# Report: Seminar Advanced Topics of Quantum Computing

| | |
|---|---|
| Author: | Patrick Bettermann (03683053) |
| Last compiled: | 2022–08–08 |
| Source code: | https://github.com/Pabet/Hamiltonian-Simulation |

## Hamiltonian Simulation

### Hamiltonian

The Hamiltonian of a system can be understood as the sum of the kinetic energies of all the particles, plus the potential energy of the particles associated with the system. In the field of Quantum Computing the Hamiltonian operator generates the evolution of a quantum state.
The problem statement can be written as follows:

Given a description of a Hamiltonian H, and evolution time t, some initial state $|\psi(0)\rangle$, produce the final state $|\psi(t)\rangle$ (to some error $\epsilon$) [2]

However problems arise when this is tried on a classical computer.

- A classical computer can't store a quantum state efficiently. Due to the nature of qubits, classical 0/1-bits can't represent the state of a system without exponential storage. [2]
- A classical computer cannot produce a complete description of the state since we only have a copy of the quantum state which we can use to create statistics about what could happen if we measure the state.[2]

### Schrödinger Equation

If $|\psi(t)\rangle$ is the state of the system at time t, then

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

If $H$ is dependent on time we can write by integrating both sides:

$$|\psi(t)\rangle = e^{-iHt/\hbar}|\psi(0)\rangle\,[1] \tag{2}$$

$U = e^{-iHt/\hbar}$ is a unitary operator

$H$ represents the total energy of the system. Therefore we can break it down into kinetic energy

$$E_{kin} = \frac{1}{2}mv^2 \tag{3}$$

with the momentum $p = mv, v = \frac{p}{m}$ we can rewrite the equation to

$$E_{kin} = \frac{p^2}{2m} \tag{4}$$

Since $e^{\frac{-ip^2t}{2m\hbar}}$ and $e^{\frac{-iE_{pot}t}{\hbar}}$ don't commute the following equations holds and we can't simply split our Hamiltonian into two different terms.

$$e^{-iHt/\hbar} \neq e^{\frac{-ip^2t}{2m\hbar}}e^{\frac{-iE_{pot}t}{\hbar}} \tag{5}$$

Product formulas can solve this problem.

### Lattice Hamiltonians

For Hamiltonian simulation we consider a class of Hamiltonians that are called lattice Hamiltonians. Lattice systems arise for example in the Fermi-Hubbard which the authors of [4] chose as system of demonstration because it's a well known system that is used outside of the field of quantum computing. Therefore one cannot argue that the system is arbitrarily set up to be solved by quantum computers. Latice Hamiltonians consider the local interactions between qubits [3]. The principle of locality often occurs in physics where many interactions depend on the distance between particles. For numerical simulations interactions between far away particles are often not considered because forces over long distances relative to the system size tend to go towards zero. In our system we consider nearest-neighbor interactions. We talk about a lattice Hamiltonian if the Hamiltonian $H$ acts on $n$ qubits and we can decompose it as

$$H = \sum_{j=1}^{n-1} H_{j,j+1]} \tag{6}$$

where each $H_{j,j+1}$ is a Hermitian operator that acts nontrivially only on quibts $j$ and $j+1$ according to [3] The simulation of a lattice Hamiltonian $H = \sum_{j=1}^{n-1} H_{j,j+1}$ can be achieved with a Trotter-algorithm. Here the system is evolved with each interaction $h_{i,j}$ for a small time-step $\delta$ [4].

$$e^{-iHT} \simeq \prod_{n=0}^{T/\delta} \left( \prod_{<i,j>} e^{-ih_{i,j}\delta} \right) \tag{7}$$

### Lie-Trotter Formula

The idea behind product formulas is to simulate the sum-terms of a Hamiltonian by simulating each one separately for a small time slice. Our goal is to split a Hamiltonian $H$ into multiple smaller terms A, B and C for example which might be of a simpler structure than the Hamiltonian H.

$$H = A + B + C \tag{8}$$

to achieve this we utilize the Lie product formula

$$e^{A+B+C} = \lim_{n\to\infty} (e^{A/n} e^{B/n} e^{C/n})^n \tag{9}$$

with $e^A$ the matrix exponential of $A$.
In the example from the paragraph Schrödinger Euqation where $H$ represents the total energy of the system split into kinetic and potential energy the simulation of the sum-terms would correspond to infinitely switching between kinetic and potential energy terms of smaller size to continue the evolution of the system.

