Techniques in Hamiltonian Simulation: an Unofficial Review

Adittya Patil
Department of Physics
The University of Texas at Austin
Austin, TX 78712, USA

October 23, 2023

Abstract

Hamiltonian simulation is a problem which investigates the most efficient methods to model quantum systems on quantum computers. As there are several different types of ways to model the states and behaviors of these systems, different techniques arise from a singular formula in order to approximate these hamiltonians to varying degrees of complexity. In this work, we will build a foundation for the basics of how to simulate hamiltonians efficiently to varying degrees and build off of the basics to simulate them by looking at examples.

1 Introduction

Computers provide some of their biggest use cases in their work in modeling physical systems, which has a wide range of applications in the sciences and engineering. With the rise in quantum computing in the industry, they are slowly finding their place alongside their classical counterparts. One of the most common use cases for quantum computers is modeling quantum systems. Here, the quantum system we consider in most cases is a group of n interacting particles. With this system, the hamiltonian is given by a matrix of 2^n x 2^n complex entries, which is known as a unitary matrix.

The reason simulating quantum systems is such a useful task for quantum computers is because of how the memory requirement would scale for simulating a quantum system using a classical computer. For each new particle in the interacting system, the statevector, which contains 2^n entries, scales exponentially. Hence, when simulating a hamiltonian which evolves with time, the problem becomes intractable. In addition, there are hamiltonians representing specific quantum systems, such as Ising and Hubbard models, for example. These two particular types of systems have various applications in quantum electrodynamics and quantum chromodynamics. This reveals that these quantum mechanical systems are best modeled by other quantum mechanical systems.

However, in order for these techniques to be accurately represented on a quantum computer, they need to be simulated. In this case, the "thing" which we are trying to simulate is the hamiltonian matrix, which, in tandem with Schrodinger's equation, governs the dynamics of the quantum system to be modeled. However, these matrices are usually difficult to exponentiate.

This brings us to the fundamental point of Quantum Simulation, which is to use techniques to take a difficult-to-exponentiate hamiltonian, and make it easy for a gate-based quantum computer using a universal gate set to model. Our aim in this work is to investigate such techniques, and see how they can be applied to some example Hamiltonians.

2 Definitions

Before we get to defining what exactly a Hamiltonian is, it is important that we look at some of the mathematical constructs which we use to look at quantum systems in general. From this, we can properly deduce the important role which hamiltonians play in governing the dynamics of such systems.

Hilbert Space A hilbert space is a linear vector space with a few special properties. Most notably, that it supports complex vectors which have inner products. It is the vector space which is used to represent quantum states and where unitary transformations occur inside of. Hence, the dynamics of quantum systems take place in this space.

Hermitian Matrix A matrix is said to be hermitian if it holds the property that it is equal to its complex conjugate. In order for this to hold true, all Hermitian matrices must be square and contain complex entries.

Unitary Matrix A matrix is said to be unitary if it holds the property that it its complex conjugate is equal to its inverse. In order for a matrix to hold this property, it has to meet several criteria. it must be square, nonsingular, invertible and diagonalizable. If two unitary matrices are added, subtracted, or multiplied, the resultant matrix is also unitary. However, their use in quantum mechanics comes when they act on quantum states (i.e. vectors in a designated hilbert space). They are $L2$-norm preserving, which means that it preserves the inner product of the state.

Hamiltonian Generally speaking, the hamiltonian is an operator which describes the energy configuration of any particular quantum system. They can also be described as instantaneous time generators of unitary transformations. What this means is that they're things that give rise to unitary transformations when you "leave them running" for some period of time [1]. Hamiltonians are described by hermitian matrices because hermiticity guarantees that the energy spectrum is real and that time evolution is unitary (probability-preserving) [2].

3 Baker-Campbell-Hausdorf and Trotter Formula

In this work, we will be looking at a system of n particles, which is described using a 2^n vector, containing complex entries. This means that the Hamiltonian describing the system would be governed by a matrix with 2^n x 2^n complex entries.

3.1 Introduction to the BCH Formula

The hamiltonian is an operator which acts on the system in such a way that it acts on the system in a way such that it can be represented as a series of interactions which individually act on a subset of the particles. Mathematically, this can be expressed as:

$$\hat{H} = \hat{H}_1 + \hat{H}_2 + ... + \hat{H}_k \tag{1}$$

It is important to consider the commutativity that each of these terms with each other. This is because it will simplify how to represent the Hamiltonian appropriately when it comes to simulating it on a quantum computer. This can be represented using the unitary time evolution operator, which takes the form of Equation (2) and is applied in the following way in Equation (3)

