Early Fault-Tolerant Quantum Algorithms for Matrix Functions via Trotter Extrapolation

Arul Rhik Mazumder
Co-mentor: Samson Wang
Mentor: John Preskill

August 25, 2025

What is Early Fault Tolerance (EFT)?

- **Bridging the Gap**: EFT is the era transitioning from **NISQ** (noisy, intermediate-scale quantum) to **FTQC** (fully fault-tolerant quantum).

What is Early Fault Tolerance (EFT)?

- **Bridging the Gap**: EFT is the era transitioning from **NISQ** (noisy, intermediate-scale quantum) to **FTQC** (fully fault-tolerant quantum).
- **Practical Goal**: Achieve practical quantum advantage **despite hardware limitations** (limited qubits, circuit depth, and error).

What is Early Fault Tolerance (EFT)?

- **Bridging the Gap**: EFT is the era transitioning from **NISQ** (noisy, intermediate-scale quantum) to **FTQC** (fully fault-tolerant quantum).
- **Practical Goal**: Achieve practical quantum advantage **despite hardware limitations** (limited qubits, circuit depth, and error).
- **Key Focus**: Develop algorithms with provable performance that are **resource-efficient** to be viable on existing quantum hardware.

EFT for Chemistry Problems

- **The Computational Challenge:** Simulating quantum system is **intractable** for classical computers beyond few particles.

EFT for Chemistry Problems

- **The Computational Challenge:** Simulating quantum system is **intractable** for classical computers beyond few particles.
- **Quantities of Interest:** We want to use quantum computers to estimate quantities of the form $\mathrm{Tr}[\rho f(A)]$ and $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$

EFT for Chemistry Problems

- **The Computational Challenge:** Simulating quantum system is **intractable** for classical computers beyond few particles.
- **Quantities of Interest:** We want to use quantum computers to estimate quantities of the form $\mathrm{Tr}[\rho f(A)]$ and $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$
  - $f(A) = A^{-1} \rightarrow$ solve linear systems and estimate Green's function

EFT for Chemistry Problems

- **The Computational Challenge:** Simulating quantum system is **intractable** for classical computers beyond few particles.
- **Quantities of Interest:** We want to use quantum computers to estimate quantities of the form $\mathrm{Tr}[\rho f(A)]$ and $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$
  - $f(A) = A^{-1} \rightarrow$ solve linear systems and estimate Green's function
  - $f(A) = \Theta_x(A) \rightarrow$ find the ground state energy using Phase Estimation.

Key EFT Algorithmic Examples

- *Lin and Tong* [1] (QPE): Achieves Heisenberg-limited precision with **one ancilla qubit** and heavy **classical post-processing**.

Key EFT Algorithmic Examples

- *Lin and Tong* [1] (QPE): Achieves Heisenberg-limited precision with **one ancilla qubit** and heavy **classical post-processing**.
- *Wan et al.* [2] (QPE): Uses **randomized techniques** for **one ancilla qubit** and **shorter circuits**, processing multiple runs statistically.

Key EFT Algorithmic Examples

- *Lin and Tong* [1] (QPE): Achieves Heisenberg-limited precision with **one ancilla qubit** and heavy **classical post-processing**.
- *Wan et al.* [2] (QPE): Uses **randomized techniques** for **one ancilla qubit** and **shorter circuits**, processing multiple runs statistically.
- *Wang et al.* [3] (Linear Algebra): Employs **qubit-efficient** and **randomized methods** to **reduce logical qubit requirements**.

Key EFT Algorithmic Examples

- *Lin and Tong* [1] (QPE): Achieves Heisenberg-limited precision with **one ancilla qubit** and heavy **classical post-processing**.
- *Wan et al.* [2] (QPE): Uses **randomized techniques** for **one ancilla qubit** and **shorter circuits**, processing multiple runs statistically.
- *Wang et al.* [3] (Linear Algebra): Employs **qubit-efficient** and **randomized methods** to **reduce logical qubit requirements**.

A common theme is that, as shown by [1]–[3], we can estimate target quantities using fewer quantum resources (qubits and gates) by increasing classical post-processing or runtime.

Key EFT Algorithmic Examples

- *Lin and Tong* [1] (QPE): Achieves Heisenberg-limited precision with **one ancilla qubit** and heavy **classical post-processing**.
- *Wan et al.* [2] (QPE): Uses **randomized techniques** for **one ancilla qubit** and **shorter circuits**, processing multiple runs statistically.
- *Wang et al.* [3] (Linear Algebra): Employs **qubit-efficient** and **randomized methods** to **reduce logical qubit requirements**.

A common theme is that, as shown by [1]–[3], we can estimate target quantities using fewer quantum resources (qubits and gates) by increasing classical post-processing or runtime.

All these techniques use $\mathrm{Tr}[Ze^{iHT}]$ which we estimate with our algorithm.

Background

Quantum Time Evolution: Fundamentals & Challenges

- **Quantum Time Evolution (e^{-iHt}):** A core building block for many quantum algorithms, including QPE, HHL, and general simulation.

Quantum Time Evolution: Fundamentals & Challenges

- **Quantum Time Evolution (e^{-iHt}):** A core building block for many quantum algorithms, including QPE, HHL, and general simulation.
- **Trotterization (Product Formulas):**

Quantum Time Evolution: Fundamentals & Challenges

- **Quantum Time Evolution (e^{-iHt}):** A core building block for many quantum algorithms, including QPE, HHL, and general simulation.
- **Trotterization (Product Formulas):**
  - Approximates continuous time evolution e^{-iHt} by decomposing the Hamiltonian $H = \sum H_j$ into simple, implementable terms.