## Hamiltonian simulation algorithms for near-term quantum hardware[4]

Problem: Find explicit gate count for near-term simulations
The gate count of near-term quantum hardware is very limited therefore we need to make an effort to use the available hardware in the best way possible. Furthermore algorithms need to be optimized to use as few gates as possible. Evenhough the authors managed to read a circuit-depth-equivalent of 259 in the per-time error model. This is still too costly for contemporary quantum computers. So [4] remains a theoretical optimization. They achieved this optimization through optimizations on multiple levels, namely:

- Sub-circuits: programming below the circuit model
- Non-asymptotic Trotter bounds
- Optimal fermion encoding

of which the sub-circuit model is considered for this work.

### Sub-Circuit Model

The authors derive a Hamiltonian $H = H_1 + H_2 + H_3 + H_4 + H_5$ with layers $H_1$ to $H_5$ where the Hamiltonians for each layer don't commute with eachoher [4]. Depending on the encoding, i.e. the way the fermionic Fock space is represented by qubits, we would like to compute the two-, three- (compact encoding) or four-local (VC encoding) Hamiltonians directly within a quantum computer. As the authors state this is unfortunately an unrealistic goal since these k-local interactions are most likely not available on the current hardware [4]. More likely a restricted set of one- and two-qubit interactions is available.
We therefore assume that only arbitrary single-qubit unitaries are available to us and that we have access to the family of gates

$$e^{itZ\otimes Z} \tag{10}$$

for arbitrary values of t.
The type of gates which are required for Hamiltonian simulation are of the form

$$e^{i\delta Z^{\otimes k}} \tag{11}$$

for k = 3 or k = 4
The authors introduce the Sub-Circuit model as follows:

- given a set of qubits $Q$
- a set of interactions $I \subseteq Q \times Q$ indicating which pairs of qubits interact
- a two qubit interaction Hamiltonian $h$
- minimum switching time $t_{min}$

The sub-circuit $C$ is defined as $L$ pairs

$$C = \prod_{l}^{L} U_l V_l \tag{12}$$

with

$$U_l = \prod_{i \in Q} u_i^l \tag{13}$$

a layer of arbitrary single qubit unitary gates and

$$V_l = \prod_{ij \in \Gamma_l} v_{ij} \left( t_{ij}^l \right) \tag{14}$$

a layer of non-verlapping, variable time, two-qubit unitary gates and

$$v_{ij}(t) = e^{ith_{ij}} \tag{15}$$

The set $\Gamma_l \subseteq I$ contains the non overlapping pairs of qubits. The equality

$$h_{ij} = Z_i Z_j \tag{16}$$

is assumed. Since $\sigma_i \sigma_j$ equivalent to $Z_i Z_j$ up to single qubit rotations. We therefore write

$$h_{ij} = \sigma_i \sigma_j \tag{17}$$

All equation are taken from the supplementary of [5]

## Implementation

### Sub-Circuit Synthesis

The sub-circuit model is applied to the circuit $e^{itZ^{\otimes 4}}$.

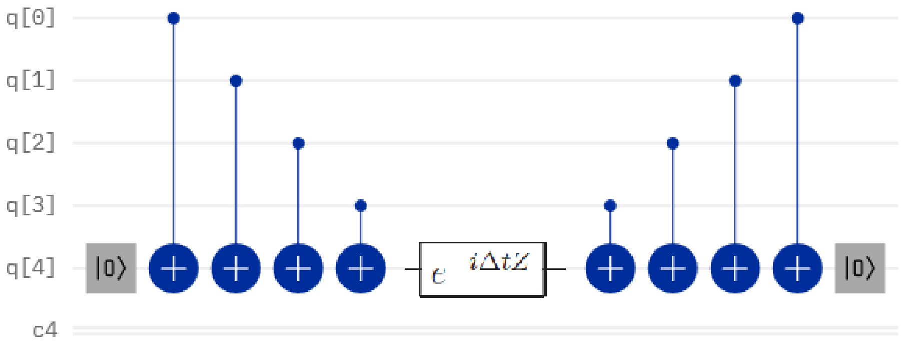

Figure 1: Quantum circuit for simulating $H = Z_1 \otimes Z_2 \otimes Z_3 \otimes Z_4$ for time $\Delta t$ [6]

Lemma 8 [5] is applied to decompose the circuit:

$$U(t) = e^{it_1^b Y_2 Z_3 Z_4} e^{i\phi Z_1 X_2} e^{it_1^b Y_2 Z_3 Z_4} e^{i\phi Z_1 X_2} e^{it_1^b Y_2 Z_3 Z_4} \tag{18}$$

where $Y_2$ denotes the Pauli-Y gate acting on the 2nd qubit and $Z_3$ the Pauli-Z gate acting on the third qubit. We further decompose circuit $e^{it_1^b Y_2 Z_3 Z_4}$ with the 3-local Hamiltonian $H = Y_2 \otimes Z_3 \otimes Z_4$.
where $Y_2$ can be represnted by

$$Y_2 = R_{Z,2}(\frac{\pi}{2}) HZH R_{Z,2}(\frac{-\pi}{2}) \tag{19}$$

with $R_{Z,2}$ denoting the rotation operator around the z-axis acting on the 2nd qubit

