Quantum Computing 3/22/2022

Lecture 14: Hamiltonian Simulation

Instructor: Dieter van Melkebeek Scribe: Joon Suk Huh

Today's lecture will focus primarily on Hamiltonian simulation. Simulating the quantum Hamiltonian was one of the initial motivations behind the quantum computer. Today we show that it is possible to efficiently simulate a Hamiltonian evolution with a quantum computer. We begin by giving some physics background, and formalize the Hamiltonian simulation in the block encoding framework. We then go on to develop an efficient algorithm, which in this case happens to be optimal. Afterwards, we mention some further related results.

1 Physics Background

At any given time, we can describe the state of a quantum system by a state vector $|\psi(t)\rangle$. Interactions within the system are characterised by a Hermitian matrix H known as the Hamiltonian, whose eigenvalue we will later see correspond to the possible energy levels of the system. A quantum system will evolve in accordance with the Schrödinger equation:

$$i\hbar \frac{d\,|\psi(t)\rangle}{dt} = H\,|\psi(t)\rangle\,, \tag{1}$$

where i is the square root of -1, $\hbar$ is the reduced Planck's constant, and H is the Hamiltonian. For the rest of this lecture, we will set $\hbar = 1$, as it is just a constant real number (i.e., we can measure time in the unit of $\hbar$). In general, H could depend on t, but we restrict our attention to the cases where H is time independent.

Before solving (1), let's first consider an analogous one-dimensional equation $f'(t) = \lambda f(t)$. The solution is $f(t) = f(0)e^{\lambda t}$. One might notice that the Schrödinger equation is of this form, except replacing λ with H and f by a vector valued function, so we would expect the Schrödinger equation to be solved using a similar method. Indeed, (1)'s solution is in terms of the matrix exponential as follows:

$$|\psi(t)\rangle = U_t\,|\psi(0)\rangle \text{ where } U_t \doteq \exp(-iHt/\hbar), \tag{2}$$

where the matrix exponential is defined analogously to the usual exponential in terms of its power series expansion. We will verify later that the above is the solution of Eq. 1.

2 Matrix Functions

Consider a function $f : \mathbb{C} \to \mathbb{C}$. In the case that f is a polynomial, then $f(A)$ is well-defined for any square matrix $A \in \mathbb{C}^{N\times N}$, by replacing every instance of x^n with A^n. But in the more general case, suppose f has an absolutely convergent Taylor expansion

$$f(z) = \sum_{k=0}^{\infty} c_k z^k \tag{3}$$

which converges absolutely for every $z \in \mathbb{C}$ with $|z| < r$, where r is the radius of convergence. Then, $f(A)$ is well-defined for every square matrix $A \in \mathbb{C}^{N\times N}$ with $\|A\| < r$ for any sub-multiplicative matrix norm (i.e., $\|AB\| \leq \|A\|\|B\|$). We can show this using the absolute convergence; if $\|A\| < r$, then $\|A^k\| < r^k$, and thus

$$\left\| \sum_{k=0}^{\infty} c_k A^k \right\| \leq \sum_{k=0}^{\infty} \|c_k A^k\| \leq \sum_{k=0}^{\infty} |c_k| \|A\|^k \tag{4}$$

Since $\|A\| < r$, by the absolute convergence, the right-hand side converges, the left-hand side must converge.

The case that A has a full basis of eigenvectors can alternately be handled as follows. Say $A = V \operatorname{Diag}(\lambda_1, .., \lambda_N) V^{-1}$, and let $D \doteq \operatorname{Diag}(\lambda_1, .., \lambda_N)$. We can then derive

$$f(A) = V \operatorname{Diag}(f(\lambda_1), .., f(\lambda_N)) V^{-1}. \tag{5}$$

To see why, consider what happens when we take A^n. We have that $A^n = VDV^{-1}VDV^{-1}..VDV^{-1}$. We see that all of the V's and V^{-1}'s in the middle cancel out, leaving us $A^n = VD^nV^{-1}$, which is simply $V \operatorname{Diag}(\lambda_1^n, .., \lambda_N^n) V^{-1}$. This extends by linearity to arbitrary polynomials, and to absolutely convergent power series.