$$e^{-i\hat{H}t} \tag{2}$$

$$|\Psi(t)\rangle = e^{-i\hat{H}t} |\Psi(0)\rangle \tag{3}$$

The commutation relations affect the way in which we can represent the hamiltonian when applying it to the unitary time evolution operator. We can decompose the hamiltonian as demonstrated in equation (1) in the following manner:

$$e^{-i\hat{H}t} = e^{-i(\hat{H}_1+\hat{H}_2+...\hat{H}_k)t} \tag{4}$$

The reason why considering commutativity is so important is because these smaller hamiltonians are not always closed under commutation. In the case which they were, we can express the right hand side of equation (4) in the following manner:

$$e^{-i(\hat{H}_1+\hat{H}_2+...\hat{H}_k)t} = e^{-i\hat{H}_1 t}e^{-i\hat{H}_2 t}...e^{-i\hat{H}_k t} \tag{5}$$

If the hamiltonians do not commute with each other, then we have to employ the Baker-Campbell-Hausdorff (BCH) formula. However, before implementing the BCH formula, we need to look at the following Taylor Expansion:

$$e^{aM} = [\sum_{n=1}^{\infty} \frac{a^n}{n!} M^n] = I + \frac{a^2}{2} M^2 + ... \tag{6}$$

where M is an operator (in this case a matrix) and a is a constant. We now will compare two different expressions to see why commutation relations are helpful in approximations:

$$\begin{aligned} e^{a(M+N)} &= I + a(M+N) + \frac{a^2}{2}(M+N)^2 + O(a^3) \\ &= I + a(M+N) + \frac{a^2}{2}(M^2 + MN + NM + N^2) + O(a^3) \end{aligned} \tag{7}$$

$$\begin{aligned} e^{a(M)}e^{a(N)} &= (I + a(M) + \frac{a^2}{2}(M^2) + O(a^3))(I + a(N) + \frac{a^2}{2}(N^2) + O(a^3)) \\ &= I + a(M+N) + \frac{a^2}{2}(M^2 + 2MN + N^2) + O(a^3) \end{aligned} \tag{8}$$

We can see from the above equations that the primary difference between the two lies in how the operators (matrices) M and N commute with each other. If we subtract equation (7) from equation (8) up to the second order terms, we find that the difference is $-MN + NM$, which just so happens to be the negative commutator of M and N. This allows us to rewrite equation (8) in terms of commutators:

$$e^{a(M)}e^{a(N)} = e^{I+(M+N)-\frac{a^2}{2}[M,N]+O(a^3)} \tag{9}$$

Equation (9) actually demonstrates the BCH formula. It can be similarly incorporated for hamiltonians as demonstrated in equation (10). The BCH formula allows us to approximate the commutation relation of two hamiltonians up to a certain order. It works as follows for a Hamiltonian $\hat{H}$ such that $\hat{H} = \hat{A} + \hat{B}$ [3]:

$$e^{-i\hat{B}t}e^{-i\hat{A}t} = e^{-i(B+A)t+\frac{t^2}{2}[B,A]-\frac{t^3}{6}[B,[B,A]]+\frac{t^3}{6}[A,[B,A]]...} \tag{10}$$

3.2 The Trotter Formula and Trotterization

The Trotter formula is the result of taking the BCH formula and using it to asymptotically approximate a hamiltonian which can be broken into two or more parts, demonstrated by

$$\begin{aligned}\lim_{n\to\infty}(e^{-i(\hat{B}/n)t}e^{-i(\hat{A}/n)t})^n &= \lim_{n\to\infty}\sum_{k=0}^{n}\frac{(i(\hat{B}+\hat{A})t)^k}{k!}(1+O(\frac{1}{n}))+O(\frac{1}{n}) \\ &= e^{-i(\hat{B}+\hat{A})t}\end{aligned} \tag{11}$$

The most useful part of this formula is that the relation demonstrated holds even if the operators $\hat{A}$ and $\hat{B}$ do not commute.

3.3 Higher-Order Product Formulas

In theory, the Trotter formula is a "perfect" way to simulate hamiltonians, where "perfect" describes with utmost precision. However, this isn't a practical representation when it comes to simulation on a quantum computer. What can be done is to make approximations of this formula [3]. Equation (9) symbolizes the simplest way a hamiltonian which can be split into two sub-hamiltonians can be approximated. In this case, a first approximation is produced. This is known as the first-order Lie-Trotter Formula. For higher order cases, the product formulas will be referred to as Suzuki formulas. We can demonstrate such examples with higher-order product formulas which are written in a cyclic pattern due to their recursive nature [4]. We can use a hamiltonian $\hat{H}$ such that $\hat{H} = \hat{A} + \hat{B}$ to evaluate the general pattern which can be used to split hamiltonians appropriately into n parts:

$$e^{\hat{H}} = e^{\hat{A}+\hat{B}} = e^{c_1x\hat{A}}e^{c_2x\hat{B}}e^{c_3x\hat{A}}...e^{c_{n-1}x\hat{B}}e^{c_nx\hat{A}} \tag{12}$$

