CMSC 657: Introduction to Quantum Information Processing
Lecture 14

Instructor: Daniel Gottesman

Fall 2024

1 Simulating Quantum Systems

1.1 The Goal of Quantum Simulation

One example of a quantum system, and a common one you will see in introductory quantum mechanics courses, is a single particle moving in one dimension. Here, we can think of the basis states as labelled by position x, and $|\psi\rangle = \int dx\, \psi(x)|x\rangle$. ($\psi(x)$ is generally an L_2 function, square integrable.) There is a position operator $\hat{x}$ defined by $\hat{x}|x\rangle = x|x\rangle$. There is also a momentum operator $\hat{p}$ telling us about the motion of the particle, but in quantum mechanics, position and momentum don't commute, so in the position representation,

$$\hat{p} = -i\hbar \frac{\partial}{\partial x}. \tag{1}$$

That is,

$$\hat{p}|\psi\rangle = -i\hbar \int dx\, \psi'(x)|x\rangle. \tag{2}$$

The fact that position and momentum don't commute means that they don't have simultaneous eigenstates, which is behind the famous Heisenberg Uncertainty Principle, which says that there is a limit to having both a well-defined position and momentum at the same time. If the wavefunction is particularly localized in position, that means it will be very spread out in momentum (contain many different momentum eigenstates). The *kinetic energy* of a particle of mass m, the amount of energy tied up in the particle's movement, is

$$\frac{\hat{p}^2}{2m} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}. \tag{3}$$

Often, there are constraints on the movement of a particle, which translate to an energy cost to be in different positions. This is encapsulated as a *potential energy* $V(\hat{x})$ and H is the sum of the potential and kinetic energy:

$$H = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(\hat{x}). \tag{4}$$

A single particle in one dimension (or three dimensions, or any fixed dimension) is actually easy to simulate classically, but if you start to talk about many particles, it gets harder rapidly, exponentially as the number of particles. The basis states are labelled by a list of positions of the particles, and of course we can have arbitrary superpositions of these. A quantum computer can simulate such a system by discretizing the position into small steps. This still leaves an exponential number of basis states, but only polynomially many qubits are needed to capture the state of the system. In a classical computer, however, you would need to separately keep track of the exponentially many amplitudes of all those basis states.

Another class of systems is a bit more natural for simulating on a quantum computer, namely systems composed of a set of finite-dimensional particles interacting with each other. These are often called "spin systems," because one way they can occur is by having a collection of atoms with fixed positions but variable

spin (either of the electrons or the nucleus). The spins of different atoms can interact with each other electromagnetically, creating a Hamiltonian.

For our purposes, we will consider the system to be composed of qubits, or sometimes higher-dimensional qudits. The Hamiltonian is then a sum of terms representing interactions between different groups of qubits. Most often, we talk about *local* Hamiltonians, which in this context means that each term involves only a constant number of qubits (independent of the total number of qubits in the system). Thus, we can write

$$H = \sum_i H_i, \tag{5}$$

with H_i of the form $A \otimes I$, a tensor product of an operator acting on the qubits in the set S_i with the identity on other qubits. For instance, we might have qubits numbered $0, \ldots, n-1$, with $H_i = X_i X_{i+1} + Z_i Z_{i+1}$. The set S_i of qubits acted on by H_i is $\{i, i+1\}$. In this case, all the terms H_i are basically the same, just shifted to a different set of qubits. That is common, but is not always true. Also, it is often the case (as here) that the Hamiltonian does not change with time, but that doesn't have to be true either.

When we say that we want to *simulate* a Hamiltonian, that can mean a couple of different things. Sometimes it is used to mean "find a description of the ground state of the Hamiltonian." This turns out to be hard in general, even for a quantum computer (subject to the usual complexity-theoretic caveats), although in many specific cases, it is not and can even be done on a classical computer. Another possible meaning is to have the computer replicate the *dynamics* of the system, meaning to calculate the state of the system at time t for arbitrary times in some interval. In a classical simulation of a quantum system, typically, we can ask the computer to output a full description of the quantum state (or an approximation to it) at time t. In a quantum computer, the state of the qubits is a representation of the quantum state of the system, but because of the limits on measurement in quantum mechanics, we don't learn the full wavefunction, and indeed a classical description of this would be so large, it would remove most of the quantum advantage. Instead, we focus on specific measurements or specific properties that we would like to learn about the system, and the computer outputs those by making an appropriate measurement on the representation of the system's state in its quantum memory.

