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

Arul Rhik Mazumder
Co-mentor: Samson Wang

August 2, 2025

Presentation Overview

This presentation covers:

- **Recap of Base Algorithm and Comparisons**:
  - Reminder of motivation for simulating time evolution, trotterization and Richardson Extrapolation for error reduction.
  - Analysis of fundamental primitives and cost estimates.
  - Comparison of time evolution methods.
- **Understanding Commutators**:
  - Definition and significance in Hamiltonian simulation.
  - Commutator scaling and bounds.
- **Partial Randomization**:
  - Physical and algorithmic motivations.
  - Hamiltonian decomposition and circuit representation.
  - **Our Algorithm**: Combining Richardson Extrapolation with Partial Randomization for enhanced performance.

Base Algorithm and Comparisons

Recap: Motivation & Background

- Simulating time evolution e^{-iHt} is fundamental to:
  - Quantum Phase Estimation (QPE)
  - Harrow-Hassidim-Lloyd (HHL) algorithm
  - General quantum simulation

Recap: Motivation & Background

- Simulating time evolution e^{-iHt} is fundamental to:
  - Quantum Phase Estimation (QPE)
  - Harrow-Hassidim-Lloyd (HHL) algorithm
  - General quantum simulation
- **Trotterization:** A practical approach to approximate e^{-iHt} using native gate sequences.

Recap: Motivation & Background

- Simulating time evolution e^{-iHt} is fundamental to:
  - Quantum Phase Estimation (QPE)
  - Harrow-Hassidim-Lloyd (HHL) algorithm
  - General quantum simulation
- **Trotterization:** A practical approach to approximate e^{-iHt} using native gate sequences.
- **The Challenge:** Circuit depth typically scales unfavorably with inverse error ($\mathcal{O}(1/\varepsilon)$).

Recap: Motivation & Background

- Simulating time evolution e^{-iHt} is fundamental to:
  - Quantum Phase Estimation (QPE)
  - Harrow-Hassidim-Lloyd (HHL) algorithm
  - General quantum simulation
- **Trotterization:** A practical approach to approximate e^{-iHt} using native gate sequences.
- **The Challenge:** Circuit depth typically scales unfavorably with inverse error ($\mathcal{O}(1/\varepsilon)$).
- **Our Solution:** Employ **Richardson Extrapolation** to enhance accuracy *without increasing quantum circuit depth*.

Recap: Motivation & Background

- Simulating time evolution e^{-iHt} is fundamental to:
  - Quantum Phase Estimation (QPE)
  - Harrow-Hassidim-Lloyd (HHL) algorithm
  - General quantum simulation
- **Trotterization:** A practical approach to approximate e^{-iHt} using native gate sequences.
- **The Challenge:** Circuit depth typically scales unfavorably with inverse error ($\mathcal{O}(1/\varepsilon)$).
- **Our Solution:** Employ **Richardson Extrapolation** to enhance accuracy *without increasing quantum circuit depth*.
- **Why Trotter is appealing:** It's NISQ-friendly due to low overhead, commutator scaling, and ease of compilation.

Recap: Richardson Extrapolation for Trotter Error

- **Trotter Error Expansion:** The computed observable $O(\delta)$ can be expressed as:

$$O(\delta) = O_{\text{exact}} + c_1\delta^p + c_2\delta^{2p} + \dots$$

Recap: Richardson Extrapolation for Trotter Error

- **Trotter Error Expansion:** The computed observable $O(\delta)$ can be expressed as:

$$O(\delta) = O_{\text{exact}} + c_1\delta^p + c_2\delta^{2p} + \dots$$

- **Extrapolation Method:**
  - Run the Trotter simulation at multiple distinct step sizes δ_i.
  - Combine results linearly to cancel leading-order errors:

$$O_{\text{extrapolated}} = \sum_i \alpha_i O(\delta_i), \quad \text{with } \sum_i \alpha_i = 1$$

Recap: Richardson Extrapolation for Trotter Error

- **Trotter Error Expansion:** The computed observable $O(\delta)$ can be expressed as:
$$O(\delta) = O_{\text{exact}} + c_1\delta^p + c_2\delta^{2p} + \dots$$

- **Extrapolation Method:**
  - Run the Trotter simulation at multiple distinct step sizes δ_i.
  - Combine results linearly to cancel leading-order errors:
$$O_{\text{extrapolated}} = \sum_i \alpha_i O(\delta_i), \quad \text{with } \sum_i \alpha_i = 1$$

- **Key Benefit:** This process is purely classical post-processing, requiring *no additional quantum circuit depth*.

Recap: Richardson Extrapolation for Trotter Error

- **Trotter Error Expansion:** The computed observable $O(\delta)$ can be expressed as:
$$O(\delta) = O_{\text{exact}} + c_1\delta^p + c_2\delta^{2p} + \dots$$

- **Extrapolation Method:**
  - Run the Trotter simulation at multiple distinct step sizes δ_i.
  - Combine results linearly to cancel leading-order errors:
$$O_{\text{extrapolated}} = \sum_i \alpha_i O(\delta_i), \quad \text{with } \sum_i \alpha_i = 1$$

- **Key Benefit:** This process is purely classical post-processing, requiring *no additional quantum circuit depth*.
- **Result:** Effectively increases the order of accuracy to $\mathcal{O}(\delta^{m+1})$, where m is the number of step sizes used for extrapolation.

Recap: Estimating General Matrix Functions

- **Goal:** Efficiently compute expressions of the form $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$.

