a quantum computer, we can estimate eigenvalues of a unitary matrix U using the *phase estimation* algorithm. To obtain m bits of accuracy, we prepare an m-qubit register in a uniform superposition state, and execute this circuit:

-figure-

The measured value of k provides an estimate of the eigenvalue to m bits of accuracy. To perform U^t conditioned on t we simulate e^{-iHt} for $t \in T \times \{1, 2, 4, 8, ..2^{m-1}\}$. This suffices to find the fractional part of $\frac{ET}{2\pi}$ to m-bit accuracy, where E is an eigenvalue of the Hamiltonian H. We choose the step size in the simulation of e^{-iHt} so that the accuracy is $\delta \approx 2^{-m}$ for $t = 2^m T$. If the Hamiltonian is geometrically local, we have seen that this approximation can be achieved with a circuit size (neglecting a log factor)

$$L = O\left(\frac{h^2(nt)^2}{\delta}\right) = O\left(h^2(nT)^2 \times \frac{2^{2m}}{2^{-m}}\right) = O\left((hT)^2 n^2 2^{3m}\right). \tag{6.211}$$

For a particular preparation of the input state $|\psi\rangle$, suppose we repeat the computation many times, and plot a histogram of the results:

-figure-

The *location* of each narrow peak estimates an energy eigenvalue E_a, modulo $2\pi/T$. The *height* of the peak estimates $|\langle E_a|\psi\rangle|^2$ – the overlap $|\psi\rangle$ with the corresponding energy eigenstate $|E_a\rangle$. To compute the energy eigenvalue to accuracy polynomial in n, we choose

$$\delta \approx 2^{-m} \approx 1/n^c \implies m = c\ \log_2 n. \tag{6.212}$$

The algorithm is efficient: the quantum circuit size is

$$O\left(2^{3m}h^2(nT)^2\right) = O(n^{3c}n^2). \tag{6.213}$$

However, if we want to estimate (say) the ground state energy E_0 to polynomial accuracy in quantum polynomial time, we must be able to prepare a state $|\psi\rangle$ whose overlap with the ground state $|E_0\rangle$ is no worse than polynomially small:

$$|\langle E_0|\psi\rangle|^2 > 1/\text{poly}(n). \tag{6.214}$$

If that is the case, we can get a good estimate of E_0 in only polynomially many trials. As a bonus, when we obtain the value E_0 for the measured eigenvalue E_0, then we have projected state $|\psi\rangle$ onto the ground state $|E_0\rangle$, and therefore we can compute further properties of $|E_0\rangle$, such as the distribution $\text{Prob}(a) = \langle E_0|\Pi_a|E_0\rangle$, where Π_a is projector onto eigenspace of an efficiently measurable observable.

The catch is that the overlap of a randomly chosen n-qubit state with $|E_0\rangle$ is exponentially small, so preparing $|\psi\rangle$ with a polynomially small overlap is not necessarily easy. One way we might attempt to construct a state with a significant overlap with the ground state is by appealing to the *quantum adiabatic theorem.* We prepare the ground state of a Hamiltonian H_{easy} whose ground state is easy to find classically. Then we simulate Schroedinger evolution governed by a time-dependent Hamiltonian $H(t)$ such that

$$H(0) = H_{\text{easy}}, \quad H(T) = H_{\text{hard}}, \quad 0 \le t \le T, \tag{6.215}$$

where H_{hard} is the Hamiltonian whose ground state we wish to construct. For example, we might choose

$$H(t) = (1 - t/T)H_{\text{easy}} + (t/T)H_{\text{hard}}. \tag{6.216}$$

The quantum adiabatic theorem says that if $|\psi(0)\rangle$ has a large overlap with the ground state of H_{easy}, then $|\psi(T)$ has a large overlap with the ground state of H_{hard}, *if* T is long enough (i.e. the Hamiltonian $H(t)$ changes slowly enough). But how slow is slow enough? Let $E_0(t)$ denote the energy of the ground state of $H(t)$ and let $E_1(t)$ denote the energy of the first excited state of $H(t)$. Define $\Delta = \min_{t\in[0,T]}(E_1(t) - E_0(t))$; we say Δ is the minimum *energy gap* between the ground and first excited state. Then the adiabatic theorem says that $T > A/\Delta^c$ is slow enough, where A and c are constants independent of the number of qubits n. Therefore, we have a complete polynomial algorithm for computing the ground state energy of a local Hamiltonian to polynomial accuracy in polynomial time provided that

$$\Delta = \frac{1}{\text{poly}(n)}. \tag{6.217}$$

But if Δ becomes exponentially small in n during the evolution in which $H(t)$ interpolates between H_{easy} and H_{hard}, then this algorithm may require exponential time.

Unfortunately, we'll see that it follows from weak complexity-theoretic assumptions that there are local Hamiltonians for which computing ground state energy is hard even for a quantum computer. (For example, otherwise we would be able to solve efficiently any problem in NP using a quantum computer, which seems unlikely.) So the minimum energy gap Δ must be smaller than polynomially small in such cases.

On the other hand, it is reasonable to be hopeful that computing ground state energy for quantum systems will be an important application of quantum computing. For example, as previously note, chemists say it is valuable to compute the energy of the electronic ground state of a molecule with atomic nuclei at fixed positions to accuracy $\approx 10^{-6}$ (*chemical accuracy*). They also claim that it is an adequate approximation to express

$$H = \sum_{a=1}^{M} H_a \tag{6.218}$$

and each H_a acts on ≤ 4 basis functions. Hence this Hamiltonian is local. Furthermore, chemists assert (without proof) that it is possible to evolve adiabatically from the *Hartree-Fock* Hamiltonian (which they can solve classically) to the *full configuration interaction* (FCI) Hamiltonian (which they want to solve, but don't know how to solve classically in general) while the gap satisfies $\Delta \geq$ constant. If that is true, quantum computers are likely to be a powerful tool for studying molecular chemistry.

6.9 Classical simulation of slightly entangling quantum computations

How hard is it to simulate a quantum computer using a classical computer? Clearly, we would like to understand more deeply the classical and quantum complexity of solving the time-dependent and time-independent Schroedinger equations. In particular, we wish to identify cases for which the problem seems to be hard classically and easy quantumly, for these are cases where quantum computers will find a useful niche.

More broadly, why do we believe that quantum computers are more powerful than classical ones, and what is the source of the quantum computer's power? Roughly speaking, it seems to be hard to simulate a quantum system with a classical computer because the Hilbert space of the quantum computer is exponentially large in the number of qubits n, and that exponentially large Hilbert space is needed to accommodate and describe the quantum correlations among the qubits in a many-body quantum system. From this perspective, it seems legitimate to claim that quantum entanglement is the source of the quantum computer's power. On the other hand, that claim may be too simplistic, because: 1) Some quantum computations that generate highly entangled quantum states are easy to simulate classically. We'll see an example next term when we discuss (in connection with quantum error correction) quantum computation using the Clifford group. 2) For *mixed* states, simulating a quantum computation classically might be hard even if the state remains separable (that is, unentangled) at all times during the computation — even if the correlations among the parts of the quantum computer are "classical" they could still be hard to simulate. You examined an example in a homework problem: estimating the trace of an exponentially large unitary matrix using just "one clean qubit."

So let's ask the question this way: for quantum computation with pure states, if the qubits in the computer never become highly entangled during the course of the computation, can we simulate the quantum computer efficiently with a classical computer? As we'll see, the answer is yes.

Imagine that n qudits (d-dimensional systems) are arranged in a line, and consider cutting the system into two parts anywhere along the line: there are m qudits to the left of the cut and $n-m$ qubits to the right of the cut. Suppose that for any way of choosing where we cut the chain, the entanglement between the two parts is bounded above by a constant (independent of the system size n). We could quantify the entanglement in various ways, and the conclusion would be the same, but to keep the discussion simple let us use the Schmidt number. Recall that the Schmidt number is the number of terms in the Schmidt expansion of a bipartite pure state — equivalently it is the rank of the density operator for either part.

