Toward the first quantum simulation with quantum speedup

Andrew Childs
University of Maryland

Dmitri Maslov

Yunseong Nam

Neil Julien Ross

Yuan Su

arXiv:1711.10980
Simulating Hamiltonian dynamics on a small quantum computer

Andrew Childs
Department of Combinatorics & Optimization
and Institute for Quantum Computing
University of Waterloo

based in part on joint work with
Dominic Berry, Richard Cleve,
Robin Kothari, and Rolando Somma

Workshop on What do we do with a small quantum computer?
IBM Watson, 9 December 2013
Toward practical quantum speedup

Important early goal: demonstrate quantum computational advantage
… but can we find a practical application of near-term devices?

Challenges
• Improve experimental systems
• Improve algorithms and their implementation, making the best use of available hardware

Our goal: Produce concrete resource estimates for the simplest possible practical application of quantum computers
Quantum simulation

“… nature isn’t classical, dammit, and if you want to make a simulation of nature, you’d better make it quantum mechanical, and by golly it’s a wonderful problem, because it doesn’t look so easy.”

Richard Feynman (1981)  
*Simulating physics with computers*

Quantum simulation problem: Given a description of the Hamiltonian $H$, an evolution time $t$, and an initial state $|\psi(0)\rangle$, produce the final state $|\psi(t)\rangle$ (to within some error tolerance $\epsilon$)

A classical computer cannot even represent the state efficiently.

A quantum computer cannot produce a complete description of the state.

But given succinct descriptions of
• the initial state (suitable for a quantum computer to prepare it efficiently) and
• a final measurement (say, measurements of the individual qubits in some basis),
a quantum computer can efficiently answer questions that (apparently) a classical one cannot.
Product formula simulation

Suppose we want to simulate \( H = \sum_{\ell=1}^{L} H_\ell \)

Combine individual simulations with the Lie product formula. E.g., with two terms:

\[
\lim_{r \to \infty} \left( e^{-iAt/r} e^{-iBt/r} \right)^r = e^{-i(A+B)t}
\]

\[
\left( e^{-iAt/r} e^{-iBt/r} \right)^r = e^{-i(A+B)t} + O\left(\frac{t^2}{r}\right)
\]

To ensure error at most \( \epsilon \), take \( r = O\left(\frac{\|H\|t^2}{\epsilon}\right) \)

To get a better approximation, use higher-order formulas.

E.g., second order:

\[
\left( e^{-iAt/2r} e^{-iBt} e^{-iAt/2r} \right)^r = e^{-i(A+B)t} + O\left(\frac{t^3}{r^2}\right)
\]

Systematic expansions to arbitrary order are known [Suzuki 92]

Using the \( 2k \)th order expansion, the number of exponentials required for an approximation with error at most \( \epsilon \) is at most

\[
5^{2k} L^2 \|H\| t \left( \frac{L \|H\| t}{\epsilon} \right)^{1/2k}
\]

[Berry, Ahokas, Cleve, Sanders 07]
Quantum walk simulation

Quantum walk corresponding to $H$

Alternately reflect about $\text{span}\{|\psi_j\rangle\}_{j=1}^N$, $|\psi_j\rangle := |j\rangle \otimes \left( \nu \sum_{k=1}^N \sqrt{H_{jk}^*} |k\rangle + \nu_j |N + 1\rangle \right)$, and swap the two registers.

If $H$ is sparse, this walk is easy to implement.

Spectral theorem: Each eigenvalue $\lambda$ of $H$ corresponds to two eigenvalues $\pm e^{\pm i \arcsin \lambda}$ of the walk operator (with eigenvectors closely related to those of $H$).

Simulation by phase estimation

$$|\lambda\rangle \mapsto |\lambda\rangle |\arcsin \lambda\rangle \quad \text{(phase estimation)}$$

$$\mapsto e^{-i \lambda t} |\lambda\rangle |\arcsin \lambda\rangle \quad \text{(inverse phase est)}$$

Theorem: $O(t/\sqrt{\epsilon})$ steps of this walk suffice to simulate $H$ for time $t$ with error at most $\epsilon$. 

[Childs10; Berry, Childs 12]
Taylor series simulation

Main idea: Directly implement the series

\[
e^{-iHt} = \sum_{k=0}^{\infty} \frac{(-iHt)^k}{k!}
\]

\[
\approx \sum_{k=0}^{K} \frac{(-iHt)^k}{k!}
\]

Write \( H = \sum_{\ell} \alpha_{\ell} H_{\ell} \) with \( H_{\ell} \) unitary.