Recap: Estimating General Matrix Functions

- **Goal:** Efficiently compute expressions of the form $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$.
- **Approximation Strategy:** Decompose the function $f(A)$ into a sum of exponentials:

$$f(A) \approx \sum_k c_k e^{iAt_k}$$

Recap: Estimating General Matrix Functions

- **Goal:** Efficiently compute expressions of the form $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$.
- **Approximation Strategy:** Decompose the function $f(A)$ into a sum of exponentials:

$$f(A) \approx \sum_k c_k e^{iAt_k}$$

- **Problem Reduction:** This reduces the original problem to estimating two fundamental primitives:
  - $\mathrm{Tr}[Ze^{iAt}]$
  - $\mathrm{Tr}[e^{iAt}\rho e^{-iAt'}O]$

  where Z is an observable with bounded Schatten 1-norm.

Recap: Estimating General Matrix Functions

- **Goal:** Efficiently compute expressions of the form $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$.
- **Approximation Strategy:** Decompose the function $f(A)$ into a sum of exponentials:

$$f(A) \approx \sum_k c_k e^{iAt_k}$$

- **Problem Reduction:** This reduces the original problem to estimating two fundamental primitives:
  - $\mathrm{Tr}[Ze^{iAt}]$
  - $\mathrm{Tr}[e^{iAt}\rho e^{-iAt'}O]$

  where Z is an observable with bounded Schatten 1-norm.
- **Key Findings:**
  - We rigorously proved that Richardson Extrapolation can be effectively applied to *each individual term* in these estimations.

Recap: Estimating General Matrix Functions

- **Goal:** Efficiently compute expressions of the form $\mathrm{Tr}[f(A)\rho f(A)^\dagger O]$.
- **Approximation Strategy:** Decompose the function $f(A)$ into a sum of exponentials:

$$f(A) \approx \sum_k c_k e^{iAt_k}$$

- **Problem Reduction:** This reduces the original problem to estimating two fundamental primitives:
  - $\mathrm{Tr}[Ze^{iAt}]$
  - $\mathrm{Tr}[e^{iAt}\rho e^{-iAt'}O]$

  where Z is an observable with bounded Schatten 1-norm.
- **Key Findings:**
  - We rigorously proved that Richardson Extrapolation can be effectively applied to *each individual term* in these estimations.
  - We randomly compile over our Fourier terms and extrapolation scheduling *in one go* to improve sample complexity.

Recap: Analyzed Primitives & Cost Estimates

We rigorously analyzed the complexity of estimating our fundamental primitives to desired precision ε:

Recap: Analyzed Primitives & Cost Estimates

We rigorously analyzed the complexity of estimating our fundamental primitives to desired precision ε:

- **Primitive 1:** $\mathrm{Tr}[\rho e^{iAT}]$

Recap: Analyzed Primitives & Cost Estimates

We rigorously analyzed the complexity of estimating our fundamental primitives to desired precision ε:

- **Primitive 1:** $\mathrm{Tr}[\rho e^{iAT}]$

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

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

Recap: Analyzed Primitives & Cost Estimates

We rigorously analyzed the complexity of estimating our fundamental primitives to desired precision ε:

- **Primitive 1:** $\mathrm{Tr}[\rho e^{iAT}]$

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

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

- **Primitive 2:** $\mathrm{Tr}[e^{iAT}\rho e^{-iAT'}O]$

Recap: Analyzed Primitives & Cost Estimates

We rigorously analyzed the complexity of estimating our fundamental primitives to desired precision ε:

- **Primitive 1:** $\mathrm{Tr}[\rho e^{iAT}]$

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

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

- **Primitive 2:** $\mathrm{Tr}[e^{iAT}\rho e^{-iAT'}O]$

$$C_{\text{gate}} = \mathcal{O}\left(\Gamma(\Upsilon\lambda_{\text{comm}}T_{\text{max}})^{1+\frac{1}{p}}\log\left(\frac{1}{\varepsilon}\right)\cdot\log\left(\frac{\log\log\left(\frac{1}{\varepsilon}\right)}{\varepsilon}\right)\right)$$

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

Recap: Comparison of Time Evolution Methods

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p}\, T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}}\, T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Recap: Comparison of Time Evolution Methods

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p}\, T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Notes:

- All methods estimate $\mathrm{Tr}[\rho e^{iAt}]$ to ε error

Recap: Comparison of Time Evolution Methods

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Notes:

- All methods estimate $\text{Tr}[\rho e^{iAt}]$ to ε error
- Qubitization is not early fault-tolerant requiring $\lceil \log \Gamma \rceil + 3$ ancilla qubits.

Recap: Comparison of Time Evolution Methods

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Notes:

- All methods estimate $\text{Tr}[\rho e^{iAt}]$ to ε error
- Qubitization is not early fault-tolerant requiring $\lceil \log \Gamma \rceil + 3$ ancilla qubits.
- Exponential improvement on ε^{-1} scaling compared Trotter.

Recap: Comparison of Time Evolution Methods (Continued)

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p}) T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Our algorithm:

- Sub-quadratic time complexity.

[1] Λ is the max operator norm in the Hamiltonian decomposition.

Recap: Comparison of Time Evolution Methods (Continued)

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p}) T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p} (\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Our algorithm:

- Sub-quadratic time complexity.
- Commutator scaling similar to Trotter using λ_{comm}.

[1] Λ is the max operator norm in the Hamiltonian decomposition.

Recap: Comparison of Time Evolution Methods (Continued)

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p} T^{1+1/p} \varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Our algorithm:

- Sub-quadratic time complexity.
- Commutator scaling similar to Trotter using λ_{comm}.
- In practice $\lambda_{\text{comm}} << \Lambda$ [1]

[1] Λ is the max operator norm in the Hamiltonian decomposition.