Exercise 1. *Recall the Pauli operators X, Y, and Z, and the notation $\vec{a} \cdot \vec{\sigma} \doteq a_x X + a_y Y + a_z Z$ for $\vec{a} = (a_x, a_y, a_z) \in \mathbb{C}^3$. Show that for every $\vec{a} \in \mathbb{R}^3$ with $\|\vec{a}\|_2 = 1$, and $\theta \in \mathbb{R}$*

$$\exp(i\theta\, \vec{a} \cdot \vec{\sigma}) = \cos(\theta)\, I + i \sin(\theta)\, \vec{a} \cdot \vec{\sigma}.$$

In particular,

$$\exp(i\theta X) = \begin{bmatrix} \cos\theta & i\sin(\theta) \\ i\sin\theta & \cos\theta \end{bmatrix} \quad \text{and} \quad \exp(i\theta Z) = \begin{bmatrix} e^{i\theta} & 0 \\ 0 & e^{-i\theta} \end{bmatrix}.$$

Evolution under time-independent Hamiltonian We now get back to the specific question of defining the matrix exponential. The numerical exponential has a Taylor expansion that converges absolutely everywhere:

$$\exp(z) = \sum_{k=0}^{\infty} \frac{1}{k!} z^k \text{ for every } z \in \mathbb{C}. \tag{6}$$

The $\frac{1}{k!}$ is decreasing very quickly, which is something that we use in later results. We can plug in a matrix A into this definition, to get

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k \text{ for every } A \in \mathbb{C}^{N\times N}. \tag{7}$$

Since the exponential of complex numbers converges everywhere, there are no restrictions on the norm of A. To see that the matrix exponential actually works as a solution to the Schrödinger equation, we evaluate $\frac{d}{dt} \exp(At)$. We have

$$\frac{d}{dt} \exp(At) = \frac{d}{dt} \sum_{k=0}^{\infty} \frac{1}{k!} A^k t^k \tag{8}$$

Since this series converges absolutely, we can exchange the summation and the derivative:

$$\frac{d}{dt}\exp(At) = \sum_{k=0}^{\infty} \frac{d}{dt}\frac{1}{k!}A^k t^k \tag{9}$$

By linearity of the derivative and the fact that A is constant with respect to t, the linear transformation A also commutes with $\frac{d}{dt}$, so $\frac{d}{dt}A^k t^k = A^k \frac{d}{dt}t^k = A^k k t^{k-1}$. Plugging this in:

$$\begin{aligned}\frac{d}{dt}\exp(At) &= \sum_{k=0}^{\infty} \frac{1}{k!}A^k k t^{k-1} \\ &= \sum_{k=0}^{\infty} \frac{k}{k!}AA^{k-1}t^{k-1} \\ &= \sum_{k=0}^{\infty} \frac{1}{(k-1)!}AA^{k-1}t^{k-1}\end{aligned} \tag{10}$$

From here, we can take the first A into the front of the sum, and do a reindexing $k' = k-1$, yielding

$$\frac{d}{dt}\exp(At) = A\sum_{k'=0}^{\infty} \frac{1}{k'!}A^{k'}t^{k'}, \tag{11}$$

which is simply of the form $A\exp(At)$. Thus, $\frac{d}{dt}\exp(At) = A\exp(At)$, as desired. We can verify this in the particular case of the Schrödinger equation. When we define $|\psi(t)\rangle \doteq U_t |\psi(0)\rangle$ for $U_t \doteq \exp(-iHt/\hbar)$, we have

$$\begin{aligned}i\hbar\frac{d}{dt}|\psi(t)\rangle &= i\hbar\frac{d}{dt}U_t|\psi(0)\rangle \\ &= i\hbar\cdot(-iH/\hbar)U|\psi(0)\rangle \\ &= H|\psi(t)\rangle\end{aligned} \tag{12}$$

Thus, the matrix exponential is a solution to the Schrödinger equation. From here, we can move on to the computational problem of simulating the matrix exponential.

