Quantum Algorithms 3/9/2023

Lecture 14: Hamiltonian Simulation

Instructor: Dieter van Melkebeek

This lecture focuses 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.

We start by finishing the discussion on the LCU method from last lecture. The method will be an important ingredient in Hamiltonian simulation, as well.

1 Linear Combination of Unitaries

Let us first present a solution to Exercise #10, where we consider the special case of the sum of two n-qubit unitaries U_1 and U_2. The goal is to construct a block encoding for $\frac{1}{q}(U_1 + U_2)$ for q as large as possible and with as few ancillas as possible, given access to a unitary U on $n+1$ such that $U|b\rangle|\psi\rangle = |b\rangle U_{b+1}|\psi\rangle$.

Note that we cannot hope to do better than $q = 2$. This is because in the case where $U_1 = U_2$, the sum $U_1 + U_2$ has two-norm 2 and thus needs to be reduced by a factor at least 2 in order to appear as the top left corner of another unitary matrix.

The given unitary is U equals

$$\begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix}.$$

We'd like to transform it into a matrix that, up to a scalar, looks like

$$\begin{bmatrix} U_1 + U_2 & * \\ * & * \end{bmatrix}.$$

Adding the top right block to the top left block and the bottom left blocks to the top left block achieves the goal:

$$\begin{bmatrix} I & I \\ * & * \end{bmatrix} \begin{bmatrix} U_0 & 0 \\ 0 & U_1 \end{bmatrix} \begin{bmatrix} I & * \\ I & * \end{bmatrix} = \begin{bmatrix} U_0 + U_1 & * \\ * & * \end{bmatrix}.$$

These partially specified operations cannot be extended to unitary operations, but they can if we scale both with a factor of $1/\sqrt{2}$. A natural extension for both is the Hadamard operator at the block level, which yields the desired block encoding with $q = 2$:

$$(H \otimes I)U(H \otimes I) = \begin{bmatrix} \frac{1}{2}(U_0 + U_1) & * \\ * & * \end{bmatrix}.$$

The block encoding uses no additional ancillas (beyond the one for U).

This completes the solution for the special case of the sum of two unitaries. We can generalize the construction for the general case of a linear combination $\sum_{i=1}^{k} q_i U_i$, where $U_1, \ldots, U_k$ are unitaries

on n qubits, and $q_i \in \mathbb{C}$ for $i \in [k]$. Like in the special case, we assume a uniform way of accessing the unitaries in the form of a *selector operator* U on $\ell = \log k$ additional qubits that maps $|i\rangle |\psi\rangle$ to $|i\rangle U_i |\psi\rangle$.

For simplicity, let us assume that $q_i \in (0, \infty)$. This situation can be obtained by dropping the terms with $q_i = 0$ and incorporating the phases of each nonzero q_i into the corresponding U_i. Because of the case of identical U_i's, as before, the best we can hope for is to block encode $\frac{1}{q}\sum_{i=1}^k q_i U_i$, where $q = \sum_{i=1}^k q_i$.

Consider replacing the first block row in U by a linear combination of the block rows with a coefficient vector $|\alpha\rangle \doteq \sum_{i=1}^k \alpha_i |i\rangle$ of 2-norm 1, and similarly the block columns and a coefficient vector $|\beta\rangle \doteq \sum_{i=1}^k \beta_i |i\rangle$. We have that

$$(\langle\alpha| \otimes I)U(|\beta\rangle \otimes I) = (\langle\alpha| \otimes I) \begin{bmatrix} U_1 & 0 & \dots \\ 0 & U_2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} (|\beta\rangle \otimes I) = \sum_i \overline{\alpha_i}\beta_i U_i$$

and want $\overline{\alpha_i}\beta_i = q_i/c$ for some scalar c that is as small as possible, ideally $c = q$. Note that $\langle\alpha|\beta\rangle = \frac{q}{c}$, so the smallest c is obtained when the inner product $\langle\alpha|\beta\rangle$ is largest. This happens when $|\alpha\rangle$ and $|\beta\rangle$ are parallel. Thus, we set $\alpha_i = \beta_i = \sqrt{\frac{q_i}{q}}$ and this way achieve $c = q$. Note that the square roots are well-defined as we made sure that $q_i \in (0, \infty)$. Instead of incorporating the phases of the q_i into the corresponding unitaries U_i, we can leave them as are and define $\sqrt{\exp(i\theta)r}$ for $\theta \in [0, frm{-}e\pi)$ and $r \in (0, \infty)$ as $\exp(i\theta/2)\sqrt{r}$ (where in the last two expressions. i denotes the imaginary unit rather than an index).