More specifically, let us formulate a problem like this:

Problem 1 (Simulating Quantum Systems)**.** *Given Hamiltonian H, initial state $|\psi(0)\rangle$, time T, and measurement M, let $|\psi(t)\rangle$ be the solution of equation Schrödinger equation for H with this initial state. Find (to some specified accuracy) the probabilities P_i of the various measurement outcomes i when M is performed on the state $|\psi(T)\rangle$.*

Often, M is a Hermitian operator with eigenvalues m_i and the associated measurement is a projective measurement on the eigenstates $|i\rangle$ of M. Then instead of the detailed probabilities, we may be interested in the *expectation value* of M:

$$\langle\psi(T)|M|\psi(T)\rangle = \sum_i m_i |\alpha_i|^2, \tag{6}$$

where $|\psi(T)\rangle = \sum_i \alpha_i |i\rangle$.

If you want to find the probabilities exactly, a quantum computer can't do that, or at least we don't generally know how to do that better than with a classical computer. Since the idea is to have the state of the quantum computer be $|\psi(t)\rangle$ or some representation thereof, the way to find the probability of particular measurement outcomes is to do the measurement M on the state and repeat the state preparation/measurement process many times to accumulate statistics on how often each outcome occurs. This means the natural output of a quantum simulation is just an approximation to the true outcome distribution.

1.2 Simulating Commuting Hamiltonians

Solving Schrödinger's equation is actually very easy, in a sense. H is a Hermitian operator, so we can diagonalize it. This gives us the eigenstates of H. If $|\psi_E\rangle$ is an eigenstate of H with energy E and $|\psi(t=0)\rangle =$

$|\psi_E\rangle$, then I claim that $|\psi(t)\rangle = e^{i\phi(t)}|\psi_E\rangle$ for all t. Supposing this is true, we have

$$\begin{aligned} i\hbar\frac{d|\psi(t)\rangle}{dt} =& i\hbar(i\phi'(t))e^{i\phi(t)}|\psi_E\rangle && (7)\\ =& H|\psi(t)\rangle && (8)\\ =& Ee^{i\phi(t)}|\psi_E\rangle && (9)\\ -\hbar\phi'(t) =& E. && (10)\end{aligned}$$

Thus, $\phi(t) = -Et/\hbar$. (The constant of integration is set to 0 since we have set $\phi(0) = 0$.) Thus, the solution is

$$|\psi(t)\rangle = e^{-iEt/\hbar}|\psi(t=0)\rangle. \tag{11}$$

Energy eigenstates pick up a phase which grows linearly with t. For a general initial state, we can write $|\psi(t=0)\rangle = \sum_E \alpha_E|\psi_E\rangle$, and by linearity,

$$|\psi(t)\rangle = \sum_E \alpha_E e^{-iEt/\hbar}|\psi_E\rangle. \tag{12}$$

Unfortunately, this solution is rarely sufficient in practice. The problem is that for an n-qubit system, H is a matrix of size $2^n \times 2^n$, and diagonalizing it is too expensive for anything but the smallest quantum systems.

One case where it is sufficient is for a system of small size. If H acts only on a constant number of qubits, we can diagonalize H in constant time. This is actually true even if there are many qubits, but H only acts non-trivially on a set S of constant size, like a single term for a local Hamiltonian. In that case, the eigenbasis can be chosen to be an energy eigenbasis for the qubits in S tensored with standard basis states on the other qubits.

If we have a quantum computer with the state $|\psi\rangle$ in its memory, representing the state of the quantum system we wish to simulate, and we have such a Hamiltonian that acts on just the set S of qubits, we can create a unitary operator that performs the appropriate time evolution. The specific unitary we want can be written as

$$U(t) = e^{-iHt/\hbar} \tag{13}$$

in the case where H is time-independent. Here, the exponential of a matrix can be defined via the power series for e^x:

$$e^M = \sum_{k=0}^{\infty} \frac{1}{k!}M^k = I + M + \frac{1}{2}M^2 + \frac{1}{6}M^3 + \dots. \tag{14}$$

Matrix exponentials satisfy some of the same relationships as regular numerical exponentials, but often only when the matrices involved commute with each other. In particular, if M and N commute, then $e^M e^N = e^{M+N}$. When they don't commute, this is generally not true.

Note that $U(t)$ is automatically unitary when H is Hermitian: $U(t)^\dagger = \exp(+iHt/\hbar)$ by applying the adjoint to each term in the power series. Then

$$U(t)U(t)^\dagger = e^{-iHt/\hbar}e^{+iHt/\hbar} = e^0 = I, \tag{15}$$

since $-iHt/\hbar$ and $+iHt/\hbar$ commute, and similarly $U(t)^\dagger U(t) = I$.

Note that

$$\begin{aligned} e^{-iHt/\hbar}|\psi_E\rangle =& \sum_k \frac{1}{k!}(-i/\hbar)^k H^k|\psi_E\rangle && (16)\\ =& \sum_k \frac{1}{k!}(-i/\hbar)^k E^k|\psi_E\rangle && (17)\\ =& e^{-iEt/\hbar}|\psi_E\rangle. && (18)\end{aligned}$$

This shows that $U(t)$ is the unitary time evolution operator we want. It can be implemented straightforwardly by a rotation R that maps the energy eigenbasis of S and a standard basis; $R^\dagger$ will map the standard basis to the energy eigenbasis. Therefore, if we apply R, then a phase $e^{-iEt/\hbar}$ to the basis state corresponding to the eigenstate with energy E, then $R^\dagger$, we will have implemented $U(t)$.

