# 11. Brief survey on Hamiltonian Simulation/ Power of Block Encoding

Isaac H. Kim (UC Davis)

# Recap

Qubitization is a very flexible modern framework for developing quantum algorithms.

While the unitary encoding we discussed last time is somewhat simplistic, it captures the essential ideas.

Many of the recent advances in Hamiltonian simulation algorithms use the framework of qubitization. Improvements were made in SELECT+PREPARE subroutine, which utilizes the special structure of the Hamiltonian.

$H = \sum_{i=1}^{N} \alpha_i P_i$

$\alpha_i$ all different: → Prepare requires preparation of $\log_2 N$-qubit state
cost: $O(N)$

$\alpha_i$ all same → Uniform superposition : n-qubit system $H^{\otimes n}|0\ldots0\rangle$ $n = \log_2 N$
cost: $O(\log N)$

# Hamiltonian Simulation: A brief survey

1. Qubitization: The de facto standard
2. LCU: Still good for time-dependent simulation
3. Trotter: Recent comeback

$e^{-iHt}$

# Qubitization

1. Optimal (assuming the block encoding is given as a black box)
2. Practical
3. The standard method of choice for time-independent Hamiltonian simulation
4. Especially good for realistic quantum chemistry Hamiltonians, they outperform the other methods by a wide margin.

# LCU

1. For time-independent Hamiltonians, inferior compared to qubitization.

2. This works best for time-dependent Hamiltonian, e.g., in the interaction picture. [Kieferova, Schrer, and Berry (2018), Low and Wiebe (2018)]

3. Nearly optimal

4. Realistic gate count estimate requires a lot of work in practice.

$H = H_1 + H_2$ "complicated" terms ex) $H_2 = \sum_{i=1}^{n} X_i X_{i+1} + Y_i Y_{i+1}$

"simple" ex) $H_1 = \sum_{i=1}^{n} Z_i$

↓ Interaction picture

$H(t) = e^{itH_1} H_2 e^{-itH_1} \Rightarrow e^{itH_1}(X_i X_{i+1}) e^{-itH_1} = e^{it(Z_i + Z_{i+1})} X_i X_{i+1} e^{-it(Z_i Z_{i+1})}$

# Trotter-Suzuki

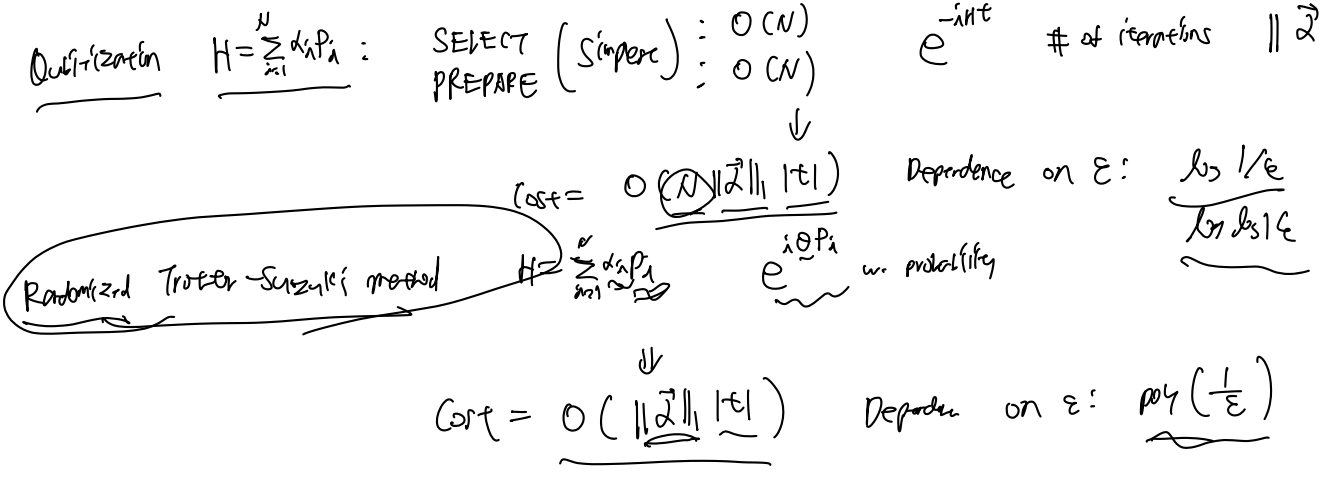
1. For a very long time, small-scale simulation indicated that Trotter method works much better than expected. Now this discrepancy is almost resolved. [Childs et al. (2o19)]

2. With randomization, the complexity of the Trotter-Suzuki method scales *differently*. [Campbell (2018)]

# Trotter-Suzuki vs. Qubitization

Qubitization: Cost is essentially determined by SELECT+PREPARE. Both subroutines scale linearly with the *number of terms* in the Hamiltonian.

Randomized Trotter-Suzuki: The cost *does not scale* with the number of terms (but it does with the absolute sum of Hamiltonian strengths). [Campbell (2018)]



# Beyond Hamiltonian Simulation

We have already discussed the utility of applying a "weird" time evolution, e.g., $e^{i\cos^{-1}(H)}$. Examples like this suggest that there is a room to study applications of "unphysical" operators.