The resulting block encoding uses $\ell = \log k$ ancillas. To state the general result, in addition to the selection operator U, we assume access to a *preparation operator* V on ℓ qubits with the property that $V|0^\ell\rangle = \sum_{i=1}^k \sqrt{\frac{q_i}{q}} |i\rangle$, where $q = \sum_{i=1}^k |q_i|$. The block encoding can then be written as

$$(V^* \otimes I)U(V \otimes I). \tag{1}$$

Lemma 1 (LCU method). *Let $U_1, \dots, U_k$ be unitaries on n qubits and $q_1, \dots, q_j \in \mathbb{C}$. Let U be a unitary on $\ell = \log k$ additional qubits such that $U|i\rangle|\psi\rangle = |i\rangle U_i|\psi\rangle$ for $i \in [k]$, and V a unitary on ℓ qubits such that $V|0^\ell\rangle = \sum_{i=1}^k \sqrt{\frac{q_i}{q}}|i\rangle$, where $q \doteq \sum_{i=1}^k |q_i|$. There exists a block encoding of $\sum_{i=1}^k$ for $\frac{1}{q}\sum_{i=1}^k q_i U_i$ using ℓ ancillas that consists of one application of U, V, and V^*.*

The block encoding can then be written as

$$(V^* \otimes I)U(V \otimes I). \tag{2}$$

Note that if all U_i are Hermitian, then so is U as well as the block encoding (2). As we have already seen and will see more of later on, Hermitian block encodings have interesting properties and often simplify constructions. As we're working with unitary operators, being Hermitian is equivalent to being a reflection. This is because unitary operators only have eigenvalues of absolute value 1, Hermitian ones only have real eigenvalues, and reflections only have eigenvalues ± 1.

2 A Little Bit of 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:

$$i\hbar \frac{d\,|\psi(t)\rangle}{dt} = H\,|\psi(t)\rangle\,, \tag{3}$$

where i denotes the imaginary unit, $\hbar$ 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 (3), 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, (3)'s solution is in terms of the matrix exponential as follows:

$$|\psi(t)\rangle = U_t\,|\psi(0)\rangle \text{ where } U_t \doteq \exp(-iHt/\hbar), \tag{4}$$

where the matrix exponential can be defined in terms of the usual exponential. We first explain how this can be done and then verify later that (4) is indeed the solution of (3).

3 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. In the more general case, suppose f has a Taylor expansion

$$f(z) = \sum_{k=0}^{\infty} c_k z^k \tag{5}$$

that 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., $\|AB\| \le \|A\|\|B\|$). We can show this using the absolute convergence; if $\|A\| < r$, then $\|A^k\| < r^k$, and thus

$$\left\| \sum_{k=\ell}^{\infty} c_k A^k \right\| \le \sum_{k=\ell}^{\infty} \|c_k A^k\| \le \sum_{k=\ell}^{\infty} |c_k| \|A\|^k \tag{6}$$

Since $\|A\| < r$, by the absolute convergence, the right-hand side converges to 0 for $\ell \to \infty$ and so does the left-hand side.

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

$$f(A) = V\,\mathrm{Diag}(f(\lambda_1), .., f(\lambda_N))V^{-1}. \tag{7}$$

As the matrices that we are interested in all can be diagonalized, this is the definition we will adopt. To see that it is consistent to the above definitions in case where both apply, consider what

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

By our definition of matrix functions, the following approximation property from last lecture immediately carries over from polynomials to arbitrary functions.