Quantum Time Evolution: Fundamentals & Challenges

- **Quantum Time Evolution (e^{-iHt}):** A core building block for many quantum algorithms, including QPE, HHL, and general simulation.
- **Trotterization (Product Formulas):**
  - Approximates continuous time evolution e^{-iHt} by decomposing the Hamiltonian $H = \sum H_j$ into simple, implementable terms.
  - Example: 1st-order Trotter $e^{(A+B)t} \approx \left(e^{At/n} e^{Bt/n}\right)^n + \mathcal{O}(t^2/n)$.

Quantum Time Evolution: Fundamentals & Challenges

- **Quantum Time Evolution (e^{-iHt}):** A core building block for many quantum algorithms, including QPE, HHL, and general simulation.
- **Trotterization (Product Formulas):**
  - Approximates continuous time evolution e^{-iHt} by decomposing the Hamiltonian $H = \sum H_j$ into simple, implementable terms.
  - Example: 1st-order Trotter $e^{(A+B)t} \approx \left(e^{At/n}e^{Bt/n}\right)^n + \mathcal{O}(t^2/n)$.
  - Implemented using native quantum gates for each $e^{-iH_j t/n}$ term.

Quantum Time Evolution: Fundamentals & Challenges

- **Quantum Time Evolution (e^{-iHt}):** A core building block for many quantum algorithms, including QPE, HHL, and general simulation.
- **Trotterization (Product Formulas):**
  - Approximates continuous time evolution e^{-iHt} by decomposing the Hamiltonian $H = \sum H_j$ into simple, implementable terms.
  - Example: 1st-order Trotter $e^{(A+B)t} \approx \left(e^{At/n}e^{Bt/n}\right)^n + \mathcal{O}(t^2/n)$.
  - Implemented using native quantum gates for each $e^{-iH_j t/n}$ term.
- **The Precision Challenge:**
  - Standard Trotterization requires a high number of steps (n) for high precision (ε), with error scaling poorly:

$$\text{Steps} \sim \mathcal{O}(1/\varepsilon) \text{ (1st-order)}, \quad \sim \mathcal{O}(\varepsilon^{-1/p}) \text{ (order } p\text{)}$$

Quantum Time Evolution: Fundamentals & Challenges

- **Quantum Time Evolution (e^{-iHt}):** A core building block for many quantum algorithms, including QPE, HHL, and general simulation.
- **Trotterization (Product Formulas):**
  - Approximates continuous time evolution e^{-iHt} by decomposing the Hamiltonian $H = \sum H_j$ into simple, implementable terms.
  - Example: 1st-order Trotter $e^{(A+B)t} \approx \left(e^{At/n}e^{Bt/n}\right)^n + \mathcal{O}(t^2/n)$.
  - Implemented using native quantum gates for each $e^{-iH_j t/n}$ term.
- **The Precision Challenge:**
  - Standard Trotterization requires a high number of steps (n) for high precision (ε), with error scaling poorly:

$$\text{Steps} \sim \mathcal{O}(1/\varepsilon) \text{ (1st-order)}, \quad \sim \mathcal{O}(\varepsilon^{-1/p}) \text{ (order } p\text{)}$$

  - This leads to deep circuits, limiting applicability on near-term hardware.

Why Trotter for Early Fault Tolerance?

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$

Why Trotter for Early Fault Tolerance?

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p}\varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$

- **Low qubit overhead:** Requires no ancillas or block-encoding circuits.

Why Trotter for Early Fault Tolerance?

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$

- **Low qubit overhead:** Requires no ancillas or block-encoding circuits.
- **Simple to compile:** Operators decompose naturally into native gates.

Why Trotter for Early Fault Tolerance?

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$

- **Low qubit overhead:** Requires no ancillas or block-encoding circuits.
- **Simple to compile:** Operators decompose naturally into native gates.
- **Sub-quadratic time complexity** compared to Random Compiler

Why Trotter for Early Fault Tolerance?

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$

- **Low qubit overhead:** Requires no ancillas or block-encoding circuits.
- **Simple to compile:** Operators decompose naturally into native gates.
- **Sub-quadratic time complexity** compared to Random Compiler
- **Commutator scaling:** Errors scale with nested commutators, which are often small in realistic systems. Performs substantially better when $\lambda_{\text{comm}} << \|H\|_1$

Our Algorithm and Results

Applying Richardson Extrapolation on Trotter Formulas

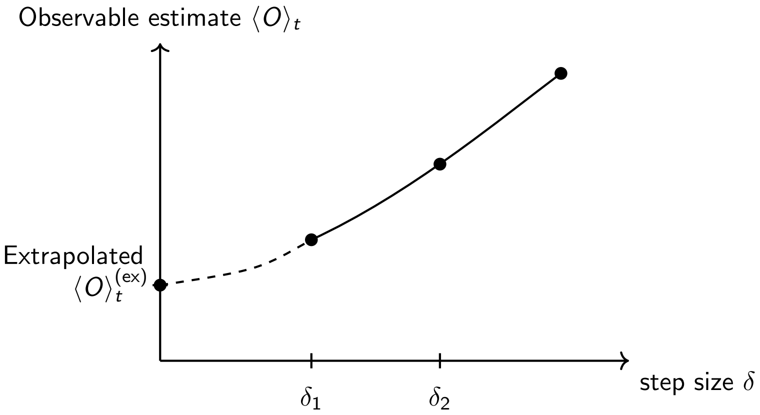
- **Classical Extrapolation for Precision:**

Applying Richardson Extrapolation on Trotter Formulas

- **Classical Extrapolation for Precision:**
  - **Goal:** Improve error scaling without increasing quantum circuit depth.