Recap: Comparison of Time Evolution Methods (Continued)

Method	**Max Depth / Sample**	**Sample Overhead**
Qubitization [1]	$\mathcal{O}\left(\Gamma\left[\Lambda T + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right]\right)$	$\mathcal{O}(1/\varepsilon^2)$
Product Formulae [2]	$\mathcal{O}\left(\Gamma(\alpha_{\text{comm}}^{(p+1)})^{1/p}) T^{1+1/p}\varepsilon^{-1/p}\right)$	$\mathcal{O}(1/\varepsilon^2)$
Random Compiler [3]	$\mathcal{O}(\Lambda^2 T^2)$	$\mathcal{O}(1/\varepsilon^2)$
Our Algorithm	$\mathcal{O}\left(\Gamma(\lambda_{\text{comm}} T)^{1+1/p}(\log(1/\varepsilon))^2\right)$	$\mathcal{O}\left((\log\log(1/\varepsilon))^2/\varepsilon^2\right)$

Our algorithm:

- Sub-quadratic time complexity.
- Commutator scaling similar to Trotter using λ_{comm}.
- In practice $\lambda_{\text{comm}} << \Lambda$ [1]
- These comparisons are for the simple primitives, used to compute matrix functions as a subroutine in HHL and QPE.

[1] Λ is the max operator norm in the Hamiltonian decomposition.

To Do:

To Do:

Develop complete algorithms: These comparisons are for simple primitives, used to compute matrix functions as a subroutine in HHL and QPE. We would like to develop and analyze the full algorithms.

To Do:

Develop complete algorithms: These comparisons are for simple primitives, used to compute matrix functions as a subroutine in HHL and QPE. We would like to develop and analyze the full algorithms.

1D extrapolation: Note that the sample complexity

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

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 $\mathrm{Tr}[Ze^{iHT}]$?

To Do:

Develop complete algorithms: These comparisons are for simple primitives, used to compute matrix functions as a subroutine in HHL and QPE. We would like to develop and analyze the full algorithms.

1D extrapolation: Note that the sample complexity

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

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 particular chemical system?

Understanding Commutators

Understanding Commutators in Hamiltonian Simulation

What is a Commutator?:

Understanding Commutators in Hamiltonian Simulation

What is a Commutator?: For two operators A and B:

$$[A, B] := AB - BA$$

In a given decomposition $H = \sum_{i=1}^{\Gamma} H_i$, commutators quantify the extent to which the Hamiltonian terms H_i fail to commute.

Why care about commutators?

Understanding Commutators in Hamiltonian Simulation

What is a Commutator?: For two operators A and B:

$$[A, B] := AB - BA$$

In a given decomposition $H = \sum_{i=1}^{\Gamma} H_i$, commutators quantify the extent to which the Hamiltonian terms H_i fail to commute.

Why care about commutators?

- Nested commutators govern the error in product formulas.

Understanding Commutators in Hamiltonian Simulation

What is a Commutator?: For two operators A and B:

$$[A, B] := AB - BA$$

In a given decomposition $H = \sum_{i=1}^{\Gamma} H_i$, commutators quantify the extent to which the Hamiltonian terms H_i fail to commute.

Why care about commutators?

- Nested commutators govern the error in product formulas.
- **Nested Commutator Norms.** Define the j-th order commutator quantity as:

$$\alpha_{\text{comm}}^{(j)} := \sum_{\gamma_1,\ldots,\gamma_j=1}^{\Gamma} \left\| [H_{\gamma_1}, [H_{\gamma_2}, \ldots, [H_{\gamma_j}, H_{\gamma_{j+1}}] \ldots]] \right\|$$

Commutator Scaling

Our Commutator Factor λ_{comm}:

$$\lambda_{\mathsf{comm}} := \sup_{\substack{j \in \mathbb{Z}_{\geq \sigma_m} \\ 1 \leq \ell \leq K}} \left(\sum_{\substack{j_1, \ldots, j_\ell \in \mathbb{Z}_{\geq p} \\ j_1 + \cdots + j_\ell = j}} \prod_{\kappa=1}^{\ell} \frac{\alpha_{\mathsf{comm}}^{(j_\kappa+1)}}{(j_\kappa+1)^2} \right)^{1/(j+\ell)}$$

Commutator Scaling

Our Commutator Factor λ_{comm}:

$$\lambda_{\text{comm}} := \sup_{\substack{j \in \mathbb{Z}_{\geq \sigma_m} \\ 1 \leq \ell \leq K}} \left(\sum_{\substack{j_1, \ldots, j_\ell \in \mathbb{Z}_{\geq p} \\ j_1 + \cdots + j_\ell = j}} \prod_{\kappa=1}^{\ell} \frac{\alpha_{\text{comm}}^{(j_\kappa+1)}}{(j_\kappa+1)^2} \right)^{1/(j+\ell)}$$

Lemma (General Commutator Growth Constant Bound [4])

For any Hamiltonian $H = \sum_{\gamma=1}^{\Gamma} H_\gamma$, we have $\lambda_{comm} \leq 4 \sum_{\gamma=1}^{\Gamma} \|H_\gamma\|$.

Examples of Commutator Scaling and Bounds

Example 1: Electronic Structure Hamiltonians (Plane Wave Basis)

$$\alpha_{\text{comm}}^{(j)} = \mathcal{O}(n^j) \Rightarrow \lambda_{\text{comm}} = \mathcal{O}(n)$$

Example 2: k-Local Hamiltonians[2]