Quantum Singular Value Transformation is a flexible framework to explore such possibilities.

# Quantum Singular Value Transformation

We assumed an input model of the following form:

$U(H)\,|G\rangle_a\,|\psi\rangle_s = |G\rangle_a H\,|\psi\rangle_s + |G_\psi^\perp\rangle_{as}$,

where $H$ is a *hermitian* matrix. It turns out that many of our conclusion follows even if $H$ is not hermitian.

Consider an input model

$U(A)\,|G\rangle_a\,|\psi\rangle_s = |G\rangle_a A\,|\psi\rangle_s + |G_\psi^\perp\rangle_{as}$,

where $A = USV^\dagger$. Using the qubitization technique, we can synthesize

$\widetilde{U}(A)\,|G\rangle_a\,|\psi\rangle_s = |G\rangle_a UP(S)V^\dagger\,|\psi\rangle_s + |\widetilde{G}_\psi^\perp\rangle_{as}$

for certain polynomials $P(x)$. [Gilyén, Su, Low, and Wiebe (2018)]

This is known as the Quantum Singular Value Transformation (QSVT).

# Quantum Singular Value Transformation

QSVT is very powerful. It often leads to simple and efficient quantum algorithms for a variety of problems.

$|\psi\rangle \to A^{-1}|\psi\rangle / \|A^{-1}|\psi\rangle\|$

1. Hamiltonian Simulation
2. Applying $A^{-1}$ (Moore-Penrose pseudo-inverse)
3. Amplitude amplification
4. Fractional query: applying $U^{\alpha}$, $0 < \alpha < 1$, given access to $U$.
5. etc…

# Block Encoding

QSVT makes quantum algorithm development very simple. If your goal is to apply a matrix function to a state, i.e., $|\psi\rangle \to f(A)\,|\psi\rangle$, you just need to figure out two things.

1. How do we encode $A$ into a unitary?
2. How well can we approximate $f(x)$ by a low-degree polynomial?

# Block Encoding

QSVT makes quantum algorithm development very simple. If your goal is to apply a matrix function to a state, i.e., $|\psi\rangle \to f(A)|\psi\rangle$, you just need to figure out two things.

1. How do we encode $A$ into a unitary? **case-by-case**
2. How well can we approximate $f(x)$ by a low-degree polynomial? (Often) **solved**

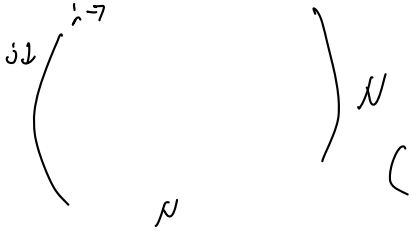
# Block Encoding Frameworks

Fortunately, there are already powerful frameworks for block encoding.

1. SELECT + PREPARE: Hamiltonian Simulation
2. Purification of Density Matrix: Machine Learning
3. Sparse Matrix: Systems of linear equations

$U = \begin{pmatrix} \rho & \cdot \\ \cdot & \cdot \end{pmatrix}$ $\rho$: Density matrix

You can implement $U$ if you can prepare a purification of $\rho$.

$A =$ 


( i: Overlaps a set of j s.t. $A_{ij} \neq 0$
2) Given i, j s.t. $A_{ij} \neq 0$ $|i\rangle|j\rangle|x\rangle \rightarrow |i\rangle|j\rangle|x \oplus A_{ij}\rangle$

# Block Encoding Arithmetics

Fortunately, there are already powerful frameworks for block encoding.

1. SELECT + PREPARE: Hamiltonian Simulation
2. Purification of Density Matrix: Machine Learning
3. Sparse Matrix: Systems of linear equations

But for your application, maybe none of these will actually work. What to do then?

# Block Encoding Arithmetics

There is no general solution to this problem, but there are well-known tricks you can use. These are all based on block encoding arithmetics.

The moral of the story will be very simple. If you have unitary encodings of $A_1, A_2, \ldots, A_n$, you can construct a unitary encoding for any element of the algebra generated by these matrices.

$U(A_1) = \begin{pmatrix} A_1 & \cdot \\ \cdot & \cdot \end{pmatrix}$ $U(A_2) = \begin{pmatrix} A_2 & \cdot \\ \cdot & \cdot \end{pmatrix}$ , - - -

$U(\tilde{A}) = \begin{pmatrix} \tilde{A} & \cdot \\ \cdot & \cdot \end{pmatrix}$ , where $\tilde{A}$ is (up to normalization) in the algebra generated by $A_1, \ldots, A_n$

# Block Encoding Arithmetics

If you have unitary encodings of $A_1, A_2, \ldots, A_n$, you can construct a unitary encoding for any element of the algebra generated by these matrices.