Applying Richardson Extrapolation on Trotter Formulas

- **Classical Extrapolation for Precision:**
  - **Goal:** Improve error scaling without increasing quantum circuit depth.
  - Perform quantum simulations at multiple Trotter step sizes and then **classically extrapolate** these results to for true evolution ($\delta \to 0$).



Comparisons with our Algorithm

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p}\varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Comparisons with our Algorithm

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p}\varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

- **Exponential improvement** on $\varepsilon^{-\frac{1}{p}}$ scaling compared Trotter.

Comparisons with our Algorithm

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p}) T^{1+1/p}\varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

- **Exponential improvement** on $\varepsilon^{-\frac{1}{p}}$ scaling compared Trotter.
- **Efficient postprocessing:** Advantage achieved purely classically.

Comparisons with our Algorithm

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p}\varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

- **Exponential improvement** on $\varepsilon^{-\frac{1}{p}}$ scaling compared Trotter.
- **Efficient postprocessing:** Advantage achieved purely classically.
- **Hardware-friendly:** Well-suited to NISQ-era devices — shorter circuits + more measurements. Achieves comparable performance to asymptotically better but more resource efficient schemes.

Comparisons with our Algorithm

Method	Max Depth / Sample	Sample Overhead
Qubitization [4]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [5]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [2]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

- **Exponential improvement** on $\varepsilon^{-\frac{1}{p}}$ scaling compared Trotter.
- **Efficient postprocessing:** Advantage achieved purely classically.
- **Hardware-friendly:** Well-suited to NISQ-era devices — shorter circuits + more measurements. Achieves comparable performance to asymptotically better but more resource efficient schemes.
- **Key Question**: Can this be applied to matrix functions?

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

$$\mathrm{Tr}[f(A)\rho f(A)^\dagger O] \text{ and } \mathrm{Tr}[f(A)\rho]$$

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

$$\mathrm{Tr}[f(A)\rho f(A)^\dagger O] \text{ and } \mathrm{Tr}[f(A)\rho]$$

- Occurs in quantum algorithms (HHL, QPE, etc.).

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

$$\mathrm{Tr}[f(A)\rho f(A)^\dagger O] \text{ and } \mathrm{Tr}[f(A)\rho]$$

- Occurs in quantum algorithms (HHL, QPE, etc.).
- Standard Richardson+Trotter methods only apply to $f(A) = e^{-iAt}$ [6]

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

$$\mathrm{Tr}[f(A)\rho f(A)^\dagger O] \text{ and } \mathrm{Tr}[f(A)\rho]$$

- Occurs in quantum algorithms (HHL, QPE, etc.).
- Standard Richardson+Trotter methods only apply to $f(A) = e^{-iAt}$ [6]
- Represent $f(A)$ via Fourier series to reduce to exponentials.

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

$$\mathrm{Tr}[f(A)\rho\, f(A)^\dagger O] \text{ and } \mathrm{Tr}[f(A)\rho]$$

- Occurs in quantum algorithms (HHL, QPE, etc.).
- Standard Richardson+Trotter methods only apply to $f(A) = e^{-iAt}$ [6]
- Represent $f(A)$ via Fourier series to reduce to exponentials.

The key extension: prove Richardson extrapolation works for:

$$\mathrm{Tr}[e^{iHt_1}\rho\, e^{-iHt_2} O]$$

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

$$\mathrm{Tr}[f(A)\rho f(A)^\dagger O] \text{ and } \mathrm{Tr}[f(A)\rho]$$

- Occurs in quantum algorithms (HHL, QPE, etc.).
- Standard Richardson+Trotter methods only apply to $f(A) = e^{-iAt}$ [6]
- Represent $f(A)$ via Fourier series to reduce to exponentials.

The key extension: prove Richardson extrapolation works for:

$$\mathrm{Tr}[e^{iHt_1}\rho\, e^{-iHt_2}O]$$

As an intermediate step, we develop and prove algorithms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{m} c_k \mathrm{Tr}[Ze^{iAt_k}]$$

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

$$\mathrm{Tr}[f(A)\rho f(A)^\dagger O] \text{ and } \mathrm{Tr}[f(A)\rho]$$

- Occurs in quantum algorithms (HHL, QPE, etc.).
- Standard Richardson+Trotter methods only apply to $f(A) = e^{-iAt}$ [6]
- Represent $f(A)$ via Fourier series to reduce to exponentials.

The key extension: prove Richardson extrapolation works for:

$$\mathrm{Tr}[e^{iHt_1}\rho\, e^{-iHt_2} O]$$

As an intermediate step, we develop and prove algorithms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{m} c_k \mathrm{Tr}[Ze^{iAt_k}]$$

We have it when $Z = \rho$, and extend to when $\|Z\|_1$ is bounded.

Richardson Extrapolation for each $\mathrm{Tr}[Ze^{iAt_k}]$

For each t_k, estimate $\mathrm{Tr}[Ze^{iAt_k}]$ via Richardson-extrapolated circuits.