While in principle the Schmidt number could be as large as $d^{n/2}$ when the system is cut into two subsystems of equal size, let us assume that

$$\text{Schmidt number} \leq D = \text{constant}. \tag{6.219}$$

We claim that under this assumption:

1. There is a succinct classical description of the pure state of n qudits.
2. A computation in which the n qudits have bounded Schmidt number at all times can be efficiently simulated on a classical computer.

First, let's construct the succinct description. The n-qudit state $|\psi\rangle$ can be expanded in the standard basis as

$$|\psi\rangle = \sum c^{i_1 i_2 .. i_n} |i_1 i_2 .. i_n\rangle \tag{6.220}$$

in terms of the d^n complex numbers $\{c^{i_1 i_2 .. i_n}\}$. This is not succinct in general, but if the Schmidt rank is bounded then each $c^{i_1 i_2 .. i_n}$ can be expressed as a contraction of tensors, where each tensor has 3 indices, and each index has a constant number of values:

-figure-

Here each $(P_k)^{i_k}_{\alpha_{k-1}\alpha_k}$ has dD^2 entries (except for the sites at the ends of the chain, which have dD entries). Therefore, the state is described by ndD^2 complex parameters, a number that is linear rather than exponential in n. (In the case of a product state, this becomes nd parameters.)

To obtain this compressed description of the n-site state $|\psi\rangle$ we perform the Schmidt decomposition repeatedly ($n-1$ times in succession). In the first step we divide the system between qudits 1 and 2:

$$|\psi\rangle = \sum_{\alpha_1} \lambda_{1\alpha_1} |\psi_{1\alpha_1}\rangle |\phi_{2\alpha_1}\rangle. \tag{6.221}$$

Here the $\{\lambda_{1\alpha_1}\}$ are Schmidt coefficients, the $\{|\psi_{1\alpha_1}\rangle\}$ are elements of an orthonormal basis for the first qudit, and $\{|\phi_{2\alpha_1}\rangle\}$ are elements of an orthonormal basis for qudits $2, 3, ..n$. We can expand $|\psi_{1\alpha_1}\rangle$ in the standard basis, obtaining

$$|\psi\rangle = \sum_{\alpha_1} \sum_{i_1} \lambda_{1\alpha_1} \psi^{i_1}_{1\alpha_1} |i_1\rangle |\phi_{2\alpha_1}\rangle = \sum_{\alpha_1} \sum_{i_1} (P_1^{i_1})_{\alpha_1} |i_1\rangle |\phi_{2\alpha_1}\rangle, \tag{6.222}$$

where $(P_1^{i_1})_{\alpha_1} = \lambda_{1\alpha_1} \psi^{i_1}_{1\alpha_1}$ Now we Schmidt decompose $|\phi_{2\alpha_1}\rangle$ across the cut between qudits 2 and 3, finding

$$|\phi_{2\alpha_1}\rangle = \sum_{\alpha_2} \sum_{i_2} (P_2^{i_2})_{\alpha_1 \alpha_2} |i_2\rangle |\phi_{3\alpha_2}\rangle. \tag{6.223}$$

Note that there is no α_1 label on the states $\{|\phi_{3\alpha_2}\rangle\}$. That is because these are the states occuring in the Schmidt decomposition across the 2-3 cut that describe the right side of the cut. These states have nothing to do with α_1, which labels the states in the Schmidt decomposition across the 1-2 cut.

Repeating the Schmidt decomposition $n-1$ times we find

$$|\psi\rangle = \sum_{\alpha_1..\alpha_{n-1}} \sum_{i_1,..i_n} (P_1^{i_1})_{\alpha_1}(P_2^{i_2})_{\alpha_1\alpha_2}(P_3^{i_3})_{\alpha_2\alpha_3}..(P_n^{i_n})_{\alpha_{n-1}}|i_1 i_2..i_n\rangle. \tag{6.224}$$

This is the decomposition in terms of contracted tensors that we sought. It is called the *matrix product state* (MPS) decomposition of $|\psi\rangle$. It exists for any $|\psi\rangle$, but for generic $|\psi\rangle$, the matrices in the middle of the chain of n qudits become exponentially large in n. Under the assumption of bounded Schmidt rank, however, $(P_1)^{i_1}$ and $(P_n)^{i_n}$ are D-component vectors and $(P_k)^{i_k}$ for $k = 2, 3, ..n-1$ are $D \times D$ matrices.

Therefore, we have expressed $|\psi\rangle = \sum_{i_1,..i_n} P_1^{i_1} P_2^{i_2}..P_n^{i_n}|i_1 i_2..i_n\rangle$ where the coefficient is a product of matrices. If we don't want the endpoints to be special we can replace $P_1^{i_1}$ and $P_n^{i_n}$ by $D \times D$ matrices contracted with one another; then

$$|\psi\rangle = \sum_{i_1,..i_n} \mathrm{tr}\left(P_1^{i_1} P_2^{i_2}..P_n^{i_n}\right)|i_1 i_2..i_n\rangle \tag{6.225}$$

This description looks like the MPS locally, but with the ends of the chain now glued together.

Suppose that the qudits are arranged in a one-dimensional chain, and that a circuit of geometrically local quantum gates acts on the state, such that each gate acts on a pair of neighboring qudits. We want to see that the MPS description can be efficiently updated as this circuit is executed. If a unitary transformation U acts on a pair of neighboring qudits on the chain as in

-figure-

this unitary affects only two neighboring matrices:

$$\sum_\beta (P^i)_{\alpha\beta}(Q^j)_{\beta\gamma} \mapsto \sum_\beta \sum_{i'j'} U^{ij}_{i'j'}(P^{i'})_{\alpha\beta}(Q^{j'})_{\beta\gamma} \equiv M^{ij}_{\alpha\gamma} \tag{6.226}$$

Now we need to update the Schmidt decomposition across the P-Q cut:

-figure-

We regard $M^{ij}_{\alpha\gamma}$ as a $dD \times dD$ matrix, and we perform its *singular value decomposition* (SVD). Recall that for any matrix M, there are unitary matrices V_L and V_R such that $M = V_L^\dagger \Delta V_R$, where Δ is diagonal. This Schmidt decomposes the state

$$\sum_{ac} M_{ac}|ac\rangle = \sum_b \Delta_b |\psi_{Lb}\rangle \otimes |\psi_{Rb}\rangle \tag{6.227}$$

where $|\psi_{Lb}\rangle = \sum_a |a\rangle V^\dagger_{Lab}$ and $|\psi_{Rb}\rangle = \sum_c |c\rangle V^\dagger_{Rbc}$. Here $|ac\rangle$ is shorthand for $|\alpha i, \gamma j\rangle$, where $|\alpha\rangle$ labels the states in the Schmidt decomposition across the cut just to the left of the two neighboring sites where U acts, and γ labels states in the Schmidt decomposition across the cut just to the right.

The SVD of an $N \times N$ matrix can be computed in time $O(N^3)$ on a classical computer. In the case of $M^{ij}_{\alpha\gamma}$, for which $N = dD$, this is time $O(d^3D^3)$. In general, the middle index b would be summed over $N(= dD)$ values. But if the Schmidt rank stays bounded by D, the matrix Δ has at most D nonzero eigenvalues. We may identify $(P'^i)_{\alpha\beta} = (V_L^\dagger\sqrt{\Delta})^i_{\alpha\beta}$ and $(Q'^j)_{\beta\gamma} = (\sqrt{\Delta}V_R)^j_{\beta\gamma}$, where β is summed over D values, completing the update of the MPS. The point is that when U acts on neighboring sites labeled a, $a+1$, we need not update the Schmidt states to the left of site a or to the right of site $a+1$.

Acting on a product state, any 2-qudit state acting across a given cut can produce a state with Schmidt number at most d^2 across that cut.