Fact 2. *Suppose that $M \in \mathbb{C}^{N\times N}$ has a full orthonormal basis of eigenvectors, and $f, g : \mathbb{C} \to \mathbb{C}$. If $|f(\lambda) - g(\lambda)| \le \epsilon$ for every eigenvalue λ of M, then $\|f(M) - g(M)\|_2 \le \epsilon$.*

Exercise. Recall the Pauli operators X, Y, and Z, and the notation $\vec{a}\cdot\vec{\sigma} \doteq a_xX + a_yY + a_zZ$ 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,

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

Evolution operator under time-independent Hamiltonian We now get back to the specific question of defining the matrix exponential. As an aside, note that scalar exponential has a Taylor expansion that converges absolutely everywhere:

$$\exp(z) = \sum_{k=0}^{\infty} \frac{1}{k!} z^k \text{ for every } z \in \mathbb{C}. \tag{8}$$

The $\frac{1}{k!}$ is decreasing very quickly, which is something that we use in later results. Thus, we can plug in any matrix A into this series and get

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k \text{ for every } A \in \mathbb{C}^{N\times N}. \tag{9}$$

In the case of the evolution operator $U \doteq \exp(-iHt/\hbar)$, the matrix H is Hermitian and therefore has a full orthonormal basis of eigenvectors: $H = V\,\mathrm{Diag}(\lambda)V^*$ where $V^*V = I$. Here, we use $\mathrm{Diag}(f(\lambda))$ as a shorthand for $\mathrm{Diag}(f(\lambda_1), \dots, f(\lambda_N))$. By definition, $U = V\,\mathrm{Diag}(\exp(-i\lambda t/\hbar)V^*$.

Note that U is unitary. This has to be the case as U maps pure states to pure states, but can also be seen as follows:

$$\begin{aligned} U^*U &= (V\,\mathrm{Diag}(\exp(i\lambda t/\hbar)V^*)(V\,\mathrm{Diag}(\exp(-i\lambda t/\hbar)V^*) \\ &= V\,\mathrm{Diag}\left(\exp(i\lambda t/\hbar)\cdot\exp(-i\lambda t/\hbar)\right)V^* = I. \end{aligned}$$

To argue that $U|\psi(0)\rangle$ solves (3), we need to show that $i\hbar\frac{d}{dt}(U|\psi(0)\rangle) = H(U|\psi(0)\rangle)$. This

follows because

$$\begin{aligned}\frac{d}{dt}(U) &= V\,\mathrm{Diag}(\frac{d}{dt}(\exp(-i\lambda t/\hbar))V^* \\ &= V\,\mathrm{Diag}(\frac{-i\lambda}{\hbar}\cdot\exp(-i\lambda t/\hbar)V^* \\ &= \frac{-i}{\hbar}V\,\mathrm{Diag}(\lambda)\,\mathrm{Diag}(\exp(-i\lambda t/\hbar)V^* \\ &= \frac{-i}{\hbar}(V\,\mathrm{Diag}(\lambda)V^*)(V\,\mathrm{Diag}(\exp(-i\lambda t/\hbar)V^*) \\ &= \frac{-i}{\hbar}HU.\end{aligned}$$

Thus, the evolution operator U solves the Schrödinger equation. From here, we can move on to the problem of computing U, which we will do in the block encoding framework.

4 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*

$$A = \begin{bmatrix} M & * \\ * & * \end{bmatrix} \tag{10}$$

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 3. *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, $\tilde{O}(q\ell)$ other quantum gates, and $\ell + O(1)$ ancilla qubits.*

Note that 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 $\|H\|_2$.

Theorem 3 solves the Hamiltonian simulation problem, and it solves that in a very strong way, namely in the block encoding framework. In addition, it can be shown that the number q of applications of the black-box is optimal up to a constant factor.

The block encoding in Theorem 3 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. The success indicator comes using the block encoding in a similar way as in the last lecture, where we need to observe $|0^\ell\rangle$ 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 $\|M|\psi(0)\rangle\|_2^2 \geq (1-\epsilon)^2 \geq 1-2\epsilon$.