We will first look at a base case. The following formula depicts a second-order approximation for the same hamiltonian as referred to above. We will call it the S_2 case:

$$S_2(x) = e^{\frac{x}{2}\hat{A}}e^{x\hat{B}}e^{\frac{x}{2}\hat{A}} \tag{13}$$

It is important to note that the orders of approximation of Suzuki formulas go up by magnitudes of 2, so the next order up will be a fourth order appxoximation. Using S_2, we can construct a fourth order approximation recursively:

$$\begin{aligned}S_4(x) &= S_2(sx)S_2((1-2s)x)S_2(sx) \\ &= e^{\frac{s}{2}x\hat{A}}e^{sx\hat{B}}e^{\frac{1-s}{2}x\hat{A}}e^{(1-2s)x\hat{B}}e^{\frac{1-s}{2}x\hat{A}}e^{sx\hat{B}}e^{\frac{s}{2}x\hat{A}}\end{aligned} \tag{14}$$

where s is a constant which needs to be calculated. We can calculate it by taylor expansion of each of the terms to the third order. If we do so, and combine terms such that they include $\hat{A} + \hat{B}$ in a cyclic manner, we can find the governing relation for s for this particular approximation. Doing this, we get

$$S_4(x) = e^{x(\hat{A}+\hat{B})} e^{(2s^3+(1-2s)^3)} R_3 + O(x^5) \tag{15}$$

where R_3 is representing a 3rd degree relation between $\hat{A}$ and $\hat{B}$. We can calculate the value for s by calculating it for $2s^3 + (1-2s)^3 = 0$, as found in equation (15). This yields the result $s = \frac{1}{2-\sqrt[3]{2}}$.

We can take other fourth-order approximations by varying how we split up the approximation. For example, another way of constructing S_4 can be done as follows:

$$S_4(x) = S_2(sx)^2 S_2((1-4s)x) S_2(sx)^2 \tag{16}$$

The important part is that the number of sx terms add up to 0. s can be calculated in a similar manner per the previous example, by taylor expanding each of the terms up to the third order and combining terms such that each of them includes $\hat{A} + \hat{B}$ in a cyclic manner.

In addition, if we want to construct even higher-order approximations, we can construct them in the following manner:

$$S_6(x) = S_4(sx)^2 S_4((1-4s)x) S_4(sx)^2 \tag{17}$$

All in all, the way that the polynomial is supposed to be constructed depends on the individual characteristics of the quantum system which is supposed to be modeled.

4 The Full Quantum Simulation Algorithm

Before we take a dive into the full hamiltonian simulation algorithm, we evaluate the purpose of the algorithm, its inputs, and its outputs. The purpose of the hamiltonian simulation algorithm is to *most efficiently* simulate a given hamiltonian on a quantum computer.

In order to accomplish this, we need the following inputs: a Hamiltonian $\hat{H}$ such that $\hat{H} = \sum_{n=0}^{k} \hat{H}_n$ an initial quantum state $|\psi_0\rangle$, an accuracy parameter δ which determines the accuracy to which the hamiltonian needs to be simulated to, and a time t_f over which the system needs to be evolved over.

Given these inputs, we need to produce the following outputs: A final state $|\psi(t_f)\rangle$ such that the following condition is met:

$$| \langle \psi(t_f)| e^{-i\hat{H}t_f} |\psi_0\rangle |^2 \geq 1 - \delta \tag{18}$$

We can now look at the algorithm step-by-step [5]:

We must first make some selections about how we are going to represent the system on a quantum computer. The first thing we must do is choose a number of qubits $n = O(poly(log(n))$ which can accurately represent the system. We must also choose operators $e^{-iH_k\Delta t}$ which can be accurately represented by quantum circuits.

Now that we have our system set up, we must select an approximation method. We can select this by using the Trotterization techniques and selecting a product formula of desired approximation for our system.

Finally, we must select a Δt and j such that conditions of expected error, determined via δ, and $j\Delta t = t_f$ are met.

Algorithm 1 Hamiltonian Simulation Algorithm

1: Initialize $|\psi_0\rangle = |\psi(0)\rangle$ ▷ initial state
2: Initialize $j = 0$ ▷ interative variable
3: **while** $j\Delta t < t_f$ **do** ▷ state evolution
4: $\quad |\psi_{j+1}\rangle = U_{\Delta t} |\psi_j\rangle$
5: $\quad j = j + 1$
6: **end while**
7: $|\psi_f\rangle = |\psi(t_f)\rangle$ ▷ final state

5 Current Findings and Future Work

While hamiltonian simulation proves to be one of the more useful cases for quantum computers, there are still many improvements that can be made to the existing algorithms today which might make this task more efficient to perform. In particular, much work has been done on the sample and query complexity of hamiltonian simulation, especially in the particular scenario when the description of hamiltonian is sourced classical methods or with a black box [6].