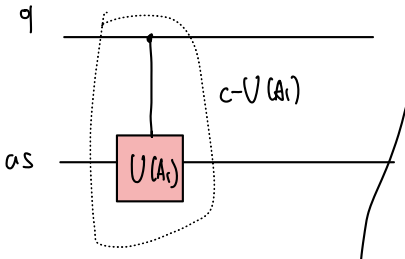
To prove this claim, what we need to do is very simple. Given unitary encodings of two matrices $A$ and $B$, construct a unitary encoding of $\alpha A + \beta B$ for any $\alpha, \beta \in \mathbb{C}$and also $AB$.

# Scalar multiplication

$$U(A_1) = \begin{pmatrix} A_1 & \cdots \\ \vdots & \ddots \end{pmatrix}$$

$$U(A_1)\,|G\rangle_a |\psi\rangle_s = |G\rangle_a A_1|\psi\rangle_s + |G_\psi^\perp\rangle_{as}$$

$$(\langle G|_a \otimes I_s)\,|G_\psi^\perp\rangle_{as} = 0$$




$$c\text{-}U(A_1)(\alpha|0\rangle + \beta|1\rangle)_q\,|G\rangle_a|\psi\rangle_s$$

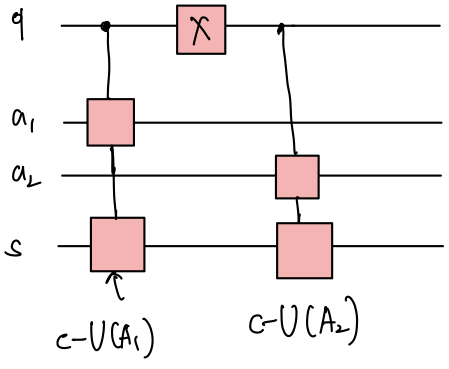
$$= \alpha|0\rangle_q|G\rangle_a|\psi\rangle_s + \beta|1\rangle_q|G\rangle_a A_1|\psi\rangle_s$$

$$= \alpha|0\rangle_q|G\rangle_a|\psi\rangle_s + |1\rangle_q|G\rangle_a (\beta A_1|\psi\rangle_s)$$

New ancilla stores $|1\rangle_q|G\rangle_a$

$|1\rangle_q|G\rangle_a$

# Addition




$$U(A_1)\,|G\rangle_{a_1}|\psi\rangle_S = |G\rangle_{a_1} A_1|\psi\rangle_S + |G^{\perp}_{\psi,1}\rangle$$

$$U(A_2)\,|G\rangle_{a_2}|\psi\rangle_S = |G\rangle_{a_2} A_2|\psi\rangle_S + |G^{\perp}_{\psi,2}\rangle$$

$$(\alpha|0\rangle + \beta|1\rangle)_q\,|G\rangle_{a_1}|G\rangle_{a_2}|\psi\rangle_S$$

$$= \alpha|0\rangle_q\,|G\rangle_{a_1}|G\rangle_{a_2}|\psi\rangle_S$$

$$\downarrow$$

$$\alpha|1\rangle_q\,|G\rangle_{a_1}|G\rangle_{a_2} A_2|\psi\rangle_S + \beta|0\rangle_q\,|G\rangle_{a_1}|G\rangle_{a_2} A_1|\psi\rangle_S + \cdots$$

$$= |1\rangle_q\,|G\rangle_{a_1}|G\rangle_{a_2} \alpha A_2|\psi\rangle_S + |0\rangle_q\,|G\rangle_{a_1}|G\rangle_{a_2} \beta A_1|\psi\rangle_S + \cdots$$

Ancillas: $(\alpha|0\rangle+\beta|1\rangle)\,|G\rangle_{a_1}|G\rangle_{a_2}$

# Mulitiplication

$U(A_1)$ $U(A_2)$

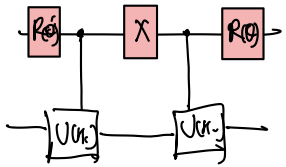
$U(A_2)\,U(A_1)\,|G_1\rangle_{a_1}|G_2\rangle_{a_2}|\psi\rangle_S = |G_1\rangle_{a_1}|G_2\rangle_{a_2} A_2A_1|\psi\rangle_S + |G_\psi^\perp\rangle$

$\left(\langle G_1|_{a_1}\langle G_2|_{a_2}\otimes I_S\right)|G_\psi^\perp\rangle = 0$

$H_s = H_0(1-s) + H_1 s$

$U(H_1)$ $U(H_2)$

$U(H_s) =$




θ, θ' determined by s

SELECT + PREPARE

# A food for thought

Question: $O\left(|t| + \log\frac{1}{\varepsilon}\right)$ ← Time-independent

↳ Time-dependent Hamiltonian?

Using the block encoding arithmetics, we can mix these different frameworks.

1. SELECT + PREPARE: Hamiltonian Simulation
2. Purification of Density Matrix: Machine Learning
3. Sparse Matrix: Systems of linear equations

$A_1$: local Hamiltonian

$A_2$: purification of density matrix

$\frac{1}{2}(A_1 + A_2)$

↑ real ↓ imaginary

Maybe a new algorithm can be developed this way?


$U = \begin{pmatrix} e^{-\varepsilon H_A} & \cdot \\ \cdot & \cdot \end{pmatrix}$

$e^{i e^{-\varepsilon H_A} t}$