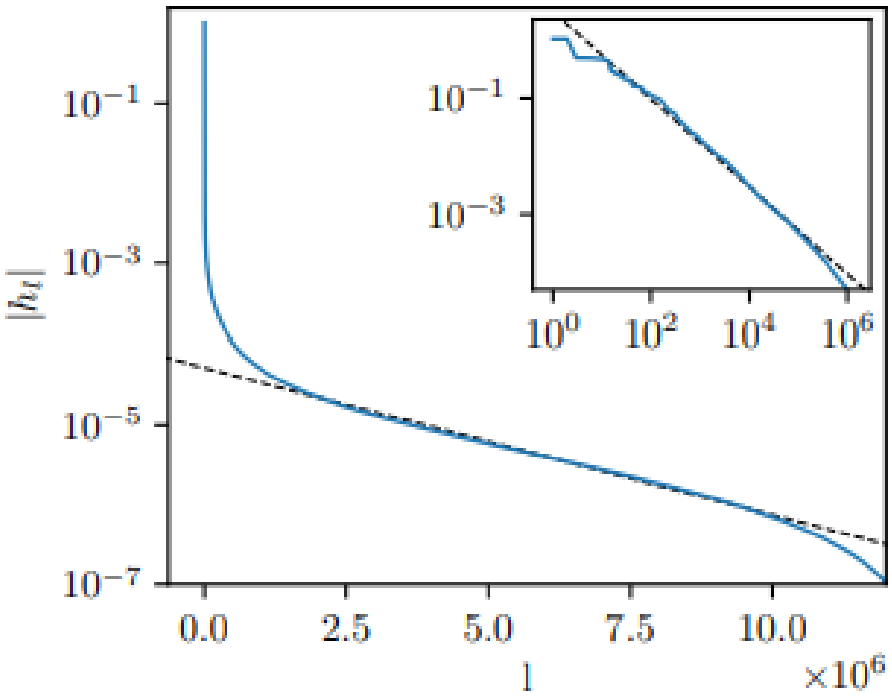
$$\alpha_{\text{comm}}^{(j)} = \mathcal{O}\left(|||H|||_1^{j-1} \|H\|_1\right) \quad \Rightarrow \quad \lambda_{\text{comm}} = \mathcal{O}\left(|||H|||_1 \|H\|_1^{\frac{1}{p+1}}\right)$$

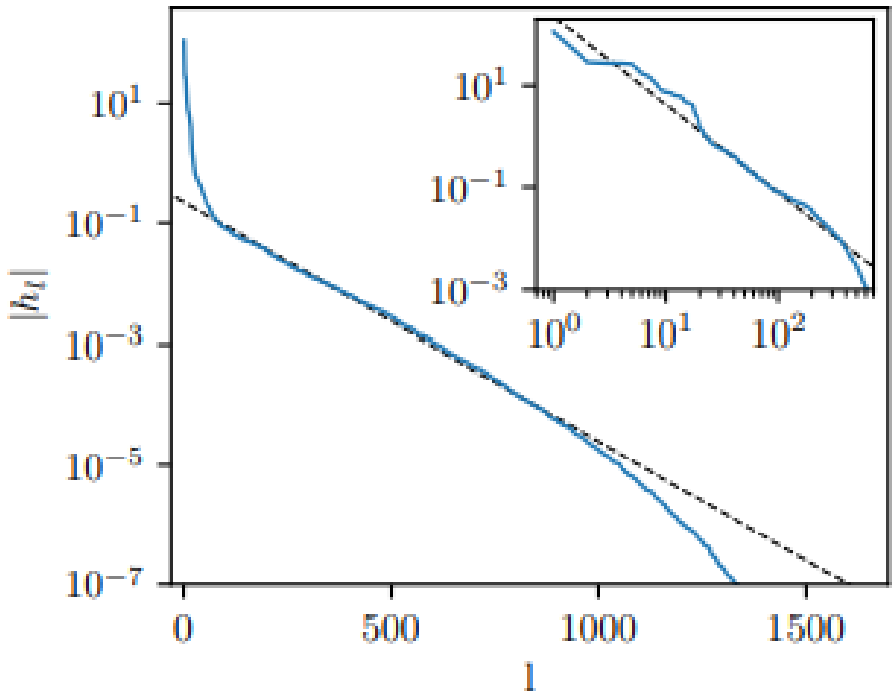
[2] Here, $|||H|||_1$ is a special induced matrix norm introduced in [2] and $\|H\|_1 = \sum_{i=1}^{\Gamma} \|H_i\|$

Partial Randomization [5]

Partial Randomization: Physical Motivation

Many useful Hamiltonians in chemistry can be decomposed into a few high weight terms and large number of low weights. [3]