3 Block Hamiltonian Simulation

Recall the definition of a block encoding:

Definition 1. *A block encoding of a matrix M acting on n qubits with m ancilla qubits is a unitary A acting on $m+n$ qubits such that*

$$A = \begin{bmatrix} M & * \\ * & * \end{bmatrix} \tag{13}$$

The rest of this lecture will be focused on the main algorithm described in the below theorem, about finding a block encoding of the matrix exponential:

Theorem 1. *There is a black-box algorithm that takes a block encoding of a Hermitian H with l ancilla qubits, $t \in [0,\infty)$, and $\epsilon \in (0,\infty)$, and produces a block encoding of a matrix M such that $\|M - \exp(iHt)\|_2 \leq \epsilon$, using $q = O(t + \log(1/\epsilon))$ controlled applications of the black-box and its inverse, $\tilde{O}(lq)$ other quantum gates, and $l + O(1)$ ancilla qubits.*

There are also some technical conditions:

- The block encoding of H presupposes $\|H\|_2 \leq 1$. This can be achieved by rescaling the Hamiltonian by $1/\|H\|_2$ and the time t by $\|H\|_2$. This will increase the overall complexity by a factor of $\|H\|_2$.
- The block encoding in Theorem 1 allows approximating $|\psi(t)\rangle$ to within ϵ in 2-norm, namely as $M|\psi(0)\rangle$, with probability $1-2\epsilon$ and success indicator, at some cost. The success indicator comes using the block encoding in a similar way as in the last lecture, where we need to observe $|0^l\rangle$ in the first l registers (the ancilla registers) in order for the rest of the qubits to be in state $M|\psi(0)\rangle$. The probability of this happening equals $\|M|\psi(0)\rangle\|_2^2 \geq (1-\epsilon)^2 \geq 1-2\epsilon$.

Theorem 1 solves the Hamiltonian simulation problem, and it solves that in a very strong way. In addition, the number q of applications of the black-box is optimal up to a constant factor.

In the next section, we will directly construct a less efficient Hamiltonian simulation algorithm which uses $O(t\log(t/\epsilon))$ queries, $\tilde{O}(t\log(t/\epsilon))$ other gates, and $O(\log(t)+\log\log(1/\epsilon))$ extra ancilla qubits. Further improvement to the efficiency of Theorem 1 can be achieved with Quantum Signal Processing, which will be covered in Lecture 17.

4 Algorithm

We now develop the algorithm for block Hamiltonian simulation.

4.1 Approach

Our approach makes use of the fact that if for all eigenvalues λ of H where $\|H\|_2 \leq 1$,

$$\left|\exp(i\lambda t) - \sum_{k=0}^{d} \frac{1}{k!}(i\lambda t)^k\right| \leq \epsilon \tag{14}$$

then

$$\left\|\exp(iHt) - \sum_{k=0}^{d} \frac{1}{k!}(iHt)^k\right\|_2 \leq \epsilon. \tag{15}$$

This fact is proved and used in the previous lectures in quantum walk and fast-forwarding (Lecture 12, 13).

From (15), we see that our goal is to find a block encoding of a linear combination of powers of a matrix. Powering in block encoding formalism can be done efficiently. Also, a linear combination of 'easy' unitaries can be done efficiently. Hence our apporach is first to find block encodings for each power of H, and then use the linear combination of unitaries (LCU) method to combine those block encodings into a block encoding of $\sum_{k=0}^{d} \frac{1}{k!}(iHt)^k$. Before delving into the analysis, we briefly recap on the LCU method:

- Input: A set of unitaries U_i and weights $q_i \in (0,\infty)$.
- Output: A block encoding of $\frac{1}{\sum_i q_i}\sum_i q_i U_i$.
- Algorithm: $\tilde{U}^* V \tilde{U}$ block encodes $\sum_i q_i U_i / \sum_i q_i$ where

$$V : |i\rangle|\psi\rangle \mapsto |i\rangle U_i|\psi\rangle, \quad \tilde{U} : |0^*\rangle \mapsto \frac{1}{\sqrt{\sum_i q_i}} \sum_i \sqrt{q_i}|i\rangle.$$