-figure-

This can be achieved by a SWAP gate acting on two maximally entangled pairs, one on each side of the cut. Likewise, acting on a state with Schmidt number D, the two-qudit gate can increase the Schmidt number to at most d^2D . This means that, starting with a product state, a circuit in which no more than G gates act across any particular cut creates a state with Schmidt number at most $D = (d^2)^G$ across that cut. (We can simulate a 2-qudit gate U acting across the cut by swapping the qubits until they are at adjacent sites on either side of the cut, applying U, and then swapping back. Only U, not the swaps, increases the Schmidt number, and it increases it by at most the factor d^2.) The quantum state admits an efficient MPS classical description provided that the Schmidt number D is bounded above by poly(n) across every cut, and furthermore this description can be efficiently updated each time a quantum gate is applied. It follows that a quantum computation acting on n qubits can be simulated efficiently by a classical computer if the initial state of the quantum computer is a product state, and the number of gates that act across each cut (for some ordering of the qubits) is $O(\log n)$, since in that case we have $D \leq (d^2)^{O(\log n)} = n^{O(\log d^2)}$.

We should also note that, for an MPS, local measurements of the qudits in the chain are easy to simulate. First let's impose the proper normalization condition on the state $|\psi\rangle$:

$$
\begin{aligned}
|\psi\rangle &= \sum_{i_1,..i_n} \mathrm{tr}\left(P_1^{i_1}P_2^{i_2}..P_n^{i_n}\right)|i_1i_2..i_n\rangle \qquad (6.228)\\
\implies \langle\psi|\psi\rangle &= \sum_{i_1,..i_n}\left|\mathrm{tr}\left(P_1^{i_1}P_2^{i_2}..P_n^{i_n}\right)\right|^2,
\end{aligned}
$$

which is conveniently written as

$$
\sum_{i_1,..i_n} \mathrm{tr}\left(\left(P_1^{i_1}\otimes P_1^{i_1*}\right)\left(P_2^{i_2}\otimes P_2^{i_2*}\right)..\left(P_n^{i_n}\otimes P_n^{i_n*}\right)\right) = \mathrm{tr}(E_1E_2,,E_n), \qquad (6.229)
$$

where $E_k = \sum_{i_k} P_k^{i_k}\otimes P_k^{i_k*}$ is a $D^2 \times D^2$ matrix. The expectation value of a product observable can be expressed as:

$$\begin{aligned}
&\langle\psi|\mathcal{O}_1 \otimes \mathcal{O}_2 \otimes .. \otimes \mathcal{O}_n|\psi\rangle = \\
&= \sum_{j_1..j_n}\sum_{i_1..i_n} \mathrm{tr}\left(P_1^{j_1 *}..P_n^{j_n *}\right)\langle j_1|\mathcal{O}_1|i_1\rangle..\langle j_n|\mathcal{O}_n|i_n\rangle \times \mathrm{tr}\left(P_1^{i_1}..P_n^{i_n}\right) \\
&= \mathrm{tr}\left(E_1(\mathcal{O}_1)E_2(\mathcal{O}_2)..E_n(\mathcal{O}_n)\right)
\end{aligned} \tag{6.230}$$

where $E_k(\mathcal{O}_k) = \sum_{i_k j_k}\left(P_k^{i_k} \otimes P_k^{j_k *}\right)\langle j_k|\mathcal{O}_k|i_k\rangle$ is also a $D^2 \times D^2$ matrix. We can evaluate the trace of a product of n constant-size matrices in time $O(n)$. This completes the proof that slightly entangled quantum computations can be classically simulated in polynomial time.

Finally, let's return to the problem of finding the energy of the ground state of a local Hamiltonian. In many one-dimensional systems studied in physics, the ground state and low-lying excited states can be well approximated by an MPS with a reasonable matrix size. That is, there is a good approximation to e.g. the ground state with maximal Schmidt number across any cut

$$D = \mathrm{poly}(n). \tag{6.231}$$

However there are local Hamiltonians (even translation-invariant ones) for which finding the ground state energy seems to be hard not only for classical computers but also for quantum computers! In these cases, either an accurate MPS representation requires matrices super-polynomial in size, or, if there is a good MPS approximation with $D = \mathrm{poly}(n)$, it must be a hard computational task to find that approximation.

6.10 QMA-completeness of the local Hamiltonian problem

As we have discussed, we expect that quantum computers can compute the ground state energy of local Hamiltonians in cases where the problem is hard classically; this may be an important application for quantum computers. On the other hand, we believe that in some cases computing the ground state energy of a local Hamiltonian is still hard even for quantum computers. Let us try to understand more deeply the reason for this belief.

Physicists are often interested in translation-invariant geometrically local Hamiltonians, in which all qubits interact in the same way with their neighbors (except for the qubits at the boundary of the sample), because such Hamiltonians provide good models of some real materials. But Hamiltonians that are not translationally invariant are also useful in physics (for example, when modeling a material with "disorder" due to dirt and other imperfections in the sample). If the Hamiltonian is not translation invariant, then we can formulate an instance of an n-qubit local Hamiltonian problem by specifying how the Hamiltonian varies from site to site in the system. Physicists sometimes refer to such (not translationally-invariant) systems as *spin glasses*. Even in the classical case, where the variable at each site is a bit rather than a qubit, finding the ground state energy of a spin glass to constant accuracy can be an NP-hard problem. Therefore, we don't expect classical or quantum computers to be able to solve the problem in general (unless NP is contained in BQP, which seems unlikely).

Let's first understand why the classical spin-glass problem can be NP-hard. Then we'll discuss the hardness of the quantum problem. We'll see that in the quantum case finding the ground state energy of a local Hamiltonian is QMA-hard. (Recall that QMA is the quantum analogue of NP: the class of problems such that the solution can be verified efficiently with a quantum computer if a suitable "quantum witness" is provided.)

For the classical case, we'll recall the notion of a"reduction" of one computational problem to another (B reduces to A if a machine that solves A can be used to solve B), and then we'll consider this sequence of reductions:

1. Any problem in NP reduces to CIRCUIT-SAT (already discussed previously); i.e., CIRCUIT-SAT is NP-complete.
2. CIRCUIT-SAT reduces to 3-SAT (3-SAT is NP-complete).
3. 3-SAT can be formulated as the problem of finding the ground state energy of a classical 3-local Hamiltonian to constant accuracy.
4. MAX 2-SAT is also NP-hard and can be formulated as the problem of finding the ground state energy of a classical 2-local Hamiltonian to constant accuracy.
5. The classical 2-local Hamiltonian problem is still hard in the case where the Hamiltonian is geometrically local, in three or even in two dimensions (cases of interest for describing real spin glasses).

(5) implies that a spin glass will not be able to relax to its ground state efficiently in any realistic physical process (which is part of what physicists mean by the word "glass").

6.10.1 3-SAT is NP-complete

Language: Recall (as discussed earlier) that if f is a uniform family of Boolean functions with variable input size, $f : \{0,1\}^* \mapsto \{0,1\}$, then the set of input strings accepted by f is called a *language*:

$$L = \{x \in \{0,1\}^* : f(x) = 1\} \tag{6.232}$$

NP: We say that a language is in NP if there is a polynomial-size uniform classical circuit family (the *verifier* $V(x,y)$) such that:

- If $x \in L$, then there exists a *witness* y such that $V(x,y) = 1$ (completeness).
- If $x \notin L$, then, for all y, $V(x,y) = 0$ (soundness).

Reduction: We say that B reduces to A if there is a polynomial-size uniform classical circuit family R mapping x to $R(x)$ such that B accepts x if and only if A accepts $R(x)$. This means we can hook up R to a machine that solves A to construct a machine that solves B.

An important problem in NP is CIRCUIT-SAT. The input to the problem is a Boolean circuit C (with an n-bit input and $G = \text{poly}(n)$ gates), and we are to evaluate the Boolean function $f(C)$, where $f(C) = 1$ if there is an input x such that $C(x) = 1$, $f(C) = 0$ otherwise. CIRCUIT-SAT is in NP because we can simulate the circuit C. Given as a witness the value of x that C accepts, we can verify efficiently that $C(x) = 1$. Furthermore, CIRCUIT-SAT is NP-complete (any problem in NP reduces to CIRCUIT-SAT), as we discussed previously. If $V(x,.)$ is the verifier for an NP problem with a fixed instance x, we may think of $V(x,.)$ as a circuit whose input is the witness y. Solving the CIRCUIT-SAT problem for this Boolean circuit tells us whether there exists a witness that the verifier accepts, and therefore solves the NP problem.