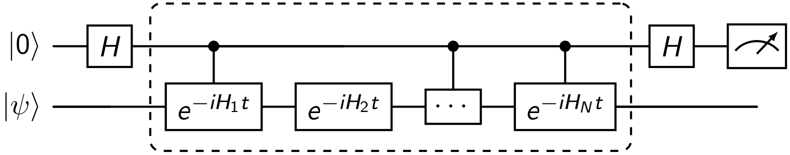
$$\mathrm{Tr}[Ze^{iAt_k}] = \sum_{j=1}^{m} b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

Richardson Extrapolation for each $\mathrm{Tr}[Ze^{iAt_k}]$

For each t_k, estimate $\mathrm{Tr}[Ze^{iAt_k}]$ via Richardson-extrapolated circuits.

$$\mathrm{Tr}[Ze^{iAt_k}] = \sum_{j=1}^{m} b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

Each $\mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$ is estimated by sampling from Hadamard circuits like the one below:



repeat $r_j := 1/s_j$ times

Algorithm for Estimating $\mathrm{Tr}[Zf(A)]$

Algorithm:

Algorithm for Estimating $\mathrm{Tr}[Zf(A)]$

Algorithm:

1. Express trace as double sum over Fourier and Richardson terms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{K} \sum_{j=1}^{m} c_k b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

Algorithm for Estimating $\mathrm{Tr}[Zf(A)]$

Algorithm:

1. Express trace as double sum over Fourier and Richardson terms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{K} \sum_{j=1}^{m} c_k b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

2. Sample pair (k, j) with probability $\frac{|c_k b_j|}{\mathcal{Z}}$, where $\mathcal{Z} = \sum_{k,j} |c_k b_j|$

Algorithm for Estimating $\mathrm{Tr}[Zf(A)]$

Algorithm:

1. Express trace as double sum over Fourier and Richardson terms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{K} \sum_{j=1}^{m} c_k b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

2. Sample pair (k, j) with probability $\frac{|c_k b_j|}{\mathcal{Z}}$, where $\mathcal{Z} = \sum_{k,j} |c_k b_j|$

3. Estimate $\mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$ via Hadamard tests

Algorithm for Estimating $\mathrm{Tr}[Zf(A)]$

Algorithm:

1. Express trace as double sum over Fourier and Richardson terms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{K} \sum_{j=1}^{m} c_k b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

2. Sample pair (k, j) with probability $\frac{|c_k b_j|}{\mathcal{Z}}$, where $\mathcal{Z} = \sum_{k,j} |c_k b_j|$
3. Estimate $\mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$ via Hadamard tests
4. Return scaled, signed estimator based on samples

Algorithm for Estimating $\mathrm{Tr}[Zf(A)]$

Algorithm:

1. Express trace as double sum over Fourier and Richardson terms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{K} \sum_{j=1}^{m} c_k b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

2. Sample pair (k, j) with probability $\frac{|c_k b_j|}{\mathcal{Z}}$, where $\mathcal{Z} = \sum_{k,j} |c_k b_j|$

3. Estimate $\mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$ via Hadamard tests

4. Return scaled, signed estimator based on samples

Gate complexity (per sample) $\mathcal{O}\left(\Gamma \log(c/\varepsilon) \cdot (a_{\max} \Upsilon \lambda_{\mathrm{comm}} t_{\max})^{1+\frac{1}{p}}\right)$,

Algorithm for Estimating $\mathrm{Tr}[Zf(A)]$

Algorithm:

1. Express trace as double sum over Fourier and Richardson terms:

$$\mathrm{Tr}[Zf(A)] = \sum_{k=1}^{K}\sum_{j=1}^{m} c_k b_j \mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$$

2. Sample pair (k, j) with probability $\frac{|c_k b_j|}{\mathcal{Z}}$, where $\mathcal{Z} = \sum_{k,j} |c_k b_j|$

3. Estimate $\mathrm{Tr}[Z\mathcal{P}^{1/s_j}(s_j t_k)]$ via Hadamard tests

4. Return scaled, signed estimator based on samples

Gate complexity (per sample) $\mathcal{O}\left(\Gamma \log(c/\varepsilon) \cdot (a_{\max} \Upsilon \lambda_{\mathrm{comm}} t_{\max})^{1+\frac{1}{p}}\right)$,

Sample complexity $\mathcal{O}\left(\frac{\|Z\|_1^2 c^2 (\log\log(1/\varepsilon))^2}{\varepsilon^2} \cdot \log\left(\frac{1}{\delta}\right)\right)$

Key Algorithmic Techniques

1. **Fourier Expansion**: to represent $\mathrm{Tr}[Zf(A)]$ term as a sum of $\mathrm{Tr}[Ze^{iHT}]$ terms.

Key Algorithmic Techniques

1. **Fourier Expansion**: to represent $\mathrm{Tr}[Zf(A)]$ term as a sum of $\mathrm{Tr}[Ze^{iHT}]$ terms.

2. **Error Series Representation**:

$$\mathrm{Tr}\left[Z\,\mathcal{P}^{1/s}(sT)\right] = \mathrm{Tr}\left[Z\,e^{iAT}\right] + \sum_{j\in\sigma\mathbb{Z}_+\geq p} s^j\,\mathrm{Tr}[Z\,\tilde{E}_{j+1,K}(T)] + \mathrm{Tr}[Z\,\tilde{F}_K(T,s)]$$

allows to use and analyze Richardson extrapolation (to get coefficients $\{b_j\}_{j=1}^m$ and schedule $\{s_j\}_{j=1}^m$) for improved gate complexity.