4.2 Approximating the Exponential Function

We first determine a small value of d that guarantees (14). Since all eigenvalues λ of the Hermitian matrix H are real, and $|\lambda| \leq \|H\|_2 \leq 1$, it suffices to ensure that

$$\forall \lambda \in [-1,1], \quad \left| \exp(i\lambda t) - \sum_{k=0}^{d} \frac{1}{k!} (i\lambda t)^k \right| \leq \epsilon. \tag{16}$$

By applying the Taylor series expansion of the exponential function, we get

$$\left| \exp(i\lambda t) - \sum_{k=0}^{d} \frac{1}{k!} (i\lambda t)^k \right| = \left| \sum_{k=d+1}^{\infty} \frac{1}{k!} (i\lambda t)^k \right|. \tag{17}$$

Using the triangle inequality, that $\lambda \leq 1$, and $t \geq 0$, we get that

$$\left| \sum_{k=d+1}^{\infty} \frac{1}{k!} (i\lambda t)^k \right| \leq \sum_{k=d+1}^{\infty} \frac{|\lambda t|^k}{k!} \leq \sum_{k=d+1}^{\infty} \frac{t^k}{k!}. \tag{18}$$

For all positive integer k,

$$\frac{k^k}{k!} \leq \sum_{l=0}^{\infty} \frac{k^l}{l!} = e^k \tag{19}$$

Therefore, $\left(\frac{k}{e}\right)^k \leq k!$ for every positive integer k, so

$$\sum_{k=d+1}^{\infty} \frac{t^k}{k!} \leq \sum_{k=d+1}^{\infty} \left(\frac{et}{k}\right)^k. \tag{20}$$

If $d \geq 2et$, we get that

$$\sum_{k=d+1}^{\infty} \left(\frac{et}{k}\right)^k \leq \sum_{k=d+1}^{\infty} \left(\frac{1}{2}\right)^k = \left(\frac{1}{2}\right)^d, \tag{21}$$

where in the last step, we used the fact that $\sum_{\ell=1}^{\infty} \left(\frac{1}{2}\right)^\ell = 1$. If $d \geq \log_2\left(\frac{1}{\epsilon}\right)$, we get

$$\left(\frac{1}{2}\right)^d \leq \epsilon. \tag{22}$$

Thus, altogether, we have that for $d \geq \max(2et, \log_2\left(\frac{1}{\epsilon}\right)) = \Theta(t + \log(1/\epsilon))$, (16) holds, and therefore

$$\| \exp(iHt) - \sum_{k=0}^{d} \frac{1}{k!} (iHt)^k \|_2 \leq \epsilon. \tag{23}$$

4.3 Powering Block Encodings

We now explain how to efficiently obtain block encodings of powers of H. We start with an elementary approach for the square.

Elementary approach Suppose that we have $(n+l)$-qubit block encoding A of an n-qubit matrix H where $A = \begin{bmatrix} H & X \\ Y & Z \end{bmatrix}$ for some X, Y, Z. We would like to perform operations on A to yield a block encoding for H^2. Notice that A^2 does not work for that purpose as

$$A^2 = \begin{bmatrix} H & X \\ Y & Z \end{bmatrix} \begin{bmatrix} H & X \\ Y & Z \end{bmatrix} = \begin{bmatrix} H^2 + XY & * \\ * & * \end{bmatrix},$$

and we have $H^2 + XY$ in the top left sector rather than the H^2 we want. Instead, consider $(I \otimes A)(A \otimes I)$:

$$(I \otimes A)(A \otimes I) = \begin{bmatrix} H & X & 0 & 0 \\ Y & Z & 0 & 0 \\ 0 & 0 & H & X \\ 0 & 0 & Y & Z \end{bmatrix} \begin{bmatrix} H & 0 & X & 0 \\ 0 & H & 0 & X \\ Y & 0 & Z & 0 \\ 0 & Y & 0 & Z \end{bmatrix} = \begin{bmatrix} H^2 & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}. \tag{24}$$