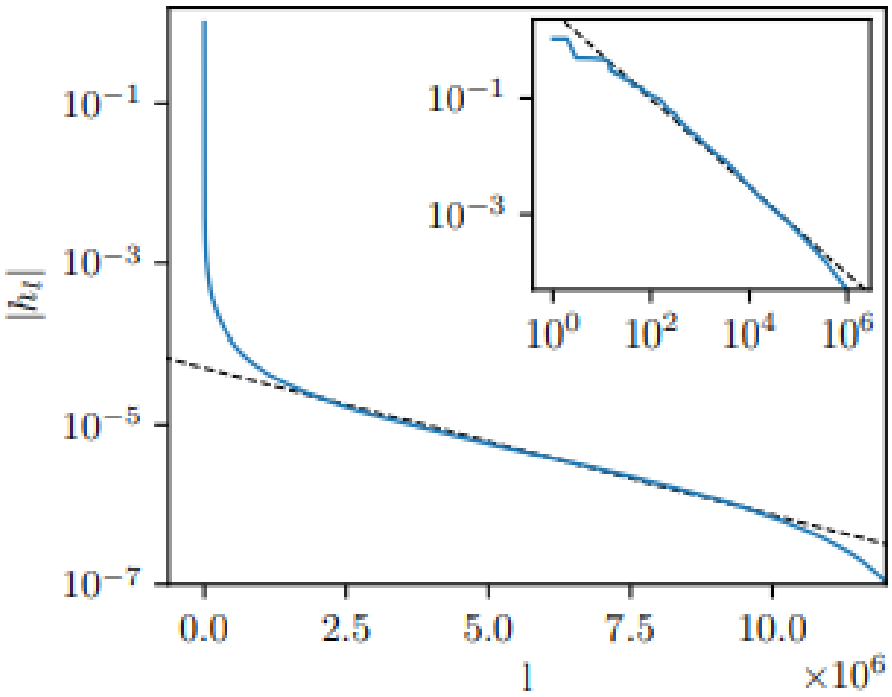


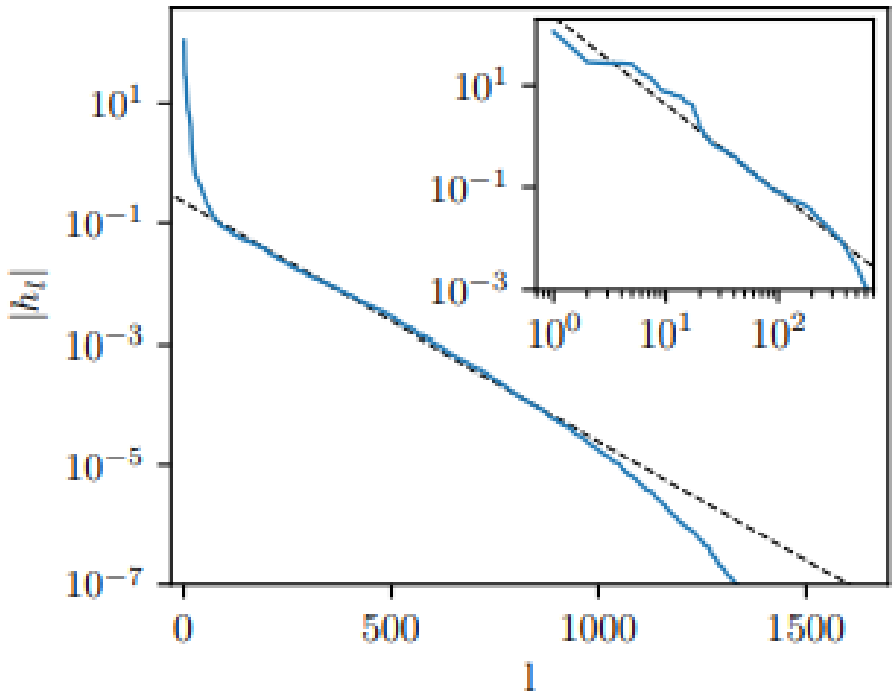


[3] Image from J. Günther, F. Witteveen, A. Schmidhuber, M. Miller, M. Christandl, and A. Harrow [5]

Partial Randomization: Physical Motivation

Many useful Hamiltonians in chemistry can be decomposed into a few high weight terms and large number of low weights. [3]



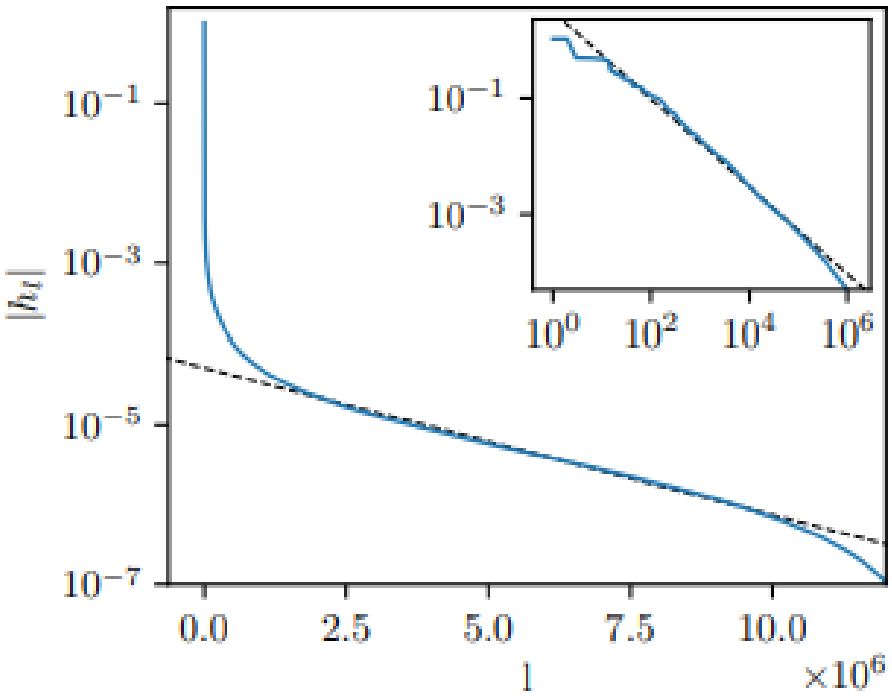


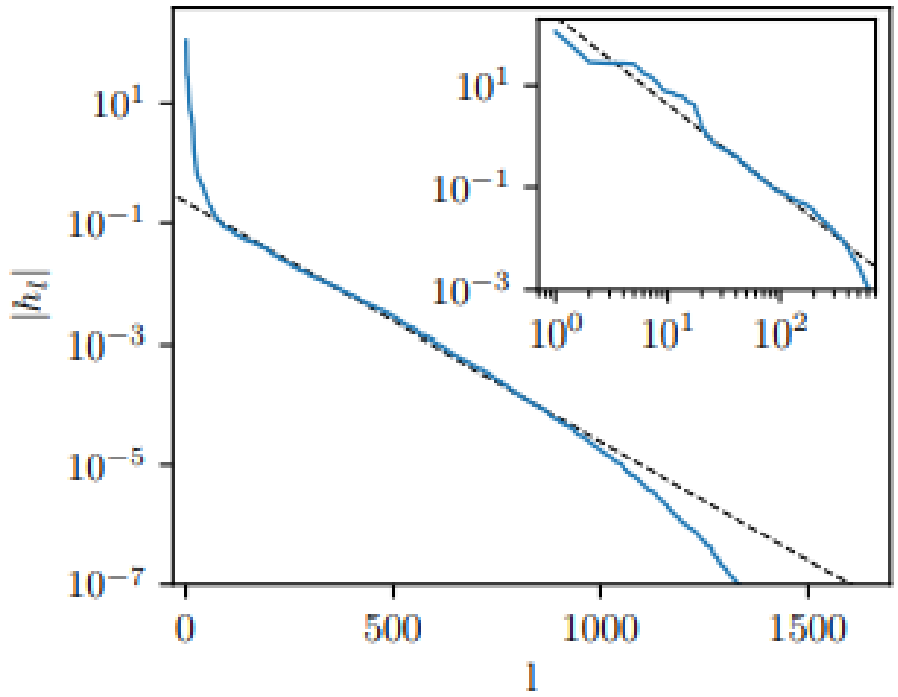
- The main plot shows exponential fit to the tail. ($|h_\ell| \approx Ae^{-b\ell}$)