In the next section, we will directly construct a less efficient Hamiltonian simulation algorithm which uses $q = O(t\log(t/\epsilon))$ queries, $\tilde{O}(q)$ other gates, and $O(\log q)$ extra ancilla qubits. Further improvement to the efficiency of Theorem 3 can be achieved with Quantum Signal Processing, which will be covered in a later lecture.

5 Algorithm

We now develop the algorithm for block Hamiltonian simulation.

5.1 Approach

Our approach makes use of a polynomial approximation of the exponential function obtained by truncating the Taylor expansion. We instantiate Fact 2 with $f(z) = \exp(z)$ and $g(z) = \sum_{k=0}^{d} \frac{1}{k!} z^k$ the Taylor series of $\exp(z)$ truncated after degree d: If for all eigenvalues λ of H,

$$\left| \exp(i\lambda t) - \sum_{k=0}^{d} \frac{1}{k!} (i\lambda t)^k \right| \leq \epsilon \tag{11}$$

then

$$\left\| \exp(iHt) - \sum_{k=0}^{d} \frac{1}{k!} (iHt)^k \right\|_2 \leq \epsilon. \tag{12}$$

From (12), we see that our goal is to find a block encoding of a linear combination of powers of a matrix. Powering in the block encoding formalism can be done efficiently. Also, a linear combination of easy unitaries can be done efficiently using the LCU method (at the cost of some additional ancillas and a possible reduction in scalar). Hence our apporach is first to find block encodings for each power of H, and then use the LCU method to combine those block encodings into a block encoding of $g(iHT) = \sum_{k=0}^{d} \frac{1}{k!} (iHt)^k$.

5.2 Approximating the exponential function

We first determine a small value of d that guarantees (11). Since all eigenvalues λ of the Hermitian matrix H are real, and $|\lambda| \leq \|H\|_2 \leq 1$, it suffices to ensure that

$$\forall \lambda \in [-1, 1], \quad \left| \exp(i\lambda t) - \sum_{k=0}^{d} \frac{1}{k!} (i\lambda t)^k \right| \leq \epsilon. \tag{13}$$

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

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

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

$$\left| \sum_{k=d+1}^{\infty} \frac{1}{k!} (i\lambda t)^k \right| \leq \sum_{k=d+1}^{\infty} \frac{|\lambda t|^k}{k!} \leq \sum_{k=d+1}^{\infty} \frac{t^k}{k!}. \tag{15}$$

For all positive integer k,

$$\frac{k^k}{k!} \leq \sum_{\ell=0}^{\infty} \frac{k^l}{\ell!} = e^k \tag{16}$$

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

$$\sum_{k=d+1}^{\infty} \frac{t^k}{k!} \leq \sum_{k=d+1}^{\infty} \left(\frac{et}{k}\right)^k. \tag{17}$$

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

$$\sum_{k=d+1}^{\infty} \left(\frac{et}{k}\right)^k \leq \sum_{k=d+1}^{\infty} \left(\frac{1}{2}\right)^k = \left(\frac{1}{2}\right)^d, \tag{18}$$

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

$$\left(\frac{1}{2}\right)^d \leq \epsilon. \tag{19}$$

Thus, altogether, we have that for $d \geq \max(2et, \log_2\left(\frac{1}{\epsilon}\right)) = \Theta(t + \log(1/\epsilon))$, (13) holds, and therefore

$$\| \exp(iHt) - \sum_{k=0}^{d} \frac{1}{k!}(iHt)^k \|_2 \leq \epsilon. \tag{20}$$

5.3 Powering in the block encoding framework

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 $(n+\ell)$-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 $(I \otimes A)(A \otimes I)$:

$$(I \otimes A)(A \otimes I) = \begin{bmatrix} H & X & 0 & 0 \\ Y & Z & 0 & 0 \\ 0 & 0 & H & X \\ 0 & 0 & Y & Z \end{bmatrix} \begin{bmatrix} H & 0 & X & 0 \\ 0 & H & 0 & X \\ Y & 0 & Z & 0 \\ 0 & Y & 0 & Z \end{bmatrix} = \begin{bmatrix} H^2 & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}. \tag{21}$$