Thus, $(I \otimes A)(A \otimes I)$ is a block encoding for H^2 that uses $2l$ ancillas. Generally, the natural extension of this technique can be used to produce a block encoding of H^d for any d, and requires $O(d)$ queries to the original block encoding, A, and $O(dl)$ ancillas qubits. $O(dl)$ ancillas qubits is very costly, so it is important to find a method that requires fewer ancillas. A method that accomplishes this is explored next.

Improved approach We can make use of the results on quantum walks to block encode the d-th power of H using just $l + O(\log d)$ ancillas and still only need $O(d)$ queries to the block encoding for H. The method requires a block encoding $\tilde{A}$ of H such that $\tilde{A}$ is its own inverse ($\tilde{A}$ is an "involution"). If we can find such $\tilde{A}$, then we can make use of the following Lemma proved in the previous lecture:

Lemma 2. *Let T_k represent the k-th Chebyshev polynomial of the first kind. If $\tilde{A}$ is a block encoding of H with l ancilla qubits such that $\tilde{A}^2 = I$ then $(\tilde{A}R)^k$ is a block encoding of $T_k(H)$ where R denotes reflection around $|0^l *\rangle$.*

Remember that $H^k = \mathbb{E}_s[T_s(H)]$ where $s \doteq \sum_{j=1}^{k} X_j$ and X_j's are independent Rademacher random variables (equal probability of being 1 and -1). With this, one can see how Lemma 2 is useful in getting a block encoding of powers in conjunction with the LCU method.

Constructing a self-inverse block encoding Given a block encoding A of H, we can construct a self-inverse block encoding as follows. First consider

$$A' \doteq \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \tag{25}$$

which acts on one more qubit than A does. Note that A' is its own inverse because

$$\begin{aligned} (A')^2 &= \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \\ &= \begin{bmatrix} AA^* & 0 \\ 0 & A^*A \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = I \end{aligned} \tag{26}$$

where we used the fact that $AA^* = I = A^*A$ because A is unitary by virtue of being a block encoding for H.

Consider the action of A' on the state $|+\rangle |0^m\rangle |\psi\rangle$.

$$\begin{aligned} A' |+\rangle |0^m\rangle |\psi\rangle =& \frac{1}{\sqrt{2}} \begin{bmatrix} A |0^m\rangle |\psi\rangle \\ A^* |0^m\rangle |\psi\rangle \end{bmatrix} \\ =& \begin{bmatrix} |0^m\rangle H |\psi\rangle + |g\rangle \\ |0^m\rangle H^* |\psi\rangle + |g'\rangle \end{bmatrix} \\ =& |+\rangle |0^m\rangle H |\psi\rangle + |\tilde{g}\rangle \end{aligned} \tag{27}$$

where

$$\tilde{g} = \begin{bmatrix} |g\rangle \\ |g'\rangle \end{bmatrix}. \tag{28}$$

Note that $|g\rangle \perp |0^m *\rangle$ and $|g'\rangle \perp |0^m *\rangle$ so $|\tilde{g}\rangle \perp |+\rangle |0^m *\rangle$.

Now, by applying a change of basis using a unitary $\tilde{U}$ such that $\tilde{U} |0\rangle |\phi\rangle = |+\rangle |\phi\rangle$, we can construct $\tilde{A} \doteq \tilde{U}^* A' \tilde{U}$ with the following properties:

- $\tilde{A}^2 = I$.
- $\tilde{A}$ block encodes H.
- $\tilde{A}$ uses one more ancilla qubit than A.