[3] Image from J. Günther, F. Witteveen, A. Schmidhuber, M. Miller, M. Christandl, and A. Harrow [5]

Partial Randomization: Physical Motivation

Many useful Hamiltonians in chemistry can be decomposed into a few high weight terms and large number of low weights. [3]





- 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. ($|h_\ell| \approx C \cdot \ell^{-\alpha}$)

[3] Image from J. Günther, F. Witteveen, A. Schmidhuber, M. Miller, M. Christandl, and A. Harrow [5]

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:
Trotter Product Formulas (Deterministic):

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:
Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.
- Better scaling with ε^{-1}

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.
- Better scaling with ε^{-1}
- Problem: Circuits become impractically large as L grows. $L = \mathcal{O}(N^4)$ where N is the number of spatial orbitals.

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.
- Better scaling with ε^{-1}
- Problem: Circuits become impractically large as L grows. $L = \mathcal{O}(N^4)$ where N is the number of spatial orbitals.

Randomized Product Formulas:

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.
- Better scaling with ε^{-1}
- Problem: Circuits become impractically large as L grows. $L = \mathcal{O}(N^4)$ where N is the number of spatial orbitals.

Randomized Product Formulas:

- Removes dependence on L, cost $\sim \mathcal{O}(\lambda^2 t^2)$.

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.
- Better scaling with ε^{-1}
- Problem: Circuits become impractically large as L grows. $L = \mathcal{O}(N^4)$ where N is the number of spatial orbitals.

Randomized Product Formulas:

- Removes dependence on L, cost $\sim \mathcal{O}(\lambda^2 t^2)$.
- Problem: t scales quadratically and λ is dominated by a few terms

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.
- Better scaling with ε^{-1}
- Problem: Circuits become impractically large as L grows. $L = \mathcal{O}(N^4)$ where N is the number of spatial orbitals.

Randomized Product Formulas:

- Removes dependence on L, cost $\sim \mathcal{O}(\lambda^2 t^2)$.
- Problem: t scales quadratically and λ is dominated by a few terms

Key Insight:

- Many Hamiltonians have a long tail of small-weight terms.
- These terms inflate L but contribute little to λ.

Partial Randomization: Algorithmic Motivation

For a Hamiltonian $H = \sum_{l=1}^{L} H_l$, we can simulate the time evolution with:

Trotter Product Formulas (Deterministic):

- Cost scales linearly with L.
- Better scaling with ε^{-1}
- Problem: Circuits become impractically large as L grows. $L = \mathcal{O}(N^4)$ where N is the number of spatial orbitals.

Randomized Product Formulas:

- Removes dependence on L, cost $\sim \mathcal{O}(\lambda^2 t^2)$.
- Problem: t scales quadratically and λ is dominated by a few terms

Key Insight:

- Many Hamiltonians have a long tail of small-weight terms.
- These terms inflate L but contribute little to λ.
- **Solution:** Treat dominant terms deterministically and tail terms randomly. Combining ε^{-1} scaling with reduced L dependence.

Partial Randomization: Decomposition

Partial randomization is based on the decomposition:

$$H = \underbrace{\sum_{l=1}^{L_D} H_l}_{=H_D} + \underbrace{\sum_{m=1}^{M} h_m P_m}_{=H_R}$$

where we assume that the operators P_m are Pauli.

Partial Randomization: Decomposition

Partial randomization is based on the decomposition:

$$H = \underbrace{\sum_{l=1}^{L_D} H_l}_{=H_D} + \underbrace{\sum_{m=1}^{M} h_m P_m}_{=H_R}$$

where we assume that the operators P_m are Pauli.

- H_D contains the terms we treat deterministically with Trotter

Partial Randomization: Decomposition

Partial randomization is based on the decomposition:

$$H = \underbrace{\sum_{l=1}^{L_D} H_l}_{=H_D} + \underbrace{\sum_{m=1}^{M} h_m P_m}_{=H_R}$$

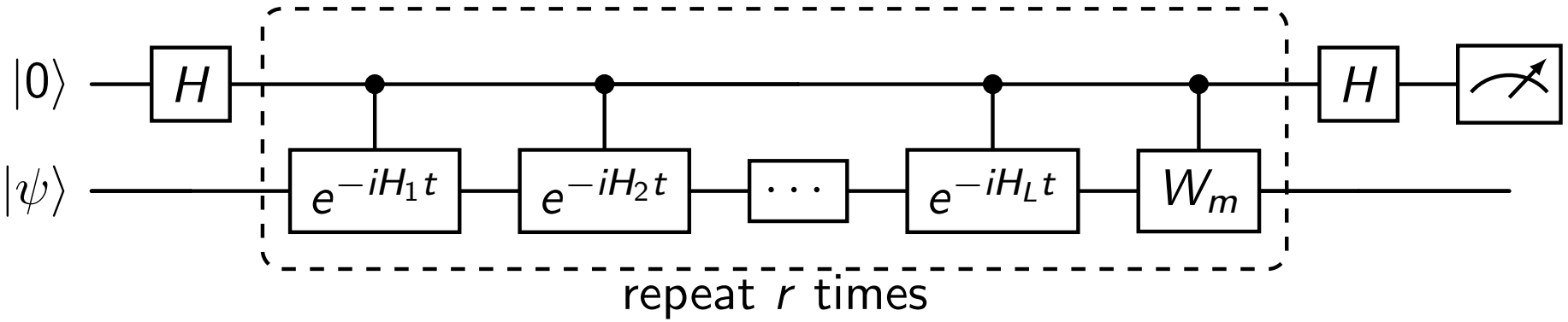
where we assume that the operators P_m are Pauli.