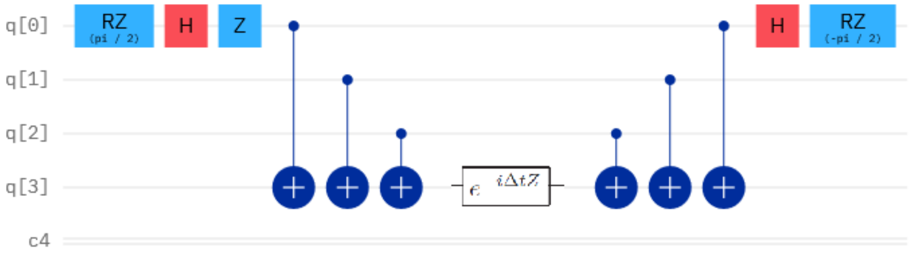

Figure 2: Quantum circuit for simulating $Y_1 \otimes Z_2 \otimes Z_3$ for time $\Delta t$

H is represented by

$$R_{Z,2}(\frac{\pi}{2}) HZ_2H R_{Z,2}(\frac{-\pi}{2}) \otimes Z_3 \otimes Z_4 \tag{20}$$

the whole circuit now has the form:

$$\begin{aligned} U(t) = {} & e^{it_1^b R_{Z,2}(\frac{\pi}{2}) HZ_2H R_{Z,2}(\frac{-\pi}{2}) \otimes Z_3 \otimes Z_4} \\ & e^{i\phi Z_1 X_2} \\ & e^{it_1^b R_{Z,2}(\frac{\pi}{2}) HZ_2H R_{Z,2}(\frac{-\pi}{2}) \otimes Z_3 \otimes Z_4} \\ & e^{i\phi Z_1 X_2} \\ & e^{R_{Z,2}(\frac{\pi}{2}) HZ_2H R_{Z,2}(\frac{-\pi}{2}) \otimes Z_3 \otimes Z_4} \end{aligned} \tag{21}$$

using $X = HZH$ with X as Pauli-X gate and H as Hadamard gate

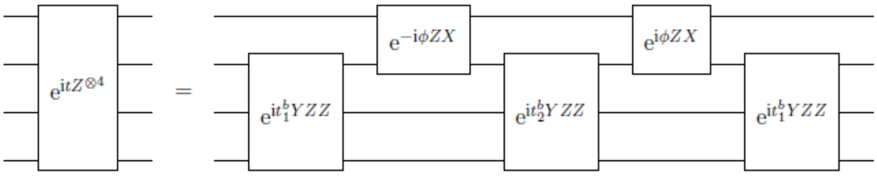

Figure 3: Sub-Circuit decomposition of depth 5 according to [5]

### Sub-Circuit Implementation

The sub-circuit decomposition of $e^{itZ^{\otimes 4}}$ is implemented as decribed in detail in subcircuit_implementation.ipynb. First the initial state of the system is defined

$$y_0 = q_1 \otimes q_2 \otimes q_3 \otimes q_4 \tag{22}$$

with $q_1 = |10\rangle$, $q_2 = |10\rangle$ ,$q_3 = |01\rangle$, $q_4 = |01\rangle$

The values for the qubits are chosen in a way that the result of time-evolution is not zero since in the circuit $e^{itZ^{\otimes 4}}$ the phase shift $e^{i\Delta t}$is applied to the system if the parity of the n qubits is even [6]. The algorithm works by subsequently applying the following steps

- Apply $e^{it_1^b R_{Z,1}(\frac{\pi}{2}) HZHR_{Z,1}(\frac{-\pi}{2}) \otimes Z_3 \otimes Z_4}$ to q2-q4
- Apply $e^{i\phi Z_1 X_2}$ to q1-q2
- Apply $e^{it_2^b R_{Z,1}(\frac{\pi}{2}) HZHR_{Z,1}(\frac{-\pi}{2}) \otimes Z_3 \otimes Z_4}$ to q2-q4
- Apply $e^{i\phi Z_1 X_2}$ to q1-q2
- Apply $e^{it_1^b R_{Z,1}(\frac{\pi}{2}) HZHR_{Z,1}(\frac{-\pi}{2}) \otimes Z_3 \otimes Z_4}$ to q2-q4

with

$$\phi = (\frac{1}{4}(3 + 2\sqrt{2})t)^{\frac{1}{3}(23)}$$

and

$$t_1^a = \frac{1}{2} tan^{-1} \left( \sqrt{2} sec(t) csc(2\phi) \sqrt{cos(2) - cos(4\phi)} \right) + \pi c \quad (24)$$

$$t_1^b = tan^{-1} \left( -2tan(t) cot(2\phi) \right) + \pi c \quad (25)$$

$$t_2^a = tan^{-1} \left( \frac{csc(2\phi) \sqrt{cos(2t) - cos(4\phi)}}{\sqrt{2}} \right) + \pi c \quad (26)$$

$$t_2^b = tan^{-1} \left( sin(t) csc(2\phi) \right) + 2\pi c \quad (27)$$

according to [5]