Now we come to a further reduction that we did not discuss previously: CIRCUIT-SAT reduces to 3-SAT, and therefore 3-SAT, too, is an NP-complete problem (the Cook-Levin theorem). For the SAT problem, the input is a "Boolean formula" with n variables, where each variable is a bit. The formula is a conjunction of clauses, and the formula is true if and only if every clause is true. In the k-SAT problem, each clause depends on at most k of the variables, where k is a constant. (In some formulations of k-SAT, each clause is required to be a disjunction of k *literals* (variables or their negations), but that is not an important requirement, since any formula, and in particular any k-bit formula, can be expressed in conjunctive normal form.). If f is a Boolean formula, the SAT function is:

$$\begin{aligned}&\mathrm{SAT}(f) = 1 \text{ if there exists } x \text{ such that } f(x) = 1, \\ &\mathrm{SAT}(f) = 0 \text{ otherwise.}\end{aligned} \tag{6.233}$$

Now we'll show that CIRCUIT-SAT reduces to 3-SAT. For a given circuit C (the input to CIRCUIT-SAT), how do we construct the corresponding Boolean formula $R(C)$ (the input to 3-SAT)?

Suppose that the gates in the circuit C are chosen from the universal set (AND, OR, NOT), or any other gate set such that each gate has at most two input bits and one output bit. We introduce a variable for the output of each gate, and we include in the formula $R(C)$ a clause corresponding to each gate.

Here the three-variable clause $C_g(x, y, z)$ is true iff z is a valid output of the gate g when the inputs are (x, y). The circuit may also have inputs that are constants rather than variables. Then, e.g. a gate with two input bits, one of which is a constant, becomes a two-variable clause, determined by the gate and the value of the constant. Or equivalently, we can regard input of a constant as a gate with a one-bit output; the corresponding one-bit clause is true if x has the right value. The circuit also has an answer bit, which becomes a 1-bit clause $C_g(x)$ which is true iff $x = 1$ (that is, if and only if the answer is YES).

The formula $R(C)$ has as variables the input x to the circuit C, and also additional variables corresponding to the outputs of all gates in the circuit C. $R(C)$ has been constructed so that an assignment that satisfies every clause in C corresponds to a valid history of the computation of the circuit C acting on input x, where the input is accepted. If there is an input x that is accepted by the circuit C, then there will be a satisfying assignment for the 3-SAT formula $R(C)$, and conversely if there is no input that C accepts, then there will be no satisfying assignment for $R(C)$. Thus we have obtained the desired reduction of CIRCUIT-SAT to 3-SAT.

The key idea we have exploited to reduce CIRCUIT-SAT to 3-SAT is that the witness for 3-SAT is a valid history of the whole computation C that accepts the input x. We can check the history efficiently because the circuit C has polynomial size and it is easy to check each of the poly(n) gates in the execution of the circuit. Later on, we will extend this idea — that a valid history of the computation is an efficiently checkable witness — to the quantum setting.

Notice that we may think of the clauses in the formula f as the terms in a 3-local classical Hamiltonian

$$H(x) = \sum_c H_c(x_{c1}, x_{c2}, x_{c3}); \tag{6.234}$$

here the sum is over the clauses in the formula, where

$$H_c(x_{c1}, x_{c2}, x_{c3}) = \begin{cases} 0 & \text{if clause } c \text{ is true for assignment } x_{c1}, x_{c2}, x_{c3}, \\ 1 & \text{otherwise} \end{cases} \tag{6.235}$$

Then $\min_x H(x) = 0$ if there is an assignment that satisfies every clause, while $\min_x H(x) \geq 1$ if there is no satisfying assignment (the number of violated clauses is ≥ 1 for any assignment). We conclude that finding the minimum value of a 3-local classical Hamiltonian is NP-hard: if we could do it we could solve 3-SAT. and hence any problem in NP. This conclusion implies, as asserted earlier, that finding the ground state energy of a 3-local classical Hamiltonian to constant accuracy must be hard in general for quantum computers, too, unless NP $\subseteq$ BQP.

6.10.2 Frustrated spin glasses

In fact, finding the ground state energy to constant accuracy is NP-hard even for a 2-local classical Hamiltonian

$$H(x) = \sum_c H_c(x_{c1}, x_{c2}). \tag{6.236}$$

Although 2-SAT (deciding whether a 2-SAT formula can be satisfied) is easy (there is a poly-time algorithm), MAX-2SAT is an NP-hard problem. MAX-2SAT is the problem of finding the *minimum* number of violated clauses for any assignment, which is equivalent to minimizing the Hamiltonian function H.

Furthermore, we can make the 2-local Hamiltonian geometrically local without losing hardness. An example is the *Ising spin-glass model* in three dimensions. Suppose the binary variables are *spins* sitting at the sites of a cubic lattice, taking values $Z_i \in \{\pm 1\}$ at site i in the lattice. Consider the Hamiltonian

$$H = -\sum_{\langle ij \rangle} J_{ij} Z_i Z_j \tag{6.237}$$

Here $\langle ij \rangle$ labels the edge in the lattice that connects two nearest-neighbor sites with labels i and j.

$J_{ij} \in \{\pm 1\}$ encodes the instance of the problem. If $J_{ij} = +1$, then we say the edge $\langle ij \rangle$ is *ferromagnetic*; it is energetically favorable for the neighboring spins to *align* (both $+1$ or both -1). If $J_{ij} = -1$, then we say the edge $\langle ij \rangle$ is *anti-ferromagnetic*; it is energetically favorable for the neighboring spins to *anti-align* (either $+1$ and -1 or -1 and $+1$). If all edges were ferromagnetic, it would be easy to minimize the energy — all spins would align. But anti-ferromagnetic edges can generate *frustration.* This means it is not possible to minimize $-J_{ij} Z_i Z_j$ for all edges simultaneously.

For purposes of visualization, it is convenient to represent spins by lattice cells– i.e. by squares on the 2D square lattices or by cubes in the 3D cubic lattice. There is a $-J_{ij} Z_i Z_j$ term coupling neighboring spins associated with each edge where two squares meet in 2D on each face where two cubes meet in 3D.

Consider a site in 2D where 4 edges meet. If *one* of these edges is antiferromagnetic and the other three are ferromagnetic, then there must be an odd number of *excited* edges meeting at this site. More generally, if the number of antiferromagnetic edges is odd, then there must be an odd number of excited edges and if the number of antiferromagnetic edges is even, then there must be an even number of excited edges. If the number of $J = -1$ edges meeting at a site is odd, we say that there is an *Ising vortex* at that site. For any spin configuration, there are *domain walls* of excited edges, where the walls end on Ising vortices.

Minimizing the energy, then, is equivalent in 2D to finding the minimum “chain” of excited edges with boundary points at the positions of the Ising vortices. There is a poly-time classical algorithm that finds the minimal chain. In 3D, there is an Ising vortex on an edge if there are an odd number of $J = -1$ faces that meed at that edge. These vortices form closed loops, and each spin configuration has a domain wall of excited *faces* bounded by the vortex loops. To minimize the energy, then, we find the minimum *area* surface with a specified 1D boundary; *this* problem is NP-hard. Finding the minimum energy configuration is hard because there are many ways for the domain walls to be "pinned" — stuck at local minima in the energy, such that many spins need to flip at once to find a lower energy configuration. *Local searching* for the global energy minimum fails.

In fact, there are also NP-hard spin glass problems in 2D, if we introduce local *magnetic field* terms in H as well as antiferromagnetic terms. For example, on a square lattice consider

$$H = -\sum_{\langle ij\rangle} J_{ij} Z_i Z_j - \sum_i h_i Z_i \tag{6.238}$$

where $J_{ij} \in \{1, 0, -1\}$ and $h_i \in \{1, 0, -1\}$. The local magnetic field $\{h_i\}$ compounds the frustration: Each spin wants to align with the local field, but by doing so the edge connecting the spins might become excited.