One particular type of hamiltonian which is considered difficult to simulate with current methods is the sparse hamiltonian. A **d-sparse hamiltonian** is a hamiltonian with at most d nonzero entries in any of its rows or columns. In particular, the way the sparse hamiltonian problems works is that we have access to the hamiltonian via a black box which takes in a value i for a row position and a value j for the column position, each of which can take values between 1 and d to denote the position nonzero entry of the hamiltonian matrix. Our goal is to perform the hamiltonian simulation algorithm with the fewest number of queries to the black box as possible, given the same parameters as defined in the algorithm. An additional step which needs to be taken is to upper-bound

the number of two-qubit gates which can be used since the time complexity of the algorithm varies based on the number of queries to the black box and the number of two-qubit gates used [7].

Another method for hamiltonian simulation arises from continuous- and discrete-time quantum random walks. In this case, a quantum walk is a time-homogeneous quantum process whose information can be stored on a graph [8]. We can define a **coin operator**, which is a unitary operator which operates in the Hilbert space of the system that helps evolve the system in accordance with discretized time slices in accordance with a chosen shift operator. It is the main component of a discrete-time quantum random walk. It is chosen specifically for each system such that it can approximate the behavior of a continuous-time quantum random walk. In addition, if we are given the spectral decomposition of the graph of this hamiltonian in addition to a few more bounds, then we can solve for the discrete-time evolution of the system at or near important values of the given parameters [9]. This technique works well, even for sparse hamiltonians. It is also important to note that the complexity of the simulation is linear in the total evolution time, an improvement over simulations based on high-order approximations of the Lie product formula [8].

Another modification to the quantum simulation algorithm which can cause it to produce more accurate results are by randomizing the order for which the operands are applied in the chosen product formula. One such example deals with equations (7) and (8). It is not possible to approximate equation (7) using a product of only two exponentials of M and N in this case. We would have to implement a similar term like equation (8), except with the matrix operands in reverse order, and take a linear combination of the unitaries so that it can be applied to a circuit. However, this would result in a circuit with many ancilla qubits and would have a high cost which varies with the number of summands in the hamiltonian matrix. A solution to this could be to take one of the operations at random, thereby implementing a quantum channel that gives a good approximation to the desired evolution [10].

In addition to these methods, there are constantly new methods being developed to make hamiltonian simulations more efficient and more accurate by finding new ways to simulate higher-order terms.

6 Conclusion

Hamiltonian Simulation is one of the most important up-and-coming use cases for quantum computers. Starting from a single formula, many different techniques could branch out to improve how the formula, and thereby the algorithm, can accurately simulate hamiltonians. These simulation methods prove to be far more efficient at simulating such systems compared to classical computers with the available hardware.

In this work, we looked at the needed formulas into breaking hamiltonians into several different sub-hamiltonians. In addition to this, we also looked at how such hamiltonians can be approximated to varying degrees and how

such approximations can be varied with time in an algorithm setup. Finally, we looked at some further improvements to the algorithm and other potential methods of quantum simulation.

7 Acknowledgements

I would like to thank Dr. Nick Hunter-Jones in his guidance for this project and helping me to build a solid foundation in understanding these concepts. I would also like to thank Shivan Mittal for helping me to understand these concepts further and for discussing and working out examples and textbook problems to guage a deeper understanding of the topic material.

References

[1] **Scott Aaronson**, Lecture 25: Hamiltonians.

[2] **Carl F. Bender**, Making Sense of non-Hermitian Hamiltonians.

[3] **Andrew M. Childs, Minh C. Tran, Nathan Wiebe, Shuchen Zhu**, A Theory of Trotter Error.

[4] **Naomichi Hatano, Masuo Suzuki**, Finding Exponential Product Formulas of Higher Orders.

[5] **Michael Nielsen, Isaac Chuang**, Quantum Computation and Quantum Information

[6] **Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, Theodore J. Yoder**, Hamiltonian simulation with optimal sample complexity

[7] **Dominic W. Berry, Andrew M. Childs, Richard Cleve, Rolando D. Somma**, Exponential improvement in precision for simulating sparse Hamiltonians

[8] **Andrew M. Childs**, On the relationship between continuous- and discrete-time quantum walk

[9] **A.T. Schmitz, W.A. Schwalm**, Simulating continuous-time Hamiltonian dynamics by way of a discrete-time quantum walk

[10] **Andrew M. Childs, Aaron Ostrander, Yuan Su**, Faster Quantum Simulation by Randomization