4.4 Linear Combination of Unitaries (LCU)

In order to obtain a block encoding for H^k, we can write H^k as a linear combination of $T_j(H)$ for $j \leq k$, and apply the LCU method. However, as we are ultimately interested in the linear combination $M \doteq \sum_{k=0}^{d} \frac{1}{k!} (iHt)^k$ of those powers, we write M as a linear combination $\sum_{k=0}^{d} c_k T_k(H)$ of $T_k(H)$ for $k \leq d$, and directly apply the LCU method to get a block encoding of M.

It was shown in the previous lecture that $H^k = \mathbb{E}_s[T_s(H)]$ where $s \doteq \sum_{j=1}^{k} X_j$ and the X_j's are independent Rademacher random variables (equal probabilities of being -1 and $+1$). As shown above, we can use a self-inverse block encoding $\tilde{A}$ for H to compute a block encoding of $T_k(H)$ as $(\tilde{A}R)^k$. Put together, this means that the top left corner of

$$\sum_{k=0}^{d} c_k (\tilde{A}R)^k \tag{29}$$

equals M. Thus, it suffices to find a block encoding for (29) to obtain a block encoding of M. We can do so using the LCU method from the previous lecture. The LCU method actually gives us a block encoding of $M/(\sum_{k=0}^{d} |c_k|)$. Note that

$$\sum_{k=0}^{d} |c_k| \leq \sum_{k=0}^{d} \frac{1}{k!} t^k \to \exp(t) \quad \text{as } d \to \infty,$$

as $M \doteq \sum_{k=0}^{d} \frac{1}{k!} (iHt)^k = \sum_{k=0}^{d} c_k H^k$ where $c_k = \frac{(it)^k}{k!}$. The resulting algorithm has the following properties:

- Uses $O(d)$ queries and $\tilde{O}(d)$ other quantum gates.
- Uses $O(\log(d))$ additional ancillas.
- Has success probability of about e^{-t}.

Note that the success probability quickly goes to 0 for large t. That being the case, we'll use this approach for $t = 1$, and obtain the result for times larger than $t = 1$ by computing the t-th power of the one for $t = 1$.

4.5 Complete Algorithm

Our complete Hamiltonian simulation algorithm is as follows. First, use the linear combination of unitaries approach described above for $t = 1$ and with error bound $\frac{\epsilon}{t}$. We choose error bound $\frac{\epsilon}{t}$ because we will later raise our block encoding from the linear combination of unitaries approach to the power t, so the total approximation error will be less than or equal to $t\frac{\epsilon}{t} = \epsilon$. Our simulation for $t = 1$ has the following properties:

- Uses $O(\log(\frac{t}{\epsilon}))$ queries and $\tilde{O}(\log(\frac{t}{\epsilon}))$ other quantum gates.
- Uses $O(\log\log(\frac{t}{\epsilon}))$ additional ancilla qubits.
- Has success probability of about $e^{-1} \simeq 0.37$.

Now, since $M \doteq \sum_{k=0}^{d} \frac{1}{k!}(iHt)^k$ is (almost) unitary, we can use (robust) oblivious amplification to boost the success probability to (almost) one. Then, we use the improved powering technique for block encodings discussed above with exponent t to obtain the desired block encoding for simulating through time t. Our full simulation algorithm has the following properties:

- Uses $O(q)$ queries and $\tilde{O}(q)$ other quantum gates for $q = O(t\log(\frac{t}{\epsilon}))$.
- Uses $O(\log(t) + \log\log(\frac{1}{\epsilon}))$ additional ancilla qubits.
- Has success probability close to 1.

Further improvement to $q = O(t + \log(\frac{1}{\epsilon}))$ and $O(1)$ additional ancillas can be achieved with Quantum Signal Processing, which will be covered in Lecture 17.

5 Local and Sparse Hamiltonians

In most physical applications, we are interested in Hamiltonians that have additional properties. In particular, for most physical applications we are interested in *local* Hamiltonians.

Definition 2. *A Hamiltonian H is k-local if $H = \sum_{j=1}^{m} H_j$ where each H_j is a Hamiltonian acting on at most k qubits.*

Hamiltonians like this are common in physics, where Hamiltonians describe interactions between components of the system. One part of the Hamiltonian could describe the interaction between a small number of components. Often times one can set $k = 2$ as interactions typically only happen between pairs of components.