**Comparison**

To test the implementation of the sub-circuit model a theoretical solution is diretly calculated using scipy.linalg.expm to directly calculate the matrix exponential of the Hamiltonian $e^{itZ^{\otimes 4}}$. The subcircuit solution is calculated for different Trotter-steps $\delta$ ranging from $10^{-5}$ to $10^{0}$. The difference of theoretical and practical solution is then plotted on a log-log graph.

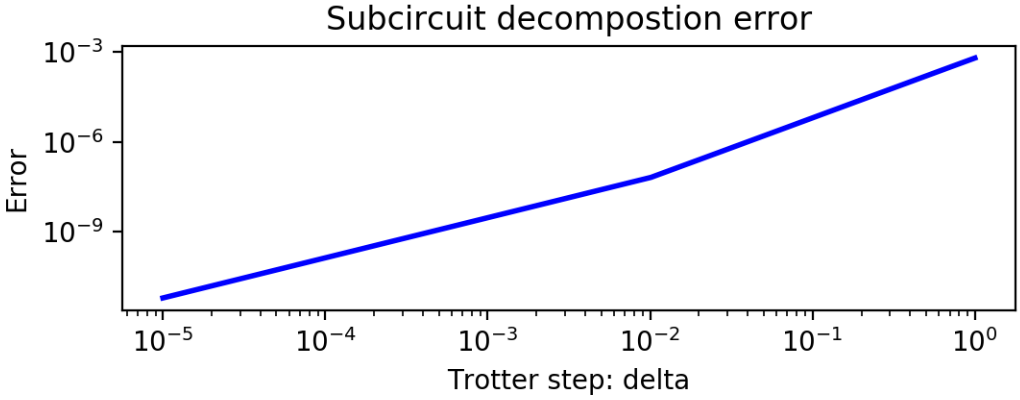

Figure 4: Comparison of the error induced by the sub-circuit model and the Trotter-step size with the theoretical solution

On the log-log scale the error scales linearly with the Trotter-step size. Smaller Trotter-steps result in a lower error. On the other hand a smaller Trotter-step size means that we need to calculate more iterations to cover the same time-evolution $\Delta t$ making the computation more costly. Overall we succeeded in decompostion the 3- and 4-local Hamiltonians into 1- and 2-qubit gates. This procedure allows us to simulate many different Hamiltonians of the form

$$H = \otimes_{k=1}^{n} \sigma_{c(k)}^{k} \tag{28}$$

with $\sigma_{c(k)}^{k} \in I, X, Y, Z$ [6]
As shown above these types gates can simply be represented by eachother using single-qubit rotations.

**Outlook**

The authors managed to greatly reduce the gate count of the quantum circuit down to 259 gates using the sub-circuit model and efficient encoding and good Trotter error bounds[4] making Hamiltonian Simulation more plausible on near-tearm hardware. However even this great reduction in gate count is not yet enough to make Hamiltonian Simulation feasible on current quantum hardware. Like in the early days of computing every single optimization counts to use the available hardware to the fullest. The sub-circuit model is a good start to optimize the process on the hardware-level and more research and optimizations will likely make Hamiltonian Simulation available on real hardware in the near future.

# References

[1] Graeme Robert Ahokas. *Improved algorithms for approximate quantum Fourier transforms and sparse Hamiltonian simulations*. University of Calgary, 2004.

[2] Andrew Childs. Qip 2021 tutorial: Quantum algorithms.

[3] Andrew M Childs and Yuan Su. Nearly optimal lattice simulation by product formulas. *Physical review letters*, 123(5):050503, 2019.

[4] Laura Clinton, Johannes Bausch, and Toby Cubitt. Hamiltonian simulation algorithms for near-term quantum hardware. *Nature communications*, 12(1):1–10, 2021.

[5] Laura Clinton, Johannes Bausch, and Toby Cubitt. Hamiltonian simulation algorithms for near-term quantum hardware - supplementary information. *Nature communications*, 12(1):1–10, 2021.

[6] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.