- H_D contains the terms we treat deterministically with Trotter
- H_R is computed with a random product formula

$$\lambda_R = \sum_{m=1}^{M} |h_m| \ll \lambda, \quad \text{and} \quad L_D \ll L,$$

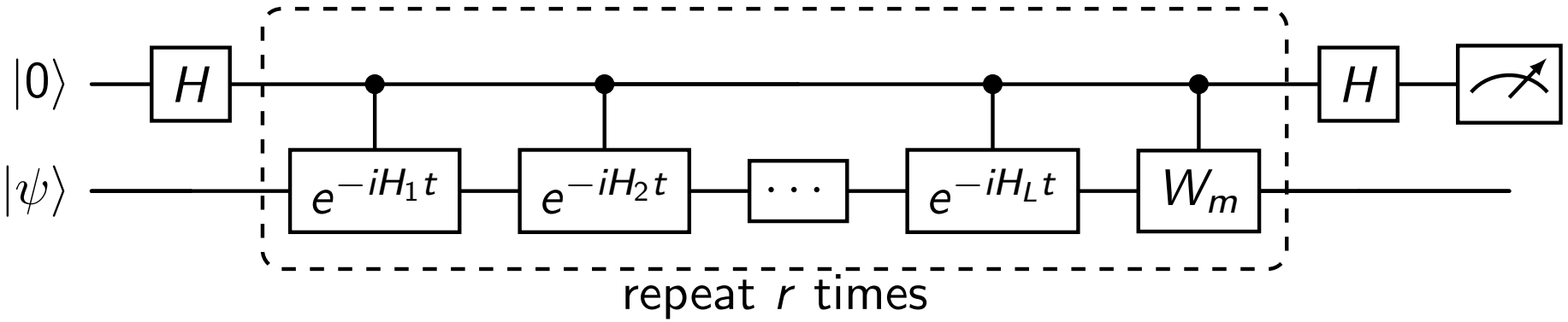
Partial Randomization: Circuit Representation

Consider the following 1st order partial random Trotter formula over r time steps. This would estimate $\mathrm{Re}[Ze^{iHT}]$



Partial Randomization: Circuit Representation

Consider the following 1st order partial random Trotter formula over r time steps. This would estimate $\mathrm{Re}[Ze^{iHT}]$



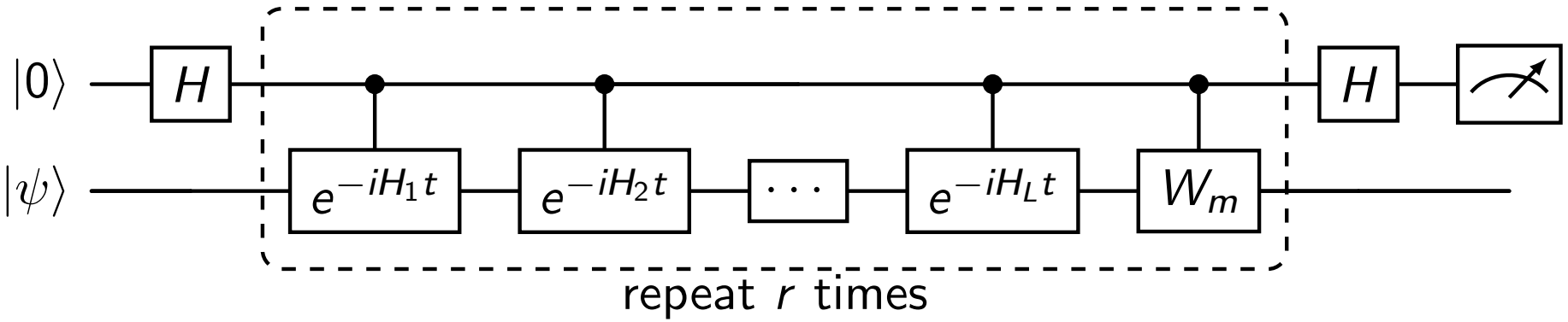
To implement each W_m we use the Randomized Taylor Expansion [3]:

$$e^{iH_R t} = \left(e^{iH_R \tau}\right)^d = \beta(\tau)^d \left(\sum_m b_m U_m\right)^d$$

$$= \beta(\tau)^d \sum_{m_1,\ldots,m_d} b_{m_1} \cdots b_{m_d} U_{m_1} \cdots U_{m_d}$$

Partial Randomization: Circuit Representation

Consider the following 1st order partial random Trotter formula over r time steps. This would estimate $\mathrm{Re}[Ze^{iHT}]$



To implement each W_m we use the Randomized Taylor Expansion [3]:

$$e^{iH_R t} = \left(e^{iH_R \tau}\right)^d = \beta(\tau)^d \left(\sum_m b_m U_m\right)^d$$

$$= \beta(\tau)^d \sum_{m_1,\ldots,m_d} b_{m_1} \cdots b_{m_d} U_{m_1} \cdots U_{m_d}$$

W_m samples $U_{m_1} \ldots U_{m_d}$ over $\{b_{m_1} \ldots b_{m_d}\}$. Note that $d = \mathcal{O}(\frac{\lambda_R^2 t^2}{r})$

Our Algorithm

Our Algorithm: Richardson + Partial Randomization

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Key Ingredients:

- Decompose Hamiltonian: $H = \sum_{l=1}^{L_D} H_l + \sum_{m=1}^{M} h_m P_m$
- Choose Richardson step sizes $\{s_k\}$ and coefficients $\{b_k\}$.