So we see that minimizing the energy of a geometrially 2-local classical Hamiltonian can be NP-hard because of frustration — there is no way to satisfy all the *clauses* and there are many local minima of the energy that are not global minima. In the quantum local Hamiltonian problem, we have $H = \sum_a H_a$ where the $\{H_a\}$ might not be mutually commuting; hence we might expect the problem to be even harder— that the H_a's cannot be simultaneously diagonalized compounds the frustration even further. Indeed the ground state could be highly entangled, with no succinct classical description. Let's try to characterize the hardness of the quantum problem.

6.10.3 The quantum k-local Hamiltonian problem

Let $H = \sum_a H_a$ be an n-qubit Hamiltonian that is *k-local* — each H_a acts nontrivially on at most k qubits, where k is a constant, and $||H_a|| \leq h =$ constant. The $2^k \times 2^k$ matrix H_a is specified to poly(n) bits of precision.

We are promised that the ground state energy E_0 , the lowest eigenvalue of H, satisfies either

1. $E_0 \leq E_{\text{low}}$, or
2. $E_0 > E_{\text{high}}$,

where $E_{\text{high}} - E_{\text{low}} > \frac{1}{\text{poly}(n)}$.
We are to output:

$$f(H) = \begin{cases} 1 & \text{if } E_0 \leq E_{\text{low}}, \\ 0 & \text{if } E_0 > E_{\text{high}}. \end{cases} \tag{6.239}$$

Note that in the formulation of the problem, there is a *promise gap* so that we can answer the YES/NO question by determining E_0 to $1/\text{poly}(n)$ accuracy.

We already know that this problem is NP-hard, even in the special case where H is classical (all H_a commute). Also recall there is a class analogous to NP for randomized computation (MA) and quantum computation (QMA): A language L is in QMA if there

exists a poly-size uniform quantum circuit family (the *verifier* V) and a single-qubit measurement $\{\Pi_0, \Pi_1\}$ such that:
1. If $x \in L$ there is a witness $|\psi_x\rangle$ such that:

$$\langle\Psi|\Pi_1|\Psi\rangle \geq 2/3 \quad \text{where} \quad |\Psi\rangle = V(|\psi_x\rangle \otimes |x\rangle \otimes |0\rangle^*). \tag{6.240}$$

2. If $x \notin L$, then $\langle\Psi|\Pi_1|\Psi\rangle \leq 1/3$ for all $|\psi_x\rangle$.

We claim: the k-local Hamiltonian problem is QMA-complete for $k \geq 5$ (famously proved by Kitaev: see Chap. 14 of KSV). The result can be improved to geometrically 2-local H in 2D (for qubits) or geometrically 2-local H in 1D (for higher dimensional qudits, with the local dimension d sufficiently large).

To verify this claim, we need to show:

1. the k-local Hamiltonian problem is in QMA.
2. *Any* problem in QMA is reducible to the k-local Hamiltonian problem. We'll show this for $k = 5$ and without geometrical constraints, as in Kitaev's original discovery.

We have already shown part (1). We have seen that if the ground state $|\psi_0\rangle$ is provided as a witness, then we can compute E_0 to $1/\text{poly}(n)$ accuracy using a circuit of size $\text{poly}(n)$ that performs the phase estimation algorithm. But how to achieve the reduction (2)? For the reduction, we'll follow the strategy used to show that 3-SAT is NP-complete: For any problem in QMA, we'll construct a witness that encodes the whole history of the computation performed by the verifier, and a Hamiltonian H such that computing the ground state energy of H amounts to checking that each step in the computation is valid.

For a given problem in QMA, suppose that the verifier circuit V_T has T gates: $U_1, U_2, \ldots U_T$ chosen from a universal set, where each U_t acts on at most two qubits. For the corresponding k-local Hamiltonian problem, we'll suppose that Merlin provides the history state encoding the computation performed by the verifier:

$$|\eta\rangle = \frac{1}{\sqrt{T+1}} \sum_{t=0}^{T} |\psi(t)\rangle \otimes |t\rangle, \tag{6.241}$$

where $|\psi(t)\rangle = (U_t U_{t-1} \ldots U_1)|\psi_{\text{initial}}\rangle$; here $|\psi_{\text{initial}}\rangle = |x\rangle \otimes |0\rangle^* \otimes |\psi_x\rangle$, where $|x\rangle$ specifies the instance of the QMA problem, $|0\rangle^*$ is the initial state of scratch space used by the verifier, $|\psi_x\rangle$ is the quantum witness testifying that x should be accepted, and $|\psi(t)\rangle$ is state of the quantum computer obtained after the first t steps of the verifier circuit have been executed. The state $|t\rangle$ is the state of a clock register that records the time $t \in \{0, 1, \cdots, T\}$; since $\langle t|s\rangle = \delta_{ts}$, the $T + 1$ states appearing in superposition in state $|\eta\rangle$ are mutually orthogonal, so $|\eta\rangle$ is properly normalized: $\langle\eta|\eta\rangle = 1$.

6.10.4 Constructing and analyzing the Hamiltonian

Now we want to choose our Hamiltonian H so that it locally "checks" the history encoded in $|\eta\rangle$; that is, H will impose an energetic penalty if a step in the circuit is invalid, or if the input is not accepted.

We will choose H to be of the form

$$H = H_{\text{in}} + H_{\text{out}} + H_{\text{prop}} + H_{\text{clock}}. \tag{6.242}$$

The purpose of H_{in} is to enforce that the verifier circuit's input qubits (other than the witness itself) are set to the right initial values. E.g. for each scratch qubit labeled j that should be in the state $|0\rangle$ at time $t = 0$, we include in H_{in} the term

$$H_{\text{in}}^{(j)} = (|1\rangle\langle 1|)^{(j)} \otimes I^{(\text{else})} \otimes (|0\rangle\langle 0|)^{(\text{clock})}. \tag{6.243}$$

There is an energy penality of 1 if the scratch qubit is set to $|1\rangle$ rather than $|0\rangle$ as the execution of the verifier circuit begins (time $t = 0$). Similar terms in H_{in} enforce that the input register is set to the desired value x. The purpose of H_{out} is to impose a penalty if the verifier circuit for the QMA problem fails to accept:

$$H_{\text{out}} = (|0\rangle\langle 0|)^{(\text{output})} \otimes I^{(\text{else})} \otimes (|T\rangle\langle T|)^{(\text{clock})}. \tag{6.244}$$

There is an energy cost of 1 if the output qubit has the value $|0\rangle$ rather than $|1\rangle$ after the verifier circuit is fully executed (time $t = T$). The purpose of H_{clock} is to impose a penalty if the clock register is not properly encoded (we'll return to this issue later).

The purpose of the H_{prop} is to impose a penalty if the state $|\psi(t)\rangle$ does not have the form $U_t|\psi(t-1)\rangle$; i.e., was not obtained by faithfully executing step t of the verifier circuit. Hence we write:

$$H_{\text{prop}} = \sum_{t=1}^{T} H_{\text{prop}}(t), \tag{6.245}$$

where

$$H_{\text{prop}}(t) = \frac{1}{2}\left(I \otimes |t\rangle\langle t| + I \otimes |t-1\rangle\langle t-1| - U_t \otimes |t\rangle\langle t-1| - U_t^\dagger \otimes |t-1\rangle\langle t|\right). \tag{6.246}$$

The action of $H_{\text{prop}}(t)$ on the relevant part of the valid history state $|\eta\rangle$ is

$$\begin{aligned} |\psi(t-1)\rangle \otimes |t-1\rangle &\mapsto \frac{1}{2}\left(|\psi(t-1)\rangle \otimes |t-1\rangle - U_t|\psi(t-1)\rangle \otimes |t\rangle + \dots\right), \\ |\psi(t)\rangle \otimes |t\rangle &\mapsto \frac{1}{2}\left(|\psi(t)\rangle \otimes |t\rangle - U_t^\dagger|\psi(t)\rangle \otimes |t-1\rangle + \dots\right). \end{aligned} \tag{6.247}$$