Key Algorithmic Techniques

1. **Fourier Expansion**: to represent $\mathrm{Tr}[Zf(A)]$ term as a sum of $\mathrm{Tr}[Ze^{iHT}]$ terms.

2. **Error Series Representation**:

$$\mathrm{Tr}\left[Z\,\mathcal{P}^{1/s}(sT)\right] = \mathrm{Tr}\left[Z\,e^{iAT}\right] + \sum_{j\in\sigma\mathbb{Z}_+\geq p} s^j \mathrm{Tr}[Z\,\tilde{E}_{j+1,K}(T)] + \mathrm{Tr}[Z\,\tilde{F}_K(T,s)]$$

   allows to use and analyze Richardson extrapolation (to get coefficients $\{b_j\}_{j=1}^m$ and schedule $\{s_j\}_{j=1}^m$) for improved gate complexity.

3. **Randomization**: improves sample overhead for circuits

Applications

Ground Energy Estimation via QPE ($f(A) = \Theta_x(A)$)

Goal: Estimate the ground energy E_0 of a Hamiltonian H.

Ground Energy Estimation via QPE ($f(A) = \Theta_x(A)$)

Goal: Estimate the ground energy E_0 of a Hamiltonian H.

1. **Approximate CDF:** Construct the approximate CDF:

$$\widetilde{C}(x) = \mathrm{Tr}[\rho\,\widetilde{\Theta}(xI - \kappa H)]$$

where $\widetilde{\Theta}$ is a quantum filter approximating the Heaviside function.

Ground Energy Estimation via QPE ($f(A) = \Theta_x(A)$)

Goal: Estimate the ground energy E_0 of a Hamiltonian H.

1. **Approximate CDF:** Construct the approximate CDF:

$$\widetilde{C}(x) = \mathrm{Tr}[\rho\,\widetilde{\Theta}(xI - \kappa H)]$$

   where $\widetilde{\Theta}$ is a quantum filter approximating the Heaviside function.

2. **Ground Energy Estimation:** By examining the behavior of $\widetilde{C}(x \pm u)$, we can identify a value x^* that approximates the ground energy E_0.

Ground Energy Estimation via QPE ($f(A) = \Theta_x(A)$)

Goal: Estimate the ground energy E_0 of a Hamiltonian H.

1. **Approximate CDF:** Construct the approximate CDF:

$$\widetilde{C}(x) = \mathrm{Tr}[\rho \widetilde{\Theta}(xI - \kappa H)]$$

   where $\widetilde{\Theta}$ is a quantum filter approximating the Heaviside function.

2. **Ground Energy Estimation:** By examining the behavior of $\widetilde{C}(x \pm u)$, we can identify a value x^* that approximates the ground energy E_0.

3. **Binary Search:** A binary search is used to efficiently find E_0, requiring $\mathcal{O}(\log(1/u))$ evaluations of $\widetilde{C}(x)$.

Ground Energy Estimation via QPE ($f(A) = \Theta_x(A)$)

Goal: Estimate the ground energy E_0 of a Hamiltonian H.

1. **Approximate CDF:** Construct the approximate CDF:

$$\widetilde{C}(x) = \mathrm{Tr}[\rho\,\widetilde{\Theta}(xI - \kappa H)]$$

where $\widetilde{\Theta}$ is a quantum filter approximating the Heaviside function.

2. **Ground Energy Estimation:** By examining the behavior of $\widetilde{C}(x \pm u)$, we can identify a value x^* that approximates the ground energy E_0.

3. **Binary Search:** A binary search is used to efficiently find E_0, requiring $\mathcal{O}(\log(1/u))$ evaluations of $\widetilde{C}(x)$.

Resource Requirements: To estimate E_0 to precision ε:

Ground Energy Estimation via QPE ($f(A) = \Theta_x(A)$)

Goal: Estimate the ground energy E_0 of a Hamiltonian H.

1. **Approximate CDF:** Construct the approximate CDF:

$$\widetilde{C}(x) = \mathrm{Tr}[\rho\,\widetilde{\Theta}(xI - \kappa H)]$$

where $\widetilde{\Theta}$ is a quantum filter approximating the Heaviside function.

2. **Ground Energy Estimation:** By examining the behavior of $\widetilde{C}(x \pm u)$, we can identify a value x^* that approximates the ground energy E_0.

3. **Binary Search:** A binary search is used to efficiently find E_0, requiring $\mathcal{O}(\log(1/u))$ evaluations of $\widetilde{C}(x)$.

Resource Requirements: To estimate E_0 to precision ε:

- **Gate Cost:** $C_{\text{gate}} = \widetilde{\mathcal{O}}\left(\Gamma\left(\frac{\Upsilon\lambda_{\text{comm}}}{\varepsilon}\right)^{1+\frac{1}{p}}\right)$

Ground Energy Estimation via QPE ($f(A) = \Theta_x(A)$)

Goal: Estimate the ground energy E_0 of a Hamiltonian H.

1. **Approximate CDF:** Construct the approximate CDF:

$$\widetilde{C}(x) = \mathrm{Tr}[\rho\, \widetilde{\Theta}(xI - \kappa H)]$$

   where $\widetilde{\Theta}$ is a quantum filter approximating the Heaviside function.

2. **Ground Energy Estimation:** By examining the behavior of $\widetilde{C}(x \pm u)$, we can identify a value x^* that approximates the ground energy E_0.