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^k for any k, and requires $O(k)$ queries to the original block encoding, A, and $O(k\ell)$ ancillas qubits. $O(k\ell)$ 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 k-th power of H using just $\ell + O(\log k)$ ancillas and still only need $O(k)$ queries to the block encoding for H. The method requires a Hermitian block encoding R of H, or equivalently, a block encoding R of H that is a reflection. If we can find such R, then we can make use of the following lemma proved in the previous lecture:

Lemma 4. *If R is a Hermitian block encoding of a matrix M, then $(RR_{|0^\ell *\rangle})^k$ block encodes $T_k(M)$ with the same number of ancillas for every $k \in \mathbb{N}$, where T_k denotes the Chebyshev polynomial of degree k.*

We next explain how to obtain a Hermitian block encoding.

Constructing a Hermitian block encoding of a Hermitian matrix. Given a block encoding A of H, we can construct a Hermitian block encoding as follows. First consider

$$A' \doteq \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}, \tag{22}$$

which acts on one more qubit than A does. Note that A' is unitary (because A is) and Hermitian (by construction). By our analysis of the solution to Exercise #10,

$$R \doteq (Had \otimes I)A'(Had \otimes I)$$

block encodes $\frac{1}{2}(A + A^*)$, where we write Had for the Hadamard gate in order to avoid confusion with the Hermitian H that we are considering. As A block encodes H, A^* block encodes H^*. Since H is Hermitian, this means that $\frac{1}{2}(A + A^*)$ block encodes H, and therefore so does R.

Note that the Hermitian block encoding R uses one more ancilla than the original block encoding A, and that R involves a controlled A and a controlled A^* operation.

5.4 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

$$M \doteq \sum_{k=0}^{d} \frac{1}{k!}(iHt)^k = sum_{k=0}^{d} \frac{(it)^k}{k!} H^k \tag{23}$$

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 $H^k = \mathbb{E}_{s_k}[T_{s_k}(H)]$ where s_k is the sum of k independent uniform ± 1 values. This means that the coefficient $\frac{(it)^k}{k!}$ on the right-hand side of (23) gets distributed over the coefficients $c_0, \ldots, c_d$. It follows that

$$\sum_{k=0}^{d} |c_k| \leq \sum_{k=0}^{d} |\frac{(it)^k}{k!}| = \sum_{k=0}^{d} \frac{t^k}{k!}. \tag{24}$$

As shown above, we can use a Hermitian block encoding R for H to compute a block encoding of $T_k(H)$ as $(RR_{|0^\ell *\rangle})^k$. Put together, this means that the top left corner of

$$\sum_{k=0}^{d} c_k (RR_{|0^\ell *\rangle})^k \tag{25}$$

equals M. Thus, it suffices to find a block encoding for (25) to obtain a block encoding of M. We use the LCU method, which actually gives us a block encoding of M/q where $q = \sum_{k=0}^{d} |c_k|$. As M is close to unitary, the success probability of the block encoding is about $1/q^2$. Since the right-hand side of (24) converges to e^t for $d \to \infty$, we can guarantee that the success probability is at least about e^{-2t}.

In summary, the resulting algorithm has the following characteristics:

- $O(d)$ queries and $\tilde{O}(d)$ other quantum gates.
- $O(\log(d))$ additional ancillas.
- Success probability of about e^{-2t}.

Note that the success guarantee 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 $t = 1$.

5.5 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 $t \cdot \frac{\epsilon}{t} = \epsilon$. Our simulation for $t = 1$ has the following characteristics:

- $O(\log(\frac{t}{\epsilon}))$ queries and $\tilde{O}(\log(\frac{t}{\epsilon}))$ other quantum gates.
- $O(\log\log(\frac{t}{\epsilon}))$ additional ancilla qubits.
- Success probability of at least e^{-2}.

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 characteristics:

- $q = O(t \log(\frac{t}{\epsilon}))$ queries and $\tilde{O}(q)$ other quantum gates.
- $O(\log q)$ additional ancilla qubits.
- 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 a later lecture.

6 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.

7 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.