Acting on $|\eta\rangle$, the terms $I \otimes |t\rangle\langle t|$ and $-U_t \otimes |t\rangle\langle t-1|$ in H_{prop} give canceling contributions. Hence

$$H_{\text{prop}}|\eta\rangle = 0 \quad \text{if } |\eta\rangle \text{ is a valid history state.} \tag{6.248}$$

Therefore, a valid history state (where the state at time t is obtained from the state at time $t-1$ by applying the proper gate), such that the initial state is also valid, is a null vector of both H_{prop} and H_{in}. Furthermore, if the quantum verifier accepts the input with probability $1 - \epsilon$, then

$$E_0 \leq \langle \eta | H_{\text{out}} | \eta \rangle = \frac{\epsilon}{T+1}. \tag{6.249}$$

(If the history is valid, the only term in the Hamiltonian that makes a nonzero contribution to $\langle H \rangle$ is the term that penalizes an incorrect final readout.) Note also that it is possible to amplify the success probability by repeating the verification of multiple copies of the witness. Actually, the amplification is a little bit subtle: Merlin might try to fool Arthur: Instead of sending a product state $|\psi_x\rangle^{\otimes m}$ (m copies of the witness) he might send an entangled state instead. But the amplification still works even in that case. Each copy sent by Merlin may be a mixed state (obtained from the partial trace over the other copies), a mixture of a state the verifier accepts with probability $\geq 2/3$

and of another state that it might reject. But Merlin cannot fool Arthur into accepting after many trials unless there is some state occuring in the ensemble of pure states that is accepted with high probability in each trial.

So now we have seen that for a problem in QMA such that the verifier accepts with high probability, the corresponding Hamiltonian H has an eigenvector with eigenvalue close to zero. There are two things left to show:

- If the verifier rejects with high probability, then
$$E_0 \geq 1/\mathrm{poly}(n) \tag{6.250}$$
(then we can choose the promise gap of size $1/\mathrm{poly}(n)$, such that $E_0 \leq E_{\mathrm{low}}$ for a YES answer and $E_0 > E_{\mathrm{high}}$ for a NO answer)
- So far H_{prop} is not local! It acts on the clock register, which is a $(T+1)$-dimensional system, and $T = \mathrm{poly}(n)$. We need to show we can encode the clock using qubits such that $H_{\mathrm{prop}} + H_{\mathrm{clock}}$ is k-local.

We'll come back to the issue of encoding the clock later. First let's try to understand better the spectrum of $H = H_{\mathrm{in}} + H_{\mathrm{out}} + H_{\mathrm{prop}}$.

Diagonalizing H_{prop}

Let's start by considering H_{prop}. We've seen that a valid history state is a null vector of H_{prop} (has eigenvalue zero). What are the other eigenspaces and eigenvalues of H_{prop}?

It's easier to compute the spectrum of H_{prop} by transforming to a *rotating frame* basis that *freezes the motion* of the state $|\psi(t)\rangle$. That is, let

$$V_t = U_t U_{t-1} \ldots U_1 \tag{6.251}$$

(the unitary applied after the first t steps of the circuit). And consider $V = \sum_{t=0}^{T} V_t \otimes |t\rangle\langle t|$. The the history state $\eta = \frac{1}{\sqrt{T+1}} \sum_{t=0}^{T} V_t |\psi(0)\rangle \otimes |t\rangle\langle t|$ is mapped by $V^\dagger$ to the "history" state for the case where $|\psi(t)\rangle$ does not depend on t at all:

$$V^\dagger |\eta\rangle \frac{1}{\sqrt{T+1}} \sum_{t=0}^{T} V_t^\dagger V_t |\psi(0)\rangle \otimes |t\rangle\langle t| = |\psi(0)\rangle \otimes \sum_{t=0}^{T} |t\rangle\langle t|. \tag{6.252}$$

Under this basis change, the Hamiltonian transforms as

$$\begin{aligned} H'_{\mathrm{prop}} = V^\dagger H_{\mathrm{prop}} V &= \sum_{t=0}^{T} \frac{1}{2} \left(I \otimes |t\rangle\langle t| + I \otimes |t-1\rangle\langle t| \right) \qquad (6.253) \\ &\quad - \sum_{t=0}^{T} \frac{1}{2} \left(V_t^\dagger V_{t-1} \otimes |t\rangle\langle t-1| + V_{t-1}^\dagger V_{t-1} \otimes |t-1\rangle\langle t| \right) \\ &= \sum_{t=0}^{T} \frac{1}{2} \left(|t\rangle\langle t| + |t-1\rangle\langle t-1| - |t\rangle\langle t-1| - |t-1\rangle\langle t| \right). \end{aligned}$$

This transformed Hamiltonian H'_{prop} acts nontrivially only on the clock register. It is a sum of overlapping 2×2 blocks:

$$H'_{\mathrm{prop}} = \sum_{t=1}^{T} \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}_{t-1,t}, \tag{6.254}$$

where the subscript means the matrix acts on the space spanned by $\{|t-1\rangle, |t\rangle\}$. Because of the overlaps, H'_{prop} is actually $\begin{pmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{pmatrix}$ in each block, except in the spaces spanned by $\{|0\rangle, |1\rangle\}$ and $\{|T-1\rangle, |T\rangle\}$, where it is

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{pmatrix}_{0,1} \quad \text{and} \quad \begin{pmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}_{T-1,T}. \tag{6.255}$$

That is, H'_{prop} acts on the clock as a $(T+1) \times (T+1)$ matrix

$$H'_{\text{prop}} = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & 0 & \cdots & & & \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \cdots & & \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \cdots & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & \cdots & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & \\ & & \cdots & 0 & -\frac{1}{2} & \frac{1}{2} & \end{pmatrix}; \tag{6.256}$$

its entries are $(-\frac{1}{2}, -\frac{1}{2}, .., -\frac{1}{2})$ just above and below the diagonal, $(\frac{1}{2}, 1, 1, .., 1, \frac{1}{2})$ on the diagonal, and 0 elsewhere.

We may express it as $H'_{\text{prop}} = I - \frac{1}{2}M$, where

$$M = \begin{pmatrix} 1 & 1 & 0 & \cdots & & & \\ 1 & 0 & 1 & 0 & \cdots & & \\ 0 & 1 & 0 & 1 & 0 & \cdots & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & \cdots & 0 & 1 & 0 & 1 & \\ & & \cdots & 0 & 1 & 1 & \end{pmatrix}. \tag{6.257}$$

That is,

$$\begin{aligned} M &: |t\rangle \mapsto |t+1\rangle + |t-1\rangle, \quad t \in \{1, 2, .., T-1\}, \\ M &: |0\rangle \mapsto |0\rangle + |1\rangle, \\ M &: |T\rangle \mapsto |T-1\rangle + |T\rangle. \end{aligned} \tag{6.258}$$

Diagonalizing M also diagonalizes H'_{prop}.

We can diagonalize M by Fourier transforming. The eigenvectors have the form

$$|\omega\rangle = \sum_{t=0}^{T} e^{i\omega t}|t\rangle. \tag{6.259}$$

In the expansion of $M|\omega\rangle$, the coefficient of $|t\rangle$ for $t \in \{1, 2, .., T-1\}$ is

$$e^{i\omega(t-1)} + e^{i\omega(t+1)} = (e^{i\omega} + e^{-i\omega})e^{i\omega t} \tag{6.260}$$

$$= (2\cos\omega)e^{i\omega t}. \tag{6.261}$$

Thus the action on $|t\rangle$ is multiplication by $2\cos\omega$, which does not depend on the sign of w. To construct an eigenstate of M, then, we may consider a linear combination of $|\omega\rangle$ and $|-\omega\rangle$, where the value of $|\omega\rangle$ is chosen so that M acts properly on the states at the *boundary* — i.e. on $|t=0\rangle$ and $|t=T\rangle$.