Then

\[
\sum_{k=0}^{K} \sum_{\ell_1, \ldots, \ell_k} \frac{(-it)^k}{k!} \alpha_{\ell_1} \cdots \alpha_{\ell_k} H_{\ell_1} \cdots H_{\ell_k}
\]

is a linear combination of unitaries.

LCU Lemma: Given the ability to perform unitaries \( V_j \) with unit complexity, one can perform the operation \( U = \sum_j \beta_j V_j \) with complexity \( O\left(\sum_j |\beta_j|\right) \). Furthermore, if \( U \) is (nearly) unitary then this implementation can be made (nearly) deterministic.

Main ideas:

- Implement \( U \) with some amplitude:
  \[
  |0\rangle|\psi\rangle \mapsto \sin \theta |0\rangle U |\psi\rangle + \cos \theta |\Phi\rangle
  \]
- Boost the amplitude for success by oblivious amplitude amplification

Query complexity: \( O\left(t \frac{\log(t/\epsilon)}{\log \log(t/\epsilon)}\right) \)

[Berry, Childs, Cleve, Kothari, Somma 14 & 15]
Quantum signal processing

Combining known lower bounds on the complexity of simulation as a function of $t$ and $\epsilon$ gives

$$\Omega\left(t + \frac{\log \frac{1}{\epsilon}}{\log \log \frac{1}{\epsilon}}\right) \text{ vs. upper bound of } O\left(t \frac{\log t}{\log \log \log t}\right)$$

Recent work, using an alternative method for implementing a linear combination of quantum walk steps, gives an optimal tradeoff.

Main idea: Encode the eigenvalues of $H$ in a two-dimensional subspace; use a carefully-chosen sequence of single-qubit rotations to manipulate those eigenvalues, performing the desired evolution.

[Low, Chuang 16]
## Algorithm comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Query complexity</th>
<th>Gate complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product formula, 1st order</td>
<td>$O\left(d^2 t^2 / \epsilon\right)$</td>
<td>$O\left(d^2 t^2 / \epsilon\right)$</td>
</tr>
<tr>
<td>Product formula, (2k)th order</td>
<td>$O\left(5^{2k} d^3 t \left(\frac{dt}{\epsilon}\right)^{1/2k}\right)$</td>
<td>$O\left(5^{2k} d^3 t \left(\frac{dt}{\epsilon}\right)^{1/2k}\right)$</td>
</tr>
<tr>
<td>Quantum walk</td>
<td>$O\left(dt / \sqrt{\epsilon}\right)$</td>
<td>$O\left(dt / \sqrt{\epsilon}\right)$</td>
</tr>
<tr>
<td>Fractional-query simulation</td>
<td>$O\left(d^2 t \frac{\log(dt/\epsilon)}{\log \log(dt/\epsilon)}\right)$</td>
<td>$O\left(d^2 t \frac{\log^2(dt/\epsilon)}{\log \log(dt/\epsilon)}\right)$</td>
</tr>
<tr>
<td>Taylor series</td>
<td>$O\left(d^2 t \frac{\log(dt/\epsilon)}{\log \log(dt/\epsilon)}\right)$</td>
<td>$O\left(d^2 t \frac{\log^2(dt/\epsilon)}{\log \log(dt/\epsilon)}\right)$</td>
</tr>
<tr>
<td>Linear combination of q. walk steps</td>
<td>$O\left(dt \frac{\log(dt/\epsilon)}{\log \log(dt/\epsilon)}\right)$</td>
<td>$O\left(dt \frac{\log^{3.5}(dt/\epsilon)}{\log \log(dt/\epsilon)}\right)$</td>
</tr>
<tr>
<td>Quantum signal processing</td>
<td>$O\left(dt + \frac{\log(1/\epsilon)}{\log \log(1/\epsilon)}\right)$</td>
<td>$O\left(dt + \frac{\log(1/\epsilon)}{\log \log(1/\epsilon)}\right)$</td>
</tr>
</tbody>
</table>
What to simulate?

Quantum chemistry? Spin systems!

Heisenberg model on a ring: 
\[ H = \sum_{j=1}^{n} (\vec{\sigma}_j \cdot \vec{\sigma}_{j+1} + h_j \sigma^z_j) \] 
\[ h_j \in [-h, h] \text{ uniformly random} \]

This provides a model of self-thermalization and many-body localization.