3. **Binary Search:** A binary search is used to efficiently find E_0, requiring $\mathcal{O}(\log(1/u))$ evaluations of $\widetilde{C}(x)$.

Resource Requirements: To estimate E_0 to precision ε:

- **Gate Cost:** $C_{\text{gate}} = \widetilde{\mathcal{O}}\left(\Gamma \left(\frac{\Upsilon \lambda_{\text{comm}}}{\varepsilon}\right)^{1+\frac{1}{p}}\right)$
- **Sample Cost:** $C_{\text{sample}} = \widetilde{\mathcal{O}}\left(\frac{1}{\eta^2}\right)$

Estimating Green's Function ($f(A) = A^{-1}$)

Goal: Compute the Green's function, which describes how a quantum system responds to perturbations. This is crucial for calculating spectral properties like excitation energies.

Estimating Green's Function ($f(A) = A^{-1}$)

Goal: Compute the Green's function, which describes how a quantum system responds to perturbations. This is crucial for calculating spectral properties like excitation energies.

Method:

- We estimate the **resolvent operator**:

$$R(\omega + i\Gamma_{\text{broad}}, \hat{H}) = (\omega + i\Gamma_{\text{broad}} - \hat{H})^{-1}$$

Estimating Green's Function ($f(A) = A^{-1}$)

Goal: Compute the Green's function, which describes how a quantum system responds to perturbations. This is crucial for calculating spectral properties like excitation energies.

Method:

- We estimate the **resolvent operator**:

$$R(\omega + i\Gamma_{\text{broad}}, \hat{H}) = (\omega + i\Gamma_{\text{broad}} - \hat{H})^{-1}$$

- A positive **broadening factor** ($\Gamma_{\text{broad}} > 0$) is included to ensure the calculation converges.

Estimating Green's Function ($f(A) = A^{-1}$)

Goal: Compute the Green's function, which describes how a quantum system responds to perturbations. This is crucial for calculating spectral properties like excitation energies.

Method:

- We estimate the **resolvent operator**:

$$R(\omega + i\Gamma_{\text{broad}}, \hat{H}) = (\omega + i\Gamma_{\text{broad}} - \hat{H})^{-1}$$

- A positive **broadening factor** ($\Gamma_{\text{broad}} > 0$) is included to ensure the calculation converges.

Resource Requirements: Compute the Green's function to precision ε:

Estimating Green's Function ($f(A) = A^{-1}$)

Goal: Compute the Green's function, which describes how a quantum system responds to perturbations. This is crucial for calculating spectral properties like excitation energies.

Method:

- We estimate the **resolvent operator**:

$$R(\omega + i\Gamma_{\text{broad}}, \hat{H}) = (\omega + i\Gamma_{\text{broad}} - \hat{H})^{-1}$$

- A positive **broadening factor** ($\Gamma_{\text{broad}} > 0$) is included to ensure the calculation converges.

Resource Requirements: Compute the Green's function to precision ε:

- **Gate Cost:** $C_{\text{gate}} = \widetilde{\mathcal{O}}\left(\Gamma\left(a_{\max}\Upsilon\lambda_{\text{comm}}\frac{1}{\Gamma_{\text{broad}}}\right)^{1+\frac{1}{p}}\right)$

Estimating Green's Function ($f(A) = A^{-1}$)

Goal: Compute the Green's function, which describes how a quantum system responds to perturbations. This is crucial for calculating spectral properties like excitation energies.

Method:

- We estimate the **resolvent operator**:

$$R(\omega + i\Gamma_{\text{broad}}, \hat{H}) = (\omega + i\Gamma_{\text{broad}} - \hat{H})^{-1}$$

- A positive **broadening factor** ($\Gamma_{\text{broad}} > 0$) is included to ensure the calculation converges.

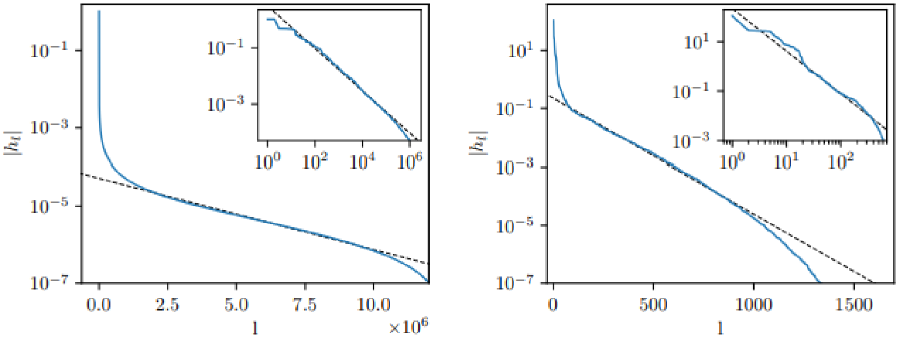
Resource Requirements: Compute the Green's function to precision ε:

- **Gate Cost:** $C_{\text{gate}} = \widetilde{\mathcal{O}}\left(\Gamma\left(a_{\max}\Upsilon\lambda_{\text{comm}}\frac{1}{\Gamma_{\text{broad}}}\right)^{1+\frac{1}{p}}\right)$
- **Sample Cost:** $C_{\text{sample}} = \widetilde{\mathcal{O}}\left(\frac{1}{\Gamma_{\text{broad}}^2\,\varepsilon^2}\right)$