The coefficient of $|0\rangle$ in $M|\omega\rangle$ is $1 + e^{iw}$; therefore, the coefficient of $|0\rangle$ in $M\left(e^{i\omega/2}|w\rangle + e^{-i\omega/2}|-\omega\rangle\right)$ is

$$\begin{aligned} e^{i\omega/2}(1+e^{i\omega}) + e^{-i\omega/2}(1+e^{-i\omega}) &= (e^{i\omega}+e^{-i\omega})(e^{i\omega/2}+e^{-i\omega/2}) \\ &= 2\cos\omega(e^{i\omega/2}+e^{-i\omega/2}). \end{aligned} \tag{6.262}$$

If follows that the "boundary condition" at $t = 0$ is consistent, for any value of ω, with an eigenvalue of $2\cos\omega$ for the vectors $e^{i\omega/2}|\omega\rangle + e^{-i\omega/2}|-\omega\rangle$.

But we must also consider the other boundary, at $t = T$. The coefficient of $|T\rangle$ in $e^{i\omega/2}|\omega\rangle + e^{-i\omega/2}|-\omega\rangle$ is $e^{i\omega/2}e^{i\omega T} + e^{-i\omega/2}e^{-i\omega T}$, while the coefficient of $|T\rangle$ in $M\left(e^{i\omega/2}|\omega\rangle + e^{-iw/2}|-\omega\rangle\right)$ is

$$(1+e^{-i\omega})e^{i\omega/2}e^{i\omega T} + (1+e^{i\omega})e^{-i\omega/2}e^{-i\omega T}. \tag{6.263}$$

We are to choose ω so that

$$(1+e^{-iw})e^{iw/2}e^{iwT} + (1+e^{iw})e^{-iw/2}e^{-iwT} = (e^{iw}+e^{-iw})(e^{iw/2}e^{iwT}+e^{-iw/2}e^{-iwT}). \tag{6.264}$$

After some cancellations, this condition becomes

$$e^{iw/2}e^{iwT} + e^{-iw/2}e^{-iwT} = e^{3iw/2}e^{iwT} + e^{-3iw/2}e^{-iwT} \tag{6.265}$$

$$\implies \cos\left(\omega\left(T+\frac{1}{2}\right)\right) = \cos\left(\omega\left(T+\frac{3}{2}\right)\right). \tag{6.266}$$

This condition is satisfied provided that

$$w(T+1) = \pi k \quad \text{where} \quad k = \text{integer}. \tag{6.267}$$

We have therefore established that $\{2\cos\omega_k\}$ are $T+1$ distinct eigenvalues of M where $\omega_k = \frac{\pi}{T+1}k$ for $k \in \{0,1,2,..,T\}$, and that the corresponding eigenvectors (up to normalization) are

$$|\psi_k\rangle = \sum_{t=0}^{T} \cos\left(\omega_k\left(t+\frac{1}{2}\right)\right)|t\rangle; \tag{6.268}$$

hence the eigenvalues of H'_{prop} are $E_k = 1 - \cos\omega_k = 2\sin^2\omega_k$. These $T+1$ eigenstates are a complete basis, and the two smallest eigenvalues are:

$$E_0 = 0, \tag{6.269}$$

$$E_1 = 2\sin^2\left(\frac{\pi}{2(T+1)}\right) \approx \frac{\pi^2}{2(T+1)^2}. \tag{6.270}$$

If $T = \text{poly}(n)$, then, the eigenvalue gap for H_{prop} is $E_1 - E_0 \geq \frac{1}{\text{poly}(n)}$.

Lower bound on the energy when the witness is rejected

But we want to consider the spectrum for the full Hamiltonian $H = H_{\text{in}} + H_{\text{out}} + H_{\text{prop}}$ (ignoring H_{clock} for now). For $H_{\text{in}} + H_{\text{out}}$, the null space is spanned by all vectors that have a valid input at $t = 0$ and answer YES (are accepted) at $t = T$:

$$(H_{\text{in}} + H_{\text{out}})\,|\text{valid input, accepted output}\rangle = 0, \tag{6.271}$$

while $\langle H_{\text{in}} + H_{\text{out}}\rangle \geq 1$ for all vectors orthogonal to this null space. Thus $H_{\text{in}} + H_{\text{out}}$ has eigenvalue gap $= 1$. Now, if the verifier accepts the input with the probability one, then there is a simultaneous null eigenvector of $H_{\rm prop}$ and of $H_{\rm in}+H_{\rm out}$. That is, there is a valid history with a valid input where the output is accepted. But if the acceptance probability is small, that means the angle between these two null spaces cannot be too small, since there is no vector that comes close to lying within both null spaces. We can relate this angle to the ground state energy of the full Hamiltonian $H = H_{\rm in} + H_{\rm out} + H_{\rm prop}$.

In general, suppose that H_1 and H_2 are two Hermitian operators, each with lowest eigenvalue zero, and eigenvalue gap at least Δ. Then

$$H_1 \geq \Delta(I - \Pi_1) \quad \text{where } \Pi_1 \text{ is projection onto null space of } H_1 \tag{6.272}$$

$$H_2 \geq \Delta(I - \Pi_2) \quad \text{where } \Pi_2 \text{ is projection onto null space of } H_2 \tag{6.273}$$

Thus $H_1 + H_2 \geq \Delta(2I - \Pi_1 - \Pi_2)$ and $\langle H_1 + H_2\rangle \geq 2\Delta - \Delta\langle \Pi_1 + \Pi_2\rangle$. Now suppose $|\psi_1\rangle$ and $|\psi_2\rangle$ are two vectors such that $|\langle\psi_1|\psi_2\rangle| = \cos\theta$ for $0 \leq \theta \leq \pi/2$. With suitable phase conventions we may choose a basis in the two-dimensional space spanned by $|\psi_1\rangle$ and $|\psi_2\rangle$ such that

$$|\psi_1\rangle = \begin{pmatrix} \cos\theta/2 \\ \sin\theta/2 \end{pmatrix} \qquad |\psi_2\rangle = \begin{pmatrix} \cos\theta/2 \\ -\sin\theta/2 \end{pmatrix} \Longrightarrow \tag{6.274}$$

$$|\psi_1\rangle\langle\psi_1| + |\psi_2\rangle\langle\psi_2| = \begin{pmatrix} 2\cos^2\theta/2 & 0 \\ 0 & 2\sin^2\theta/2 \end{pmatrix}, \tag{6.275}$$

and therefore, in any state

$$\langle |\psi_1\rangle\langle\psi_1| + |\psi_2\rangle\langle\psi_2|\rangle \leq 2\cos^2\theta/2 = 1 + \cos\theta. \tag{6.276}$$

If follows that, if Π_1 and Π_2 are projectors, where the maximum overlap between spaces projected by Π_1 and Π_2 is $|\langle\psi_1|\psi_2\rangle| = \cos\theta$, then $\langle\Pi_1 + \Pi_2\rangle \leq 1 + \cos\theta$. Thus

$$\langle H_1 + H_2\rangle \geq 2\Delta - \Delta\langle\Pi_1 + \Pi_2\rangle \tag{6.277}$$

$$\geq \Delta(1 - \cos\theta) = 2\Delta\sin^2\theta/2. \tag{6.278}$$

Having related the expectation value of a sum of nonnegative Hermitian operators to the angle between their null spaces, we now need to estimate the angle between null spaces of $H_1 = H_{\rm in} + H_{\rm out}$ and $H_2 = H_{\rm prop}$. That is, we want to find

$$\cos^2\theta = \max|\langle\eta_1|\eta_2\rangle|^2 \tag{6.279}$$

$$= \max\langle\eta_2|\Pi_1|\eta_2\rangle \tag{6.280}$$

where we maximize over η_1 in the null space of H_1 and η_2 in the null space H_2, and Π_1 is the projector onto the null space of H_1.