The transition between thermalized and localized phases (as a function of \( h \)) is poorly understood. Most extensive numerical study: fewer than 25 spins. [Luitz, Laflorence, Alet 15]

Could explore the transition by preparing a simple initial state, evolving, and performing a simple final measurement. Focus on the cost of simulating dynamics.

For concreteness: \( h = 1, \ t = n, \ \epsilon = 10^{-3}, \ 20 \leq n \leq 100 \)
## Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Gate complexity ( (t, \epsilon) )</th>
<th>Gate complexity ( (n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product formula (PF), 1st order</td>
<td>( O(t^2/\epsilon) )</td>
<td>( O(n^5) )</td>
</tr>
<tr>
<td>Product formula (PF), ((2k))th order</td>
<td>( O(5^{2k}t^{1+1/2k}/\epsilon^{1/2k}) )</td>
<td>( O(5^{2k}n^{3+1/k}) )</td>
</tr>
<tr>
<td>Quantum walk</td>
<td>( O(t/\sqrt{\epsilon}) )</td>
<td>( O(n^4 \log n) )</td>
</tr>
<tr>
<td>Fractional-query simulation</td>
<td>( O(t \frac{\log^2(t/\epsilon)}{\log \log(t/\epsilon)}) )</td>
<td>( O(n^4 \frac{\log^2 n}{\log \log n}) )</td>
</tr>
<tr>
<td>Taylor series (TS)</td>
<td>( O(t \frac{\log^2(t/\epsilon)}{\log \log(t/\epsilon)}) )</td>
<td>( O(n^3 \frac{\log^2 n}{\log \log n}) )</td>
</tr>
<tr>
<td>Linear combination of q. walk steps</td>
<td>( O(t \frac{\log^{3.5}(t/\epsilon)}{\log \log(t/\epsilon)}) )</td>
<td>( O(n^4 \frac{\log^2 n}{\log \log n}) )</td>
</tr>
<tr>
<td>Quantum signal processing (QSP)</td>
<td>( O(t + \log(1/\epsilon)) )</td>
<td>( O(n^3) )</td>
</tr>
</tbody>
</table>
## Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Gate complexity ((t, \epsilon))</th>
<th>Gate complexity ((n))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product formula (PF), 1st order</td>
<td>(O(t^2/\epsilon))</td>
<td>(O(n^5))</td>
</tr>
<tr>
<td>Product formula (PF), ((2k))th order</td>
<td>(O(5^{2k}t^{1+1/2k}/\epsilon^{1/2k}))</td>
<td>(O(5^{2k}n^{3+1/k}))</td>
</tr>
<tr>
<td>Quantum walk</td>
<td>(O(t/\sqrt{\epsilon}))</td>
<td>(O(n^4 \log n))</td>
</tr>
<tr>
<td>Fractional-query simulation</td>
<td>(O(t \frac{\log^2 (t/\epsilon)}{\log \log (t/\epsilon)}))</td>
<td>(O(n^4 \frac{\log^2 n}{\log \log n}))</td>
</tr>
<tr>
<td>Taylor series (TS)</td>
<td>(O(t \frac{\log^2 (t/\epsilon)}{\log \log (t/\epsilon)}))</td>
<td>(O(n^3 \frac{\log^2 n}{\log \log n}))</td>
</tr>
<tr>
<td>Linear combination of q. walk steps</td>
<td>(O(t \frac{\log^{3.5} (t/\epsilon)}{\log \log (t/\epsilon)}))</td>
<td>(O(n^4 \frac{\log^2 n}{\log \log n}))</td>
</tr>
<tr>
<td>Quantum signal processing (QSP)</td>
<td>(O(t + \log(1/\epsilon)))</td>
<td>(O(n^3))</td>
</tr>
</tbody>
</table>
Circuit synthesis

We implemented these algorithms using Quipper, a quantum circuit description language that facilitates concrete resource counts.

Gate sets:
• Clifford+$R_z$
• Clifford+$T$

Quipper can produce Clifford+$T$ circuits using recent optimal synthesis algorithms [Kliuchnikov, Maslov, Mosca 13; Ross, Selinger 16].

We verified correctness by simulating subroutines and small instances.