Refinements

Partial Randomization Motivation

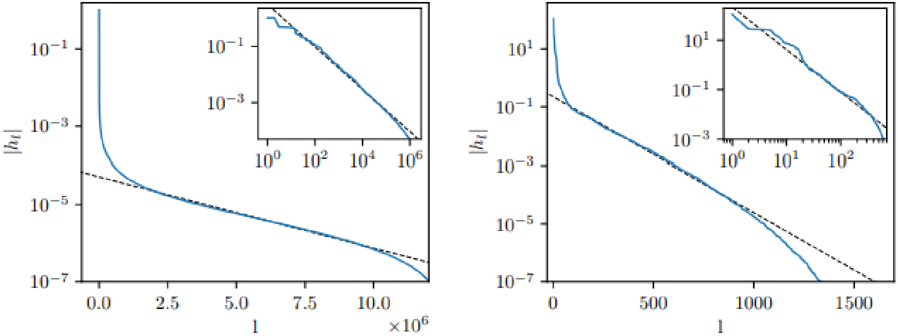
Many Hamiltonians have relatively few high-weight terms. [1]



[1] Image from J. Günther, F. Witteveen, A. Schmidhuber, M. Miller, M. Christandl, and A. Harrow [7]

Partial Randomization Motivation

Many Hamiltonians have relatively few high-weight terms. [1]

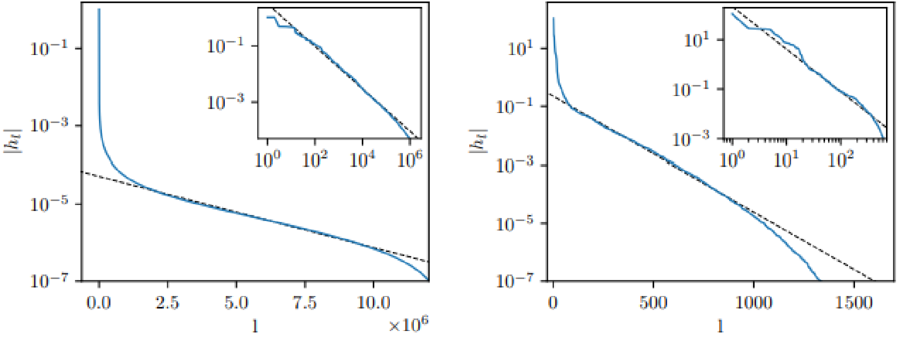


- The main plot shows exponential fit to the tail. ($|h_\ell| \approx Ae^{-b\ell}$)

[1]Image from J. Günther, F. Witteveen, A. Schmidhuber, M. Miller, M. Christandl, and A. Harrow [7]

Partial Randomization Motivation

Many Hamiltonians have relatively few high-weight terms. [1]

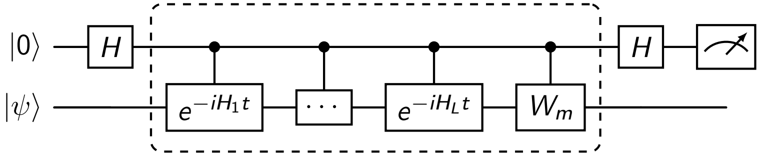


- The main plot shows exponential fit to the tail. ($|h_\ell| \approx Ae^{-b\ell}$)
- The insert shows power law fit for the large terms. ($|h_\ell| \approx C \cdot \ell^{-\alpha}$)

[1] Image from J. Günther, F. Witteveen, A. Schmidhuber, M. Miller, M. Christandl, and A. Harrow [7]

Richardson Extrapolation on Partial Randomization

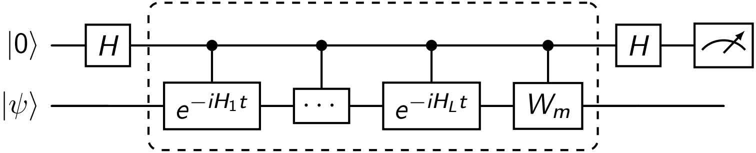
We apply product formulas on L high weight terms and randomize the rest:



repeat r times

Richardson Extrapolation on Partial Randomization

We apply product formulas on L high weight terms and randomize the rest:

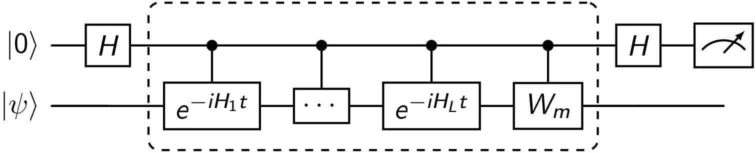


repeat r times

To implement each W_m we use the Randomized Taylor Expansion [2].

Richardson Extrapolation on Partial Randomization

We apply product formulas on L high weight terms and randomize the rest:



repeat r times

To implement each W_m we use the Randomized Taylor Expansion [2].
We apply Richardson-extrapolation on the partially random circuits for:

$$C_{\text{gate}} = \mathcal{O}\left(L_D(\Upsilon\lambda_{\text{comm}}T)^{1+\frac{1}{p}}\log^2\left(\frac{1}{\varepsilon}\right) + \lambda_R^2 T^2\right)$$

$$C_{\text{sample}} = \mathcal{O}\left(\frac{1}{\varepsilon^2}\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^2\right)$$

Fermion Systems

For systems in an η-fermion subspace, we tighten analysis with the **fermionic semi-norm**.

Fermion Systems

For systems in an η-fermion subspace, we tighten analysis with the **fermionic semi-norm**.