A vector in the null space of $H_2 = H_{\rm prop}$ is a valid history state $|\eta_2\rangle = \frac{1}{\sqrt{T+1}}\sum_{t=0}^{T}|\psi(t)\rangle \otimes |t\rangle$. The projector onto null space of H_1 acts trivially on states with $t \in \{1, 2, .., T-1\}$ so after transforming to the rotating frame

$$\langle\eta_2'|\Pi_1|\eta_2'\rangle = \frac{T-1}{T+1} + \frac{1}{T+1}\langle\tilde\eta_2|\Pi_{\rm in} + \Pi_{\rm out}'|\tilde\eta_2\rangle, \tag{6.281}$$

where $\tilde\eta_2$ is a state of the non-clock variables, $\Pi_{\rm in}$ projects onto a valid input state, and $\Pi_{\rm out}'$ projects onto

$$V_T^\dagger(|1\rangle^{\rm out} \otimes (\text{anything else})) \tag{6.282}$$

(because we have transformed to the rotating frame, and the history state is valid). We know that $\langle\tilde\eta_2|\Pi_{\rm in} + \Pi_{\rm out}'|\tilde\eta_2\rangle \leq (1 + \cos\phi)$ where ϕ is the angle between the spaces that

Π_{in} and Π_{out} project onto, as we showed above. This angle ϕ is given by $\cos^2\phi = \epsilon$ where ϵ is the max. acceptance probability — that is, if the input is valid (in the support of Π_{in}) and the history is valid, then the probability of $|1\rangle^{\text{out}}$ is at most ϵ.

Via eq.(6.281), this upper bound $\langle\tilde{\eta}_2|\Pi_{\text{in}} + \Pi'_{\text{out}}|\tilde{\eta}_2\rangle \leq 1+\sqrt{\epsilon}$ provides an upper bound on the expectation value of Π_1 in a valid history state, and hence an lower bound on the angle θ between the H_1 and H_2 null eigenspaces:

$$\cos^2\theta \leq \frac{T-1}{T+1} + \frac{1}{T+1}(1+\sqrt{\epsilon}) = 1 - \frac{1-\sqrt{\epsilon}}{T+1}, \tag{6.283}$$

where ϵ is the maximum acceptance probability. Now, $\Delta = 2\sin^2\left(\frac{\pi}{2(T+1)}\right)$ is a lower bound for the gap of H_1 or H_2 and $\langle H_1 + H_2\rangle \geq 2\Delta\sin^2\frac{\theta}{2}$, where $\sin^2\theta = 1-\cos^2\theta \geq \frac{1-\sqrt{\epsilon}}{T+1}$. Using $\sin^2\frac{\theta}{2} = \frac{\sin^2\theta}{4\cos^2\frac{\theta}{2}} \geq \frac{1}{4}\sin^2\theta$, we have

$$\begin{aligned} E_0 &\geq 4\sin^2\left(\frac{\pi}{2(T+1)}\right)\times\frac{1}{4}\left(\frac{1-\sqrt{\epsilon}}{T+1}\right) \geq \sin^2\left(\frac{\pi}{2(T+1)}\right)\times\left(\frac{1-\sqrt{\epsilon}}{T+1}\right) \\ &\geq \text{constant}\times\frac{1-\sqrt{\epsilon}}{(T+1)^3} = \frac{1-\sqrt{\epsilon}}{\text{poly}(n)}. \end{aligned} \tag{6.284}$$

To summarize, we have shown that if the acceptance probability is $\geq 1-\epsilon$, then $E_0 \leq \frac{\epsilon}{T+1}$ and if acceptance probability is $\leq \epsilon$, then $E_0 \geq \text{constant}\times(1-\sqrt{\epsilon})\frac{1}{(T+1)^3}$. With suitable amplification to make ϵ small (compared to $1/(T+1)^3$) in the case where the answer is YES, we have reduced the QMA problem to an instance of the local Hamiltonian problem.

Encoding the clock

Except ... we still need to see that the Hamiltonian can be made *local.* We'd like to encode the clock register using qubits. One way is to use a *unary* encoding with T qubits labeled by $i = \{1, 2, 3, \ldots, T\}$, where

$$\begin{aligned} |t=0\rangle &= |000..0\rangle \\ |t=1\rangle &= |100..0\rangle \\ |t=2\rangle &= |110..0\rangle \\ &\text{etc.} \end{aligned} \tag{6.285}$$

We can add a term to the Hamiltonian that imposes an energy penalty on the clock if its state is not validly encoded. The encoding is valid if a 0 is never followed by a 1. Therefore we choose

$$H_{\text{clock}} = \sum_{i=1}^{T-1}\left(|01\rangle\langle 01|\right)_{i,i+1}, \tag{6.286}$$

so that the only null vectors of H_{clock} are valid clock states.

The term H_{prop} in the Hamiltonian contains operators $|t\rangle\langle t|$, $|t\rangle\langle t-1|$, $|t-1\rangle\langle t|$ acting on the clock register. If only validly encoded clock states are allowed, we can project onto the state $|t\rangle$ by acting on just two neighboring qubits:

$$|t\rangle\langle t| = (|10\rangle\langle 10|)_{t,t+1} \tag{6.287}$$

for $t = 1, 2, 3, \ldots, T-1$, and acting on only a single qubit suffices for $t = 0$ or $t = T$.

The operators that advance or retard the time act on three adjacent qubits:

$$\begin{aligned} |t\rangle\langle t-1| &= (|110\rangle\langle 100|)_{t-1,t,t+1}, \\ |t-1\rangle\langle t| &= (|100\rangle\langle 110|)_{t-1,t,t+1} \end{aligned} \tag{6.288}$$

(for $1 \leq t \leq T-1$). Acting on three qubits suffices to locate the position of the "domain wall" between the 0's and 1's in the bit string, and to move the wall one position to the left or to the right; for $t = 0$ action on just two qubits suffices to move the wall one step to the right, and for $t = T$ action on two qubits suffices to move the wall one step to the left. Since these clock terms act on at most three qubits, the term $U_t \otimes |t\rangle\langle t-1|$ acts on at most 5 qubits, if U_t is a 2-qubit gate. Thus with this clock encoding, the Hamiltonian

$$H = H_{\rm in} + H_{\rm out} + H_{\rm prop} + H_{\rm clock} \tag{6.289}$$

is 5-local. Because the clock Hamiltonian annihilates validly encoded clock states, it trivially commutes with the rest of the terms in the Hamiltonian H, and has an energy gap of 1. Therefore, our previous analysis of the ground state energy of H still stands. This completes the demonstration that any QMA problem is reducible to the problem of estimating the ground state energy (with a $1/\text{poly}(n)$ promise gap) of a 5-local Hamiltonian.

Comments

Thus we have shown that the 5-local Hamiltonian problem is a "natural" QMA-complete problem, much as 3-SAT is a natural NP-complete problem. But while in the classical case many *practical* problems have been shown to be NP-complete, the family of problems that have been shown to be QMA-complete is still rather small, and the problems seem relatively "artificial." In any case, its interesting to see that quantum local Hamiltonian problems seem to be harder than classical ones (if QMA $\neq$ NP).

I won't discuss the tricks for reducing the QMA-complete problem to $k = 2$ (which involves clever use of perturbation theory) or for making H geometrically local (which involves encoding the clock more cleverly, among other things).

Another interesting direction to pursue using these ideas is to show that any problem in BQP can be solved using adiabatic quantum computing. The idea is to replace

$$H_{\rm prop} \to H_{\rm prop}(s) = (1-s)H_{\rm clock-init} + sH_{\rm prop} \tag{6.290}$$

where the null space of $H_{\rm clock-init}$ fixes the clock at $|t = 0\rangle$ and s varies in $[0,1]$. Then the ground state of $H(s = 0)$ is easy to construct, and the ground state of $H(s = 1)$ is the valid history state. The eigenvalue gap of $H(s)$ stays $> 1/\text{poly}(n)$ for $s \in [0,1]$, so the history state can be prepared in polynomial time by adiabatically varying s. Once we have prepared the history state, we can measure the clock, projecting out $|t = T\rangle$ with probability $1/(T+1) = 1/\text{poly}(n)$. And once we have $|\psi(T)\rangle$ we can measure the output qubit to find out if the circuit accepts. At a modest additional cost, we can can substantially improve the probability of preparing $|\psi(T)\rangle$, by adding a long "runway" after time $t = T$, so that the state of the computer remains fixed for S additional time steps; then measuring the clock prepares $|\psi(T)\rangle$ with probability $(S+1)/(S+T+1)$.