Implementation available at github.com/njross/simcount

We also applied an automated quantum circuit optimizer that we developed [arXiv:1710.07345]. CNOT/$T$ gate counts improve by about 30% for PF. Less significant improvement for TS/QSP.
Product formula implementation

We consider four methods for choosing the parameters of the PF algorithm:

**Analytic** and **Minimized** error bounds: tightened versions of previous analysis

**Commutator** error bound: exploit commutation relations among terms in the Hamiltonian

Involves extensive analysis to derive the bound and compute it for our model system

Improved asymptotic performance:

\[ O(n^{3+1/k}) \rightarrow O(n^{3+2/(2k+1)}) \]

**Empirical** error bound: extrapolate performance from small instances
## Taylor series implementation

Main implementation issue: construct circuits for the operation $\text{select}(V) = \sum_{j=1}^{\Gamma} |j\rangle\langle j| \otimes V_j$

We construct an optimized walk on a binary tree that encodes the control into a single qubit, saving a factor of about $\log \Gamma$ (between 5 and 9 in our instances).

Also give concrete error analysis. Empirical error bounds are infeasible but probably not helpful.
Quantum signal processing implementation

QSP is built from the same basic subroutines as TS (state preparation, reflection, select(V)).

To compute a sequence of rotation angles that define the algorithm, we must find the roots of a high-degree polynomial to high precision. This can be done in polynomial time (classically), but it’s expensive in practice.

Workarounds:
- Compute the gate count using placeholder angles
- Consider a segmented version of the algorithm: concatenate segments that are short enough to be simulated. Modest overhead: with $M$ angles, $O(n^{3+4/M})$ vs. $O(n^3)$ for full QSP.

Empirical error bounds:
- Empirical estimate of the error in the Jacobi-Anger expansion saves about 30-45%.
- Comprehensive empirical error bounds are just barely feasible and probably not helpful.
Product formula comparisons

<table>
<thead>
<tr>
<th>Bound</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytic/Minimized</td>
<td>5</td>
<td>4</td>
<td>3.5</td>
<td>3.333</td>
<td>3.25</td>
</tr>
<tr>
<td>Commutator</td>
<td>4</td>
<td>3.667</td>
<td>3.4</td>
<td>3.286</td>
<td>3.222</td>
</tr>
<tr>
<td>Empirical</td>
<td>2.964</td>
<td>2.883</td>
<td>2.555</td>
<td>2.311</td>
<td>2.141</td>
</tr>
</tbody>
</table>
Resource estimates (physical level)

The diagrams show resource estimates at the physical level, including CNOT gate counts and qubit requirements as functions of system size.

- **CNOT gate count (Clifford+$R_z$)**: The graph on the left illustrates the CNOT gate count for different methods as the system size increases. The methods include PF (com 4), PF (emp), TS, QSP (seg), and QSP (JA emp).

- **Qubits**: The graph on the right displays the number of qubits required for a given system size. The methods are denoted by different markers and are color-coded for clarity.
Resource estimates (logical level)

- **System size** vs. **Qubits**
- **T gate count (Clifford+T)** vs. **System size**

- PF (com 4)
- PF (emp)
- TS
- QSP (seg)
- QSP (JA emp)
Comparisons

Factoring a 1024-bit number [Kutin 06]
- 3132 qubits
- $5.7 \times 10^9$ $T$ gates

Simulating FeMoco [Reiher et al. 16]
- 111 qubits
- $1.0 \times 10^{14}$ $T$ gates

Simulating 50 spins (segmented QSP)
- 67 qubits
- $2.4 \times 10^9$ $T$ gates

Simulating 50 spins (PF6 empirical)
- 50 qubits
- $1.8 \times 10^8$ $T$ gates
Summary

This work establishes benchmarks for a simple quantum simulation that would be useful and that is classically hard.

Spin systems are much easier than factoring or quantum chemistry… … but may still be out of reach of pre-fault tolerant digital quantum computers.

Higher-order product formulas are useful even at very small sizes.

Existing analysis of product formulas is very loose.

More sophisticated algorithms (especially quantum signal processing) are competitive at surprisingly small sizes and give the best approach with rigorous guarantees.
Outlook

Super-classical quantum simulation without invoking fault tolerance?
• Improved error bounds
• Optimized implementations
• Alternative target systems
• New simulation algorithms
• Experiments!

Resource estimates for more practical models
• Architectural constraints, parallelism
• Fault-tolerant implementations

Better provable bounds for simulation algorithms
• Product formula error bounds beyond the triangle inequality
• Efficient synthesis of the QSP circuit