Unfortunately, we really do need the Hamiltonian to act on only a few qubits to do this. The problem is that just as we can take any Hermitian operator H to get a unitary $U(t)$, we can take any unitary U and get a Hermitian operator $H_U = i\hbar \ln U$. (Again defining ln of a matrix via its power series.) The time evolution of H_U thus implements U in time 1. But if U is a unitary that solves some hard problem, we don't expect to be able to implement that with a small circuit. Indeed, that is basically the definition of a hard problem.

But there are cases where we can indeed simulate Hamiltonians acting on many qubits. One example is when the Hamiltonian is a local Hamiltonian which is a sum of terms that all commute with each other. Then

$$U(t) = e^{-it\sum_i H_i/\hbar} = \prod_i e^{itH_i/\hbar}. \tag{19}$$

Each term in the product is the evolution of a Hamiltonian acting on a constant number of qubits and can therefore be implemented as above. The total number of gates needed is thus $O(m)$ when the Hamiltonian has m terms. If there are n qubits and each term acts on c qubits, then $m \leq \binom{n}{c} = O(n^c)$, so this is definitely polynomial in n.

1.3 Non-Commuting Hamiltonians

What about if the Hamiltonian terms don't commute with each other? In that case, matters are a bit more complicated, but if the Hamiltonian is local, we can still simulate the behavior on a quantum computer.

We'd really like a formula like (19) to apply in cases where the H_i's don't commute. Let's examine the product of exponentials formula a bit more closely when M and N don't commute:

$$\begin{aligned} e^M e^N =& (I + M + \frac{1}{2}M^2 + \ldots)(I + N + \frac{1}{2}N^2 + \ldots) && (20)\\ =& I + M + N + \frac{1}{2}M^2 + MN + \frac{1}{2}N^2 + \ldots && (21)\\ e^{M+N} =& I + (M+N) + \frac{1}{2}(M+N)^2 + \ldots && (22)\\ =& I + M + N + \frac{1}{2}M^2 + \frac{1}{2}(MN + NM) + \frac{1}{2}N^2 + \ldots. && (23) \end{aligned}$$

In general, the higher order terms are quite important, but when M and N are both small, say around ϵ, then

$$e^{M+N} - e^M e^N = \frac{1}{2}(NM - MN) + O(\epsilon^3). \tag{24}$$

In particular, at short times t,

$$U(t) = e^{-it\sum_i H_i/\hbar} = \prod_i e^{itH_i/\hbar} + O(m^2 t^2 \|H_i\|^2). \tag{25}$$

Here I have put in the dependence on the number of terms m in the Hamiltonian and the norm of a single term H_i. (This is under the assumption that all terms have similar size.) This formula means that we actually can simulate the behavior of this Hamiltonian for short enough times, $t \ll 1/(m\|H_i\|)$.

For longer times, we can break the evolution up into a large number of short times. If we use a time step $\Delta t = T/N$, then

$$U(t) = \prod_{j=1}^{N} e^{-i\Delta t H/\hbar}. \tag{26}$$

For small enough Δt, each term can be expanded as a product of local terms. We therefore get

$$U(t) = \prod_{j=1}^{N} \left(\prod_i e^{i\Delta t H_i/\hbar} \right) + O(m^2 T^2 \|H_i\|^2 / N). \tag{27}$$

The result won't be a perfect emulation of the time evolution, but we can get close. We have only one power of N in the denominator of the error term because each time step gives us $O(1/N^2)$ of them and we have N time steps. Thus, by picking N large enough, we can make the error term as small as we like. This is known as the *Trotter formula* and the procedure of breaking up a time evolution using this formula is colloquially called "Trotterization."

Note that the order of the terms in this product is very important, as they are non-commuting operators. In particular, we first apply a Δt time slice for each of the H_i's in succession, then we start over again.

How complex is this algorithm? There are a total of Nm terms in the product, each of which can be implemented with a constant number of gates, and we need N $m^2 T^2 \|H_i\|^2$ to get a reasonable error. Thus, the number of gates is $O(m^3 T^2 \|H_i\|^2)$. Unlike the commuting case, we do have a dependence on the length of time we want to simulate for, but at least everything remains polynomial.

You can make various improvements to this procedure, including finding clever ways of breaking up the Hamiltonian into sums of terms which all commute with each other, using other product formulas which correct for the second-order or higher terms, and giving a more careful analysis to get a tighter bound on the performance of the actual Trotter formula. There are also various other quantum simulation algorithms that use very different techniques, some of which can be better than Trotterization in certain cases.