We also consider a further generalization which is of interest for algorithmic applications, including solving systems of sparse linear equations, which we will cover next lecture:

Definition 3. *A Hamiltonian H is s-sparse if each row and column of H contains at most s nonzero entries.*

One can show that if H acts on at most k qubits, then H is s-sparse for $s = 2^k$, and thus if we can handle sparse Hamiltonians then we can handle local Hamiltonians. The most interesting algorithmic applications are for $s = O(1)$ or $s = \text{poly}\log(N)$

Simulation of sparse Hamiltonians We will see later an efficient construction of block encoding for $H/(s\|H\|_{\max})$ where $\|H\|_{\max} \doteq \max_{i,j} |H_{i,j}|$. In this case, to simulate H for t steps, simulate $H' \doteq H/(s\|H\|_{\max})$ for $t' = ts\|H\|_{\max}$ steps. The running time becomes $O(q)$ controlled applications of black box encoding of H and $\tilde{O}(q)$ other quantum gates, where $q = O(st\|H\|_{\max} + \log(1/\epsilon))$. Next lecture, this is the setting in which we will be using Hamiltonian simulation.

Finding ground states Apart from Hamiltonian simulation, another computational problem about physical quantum systems that is of central importance, is finding ground states. Eigenstates of a Hamiltonian represent stable states of a physical system, the corresponding eigenvalues represent energy levels of those states, and the ground state is the eigenstate of the Hamiltonian with lowest energy.

In contrast to Hamiltonian simulation, we do not know of an efficient quantum algorithm for this problem. It is believed that a quantum computer cannot find the ground state of an arbitrary physical system in polynomial time. In particular, the simplified problem of deciding whether the ground state either has energy at most some threshold a, or energy at least some other threshold b, for a and b sufficiently separated, has been shown to be complete for a quantum version of NP known as QMA (Quantum Merlin Arthur). It is conjectured that not all of QMA can be solved by a quantum computer in polynomial time, in which case QMA-complete problems cannot, but the conjecture remains open.

6 Other Results

The method for Hamiltonian simulation discussed in this lecture is relatively recent. An older method is based on formulas known as Lie-Trotter-Suzuki decompositions or as product formulas. Lie-Trotter-Suzuki decompositions only work for local Hamiltonians, so they are not suitable for sparse Hamiltonians. Initial analysis bounded the number of queries and quantum gates needed for Lie-Trotter-Suzuki decompositions at $O(q)$ queries and $\tilde{O}(q)$ other quantum gates with $q = O(t^2 + \frac{1}{\epsilon})$ rather than the $q = O(t + \log(\frac{1}{\epsilon}))$ achieved with Quantum Signal Processing. However, recently, more careful analysis as well as experiments show that Lie-Trotter-Suzuki decompositions may actually perform better than Quantum Signal Processing for simulating local Hamiltonians in practice (and perhaps also in theory).

The bounds on queries, quantum gates, and ancilla qubits achieved by the Quantum Signal Processing approach to Hamiltonian simulation are theoretically optimal in a blackbox setting. However, better results can be achieved for special types of Hamiltonians.

The technique we studied in this lecture applies to time-independent Hamiltonians. Simulating time-dependent Hamiltonians is a related problem which has been studied as well. For a recent algorithm for time-dependent Hamiltonian simulations, please refer to [BCS+20] and references therein.

For a general reference to the block encoding and Hamiltonian simulation please refer to [CGJ19, GSLW19].

References

[BCS+20] Dominic W Berry, Andrew M Childs, Yuan Su, Xin Wang, and Nathan Wiebe. Time-dependent Hamiltonian Simulation with L^1-norm Scaling. *Quantum*, 4:254, 2020.

[CGJ19] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 2019.

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum Singular Value Transformation and Beyond: Exponential Improvements for Quantum Matrix Arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.