- Operators are **number-preserving** (map η-electron states to η-electron states).

Fermion Systems

For systems in an η-fermion subspace, we tighten analysis with the **fermionic semi-norm**.

- Operators are **number-preserving** (map η-electron states to η-electron states).
- Gate complexity now dependent on the fermionic semi-norms of nested commutators.

Fermion Systems

For systems in an η-fermion subspace, we tighten analysis with the **fermionic semi-norm**.

- Operators are **number-preserving** (map η-electron states to η-electron states).
- Gate complexity now dependent on the fermionic semi-norms of nested commutators.
- Same bounds but with $\lambda_{\text{comm}}^{(\eta)}$ defined from $\alpha_{\text{comm}}^{(\eta)} < \alpha_{\text{comm}}$ [8]–[10] which are generally tighter than standard commutator bounds.

Next Steps:

- **1D extrapolation**: Note that the sample complexity

$$C_{\text{sample}} = \mathcal{O}\left(\frac{1}{\varepsilon^2}\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^4\right)$$

Can we improve the $\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^4$ to $\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^2$ by extrapolating directly over $\mathrm{Tr}[e^{iHT}\rho e^{-iHT'}O]$ instead of $\mathrm{Tr}[Ze^{iHT}]$?

Next Steps:

- **1D extrapolation**: Note that the sample complexity

$$C_{\text{sample}} = \mathcal{O}\left(\frac{1}{\varepsilon^2}\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^4\right)$$

Can we improve the $\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^4$ to $\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^2$ by extrapolating directly over $\mathrm{Tr}[e^{iHT}\rho e^{-iHT'}O]$ instead of $\mathrm{Tr}[Ze^{iHT}]$?

- **Resource estimation**: Can we do resource estimates for a for more chemical systems (Systems in the Low-Energy Subspace)?

Next Steps:

- **1D extrapolation**: Note that the sample complexity

$$C_{\text{sample}} = \mathcal{O}\left(\frac{1}{\varepsilon^2}\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^4\right)$$

  Can we improve the $\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^4$ to $\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)^2$ by extrapolating directly over $\mathrm{Tr}[e^{iHT}\rho e^{-iHT'}O]$ instead of $\mathrm{Tr}[Ze^{iHT}]$?

- **Resource estimation**: Can we do resource estimates for a for more chemical systems (Systems in the Low-Energy Subspace)?

- **Analyzing constant factors** in the asymptotics for more specific performance estimates.

Acknowledgments

Samson Wang **John Preskill**

Thanks also to the rest of the Preskill Group, IQIM, and SFP for giving me such an enriching summer research experience.

Feedback:

PLEASE LEAVE YOUR AUDIENCE FEEDBACK HERE

References I

[1] L. Lin and Y. Tong, "Heisenberg-limited ground-state energy estimation for early fault-tolerant quantum computers," *PRX Quantum*, vol. 3, no. 1, Feb. 2022, ISSN: 2691-3399. DOI: 10.1103/prxquantum.3.010318. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.3.010318.

[2] K. Wan, M. Berta, and E. T. Campbell, "Randomized quantum algorithm for statistical phase estimation,", vol. 129, p. 030 503, 3 Jul. 2022, 2110.12071. DOI: 10.1103/PhysRevLett.129.030503. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.129.030503.

[3] S. Wang, S. McArdle, and M. Berta, "Qubit-efficient randomized quantum algorithms for linear algebra," *PRX Quantum*, vol. 5, no. 2, Apr. 2024, ISSN: 2691-3399. DOI: 10.1103/prxquantum.5.020324. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.5.020324.

References II

[4] G. H. Low and I. L. Chuang, "Hamiltonian simulation by qubitization," *Quantum*, vol. 3, p. 163, 2019, arXiv:1610.06546. DOI: 10.22331/q-2019-07-12-163.

[5] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, "Theory of trotter error with commutator scaling," *Physical Review X*, vol. 11, no. 1, p. 011 020, Feb. 2021. DOI: 10.1103/PhysRevX.11.011020. [Online]. Available: http://dx.doi.org/10.1103/PhysRevX.11.011020.

[6] J. D. Watson and J. Watkins, *Exponentially reduced circuit depths using trotter error mitigation*, 2024. DOI: 10.48550/ARXIV.2408.14385. arXiv: 2408.14385. [Online]. Available: https://arxiv.org/abs/2408.14385.

[7] J. Günther, F. Witteveen, A. Schmidhuber, M. Miller, M. Christandl, and A. Harrow, "Phase estimation with partially randomized time evolution," 2503.05647, 2025.

References III

[8] S. McArdle, E. Campbell, and Y. Su, "Exploiting fermion number in factorized decompositions of the electronic structure hamiltonian," *Physical Review A*, vol. 105, no. 1, Jan. 2022, ISSN: 2469-9934. DOI: 10.1103/physreva.105.012403. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.105.012403.

[9] Y. Su, H. Y. Huang, and E. T. Campbell, "Nearly tight Trotterization of interacting electrons,", vol. 5, no. 1, pp. 1–58, 2021, 2012.09194, ISSN: 2521327X. DOI: 10.22331/Q-2021-07-05-495. arXiv: 2012.09194.

[10] G. H. Low, Y. Su, Y. Tong, and M. C. Tran, "Complexity of implementing trotter steps," *PRX Quantum*, vol. 4, no. 2, May 2023, ISSN: 2691-3399. DOI: 10.1103/prxquantum.4.020323. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.4.020323.