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Key Ingredients:

- Decompose Hamiltonian: $H = \sum_{l=1}^{L_D} H_l + \sum_{m=1}^{M} h_m P_m$
- Choose Richardson step sizes $\{s_k\}$ and coefficients $\{b_k\}$.

Main Loop:

- For $i = 1$ to N:

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Key Ingredients:

- Decompose Hamiltonian: $H = \sum_{l=1}^{L_D} H_l + \sum_{m=1}^{M} h_m P_m$
- Choose Richardson step sizes $\{s_k\}$ and coefficients $\{b_k\}$.

Main Loop:

- For $i = 1$ to N:
  - Sample k_i according to $|b_k|$, scale time $t_{k_i} = s_{k_i} T$, number of steps $r_{k_i} = 1/s_{k_i}$, and step size $\delta_{k_i} = T/r_{k_i}$

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Key Ingredients:

- Decompose Hamiltonian: $H = \sum_{l=1}^{L_D} H_l + \sum_{m=1}^{M} h_m P_m$
- Choose Richardson step sizes $\{s_k\}$ and coefficients $\{b_k\}$.

Main Loop:

- For $i = 1$ to N:
  - Sample k_i according to $|b_k|$, scale time $t_{k_i} = s_{k_i} T$, number of steps $r_{k_i} = 1/s_{k_i}$, and step size $\delta_{k_i} = T/r_{k_i}$
  - Generate a randomized evolution for H_R using d_{k_i} unitary block.

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Key Ingredients:

- Decompose Hamiltonian: $H = \sum_{l=1}^{L_D} H_l + \sum_{m=1}^{M} h_m P_m$
- Choose Richardson step sizes $\{s_k\}$ and coefficients $\{b_k\}$.

Main Loop:

- For $i = 1$ to N:
  - Sample k_i according to $|b_k|$, scale time $t_{k_i} = s_{k_i} T$, number of steps $r_{k_i} = 1/s_{k_i}$, and step size $\delta_{k_i} = T/r_{k_i}$
  - Generate a randomized evolution for H_R using d_{k_i} unitary block.
  - Combine with deterministic Trotter step $\mathcal{P}(\delta_{k_i})$ from H_D.

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Key Ingredients:

- Decompose Hamiltonian: $H = \sum_{l=1}^{L_D} H_l + \sum_{m=1}^{M} h_m P_m$
- Choose Richardson step sizes $\{s_k\}$ and coefficients $\{b_k\}$.

Main Loop:

- For $i = 1$ to N:
  - Sample k_i according to $|b_k|$, scale time $t_{k_i} = s_{k_i} T$, number of steps $r_{k_i} = 1/s_{k_i}$, and step size $\delta_{k_i} = T/r_{k_i}$
  - Generate a randomized evolution for H_R using d_{k_i} unitary block.
  - Combine with deterministic Trotter step $\mathcal{P}(\delta_{k_i})$ from H_D.
  - Estimate real and imaginary parts via Hadamard tests.

Our Algorithm: Richardson + Partial Randomization

Goal: Estimate $\mathrm{Tr}[Ze^{iHT}]$ using a Trotter + randomization scheme with Richardson extrapolation.

Key Ingredients:

- Decompose Hamiltonian: $H = \sum_{l=1}^{L_D} H_l + \sum_{m=1}^{M} h_m P_m$
- Choose Richardson step sizes $\{s_k\}$ and coefficients $\{b_k\}$.

Main Loop:

- For $i = 1$ to N:
  - Sample k_i according to $|b_k|$, scale time $t_{k_i} = s_{k_i} T$, number of steps $r_{k_i} = 1/s_{k_i}$, and step size $\delta_{k_i} = T/r_{k_i}$
  - Generate a randomized evolution for H_R using d_{k_i} unitary block.
  - Combine with deterministic Trotter step $\mathcal{P}(\delta_{k_i})$ from H_D.
  - Estimate real and imaginary parts via Hadamard tests.

Output: Final estimate is the average over N samples:

$$\hat{Y}_N = \frac{1}{N} \sum_{i=1}^{N} Y^{(i)}$$

Richardson Improvement to Partial Randomization

To estimate $\mathrm{Tr}[\rho e^{iHT}]$, our algorithm shows the exponential improvement in gate complexity with respect to error:

Richardson Improvement to Partial Randomization

To estimate $\mathrm{Tr}[\rho e^{iHT}]$, our algorithm shows the exponential improvement in gate complexity with respect to error:

- Standard Partial Randomization

$$C_{\text{gate}} = \mathcal{O}\left(L_D(\tilde{\alpha}_{\text{comm}}^{(p+1)})^{\frac{1}{p}} T^{1+\frac{1}{p}} \varepsilon^{-1/p} + \lambda_R^2 T^2\right)$$

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

Richardson Improvement to Partial Randomization

To estimate $\mathrm{Tr}[\rho e^{iHT}]$, our algorithm shows the exponential improvement in gate complexity with respect to error:

- Standard Partial Randomization

$$C_{\text{gate}} = \mathcal{O}\left(L_D(\tilde{\alpha}_{\text{comm}}^{(p+1)})^{\frac{1}{p}} T^{1+\frac{1}{p}} \varepsilon^{-1/p} + \lambda_R^2 T^2\right)$$
$$C_{\text{sample}} = \mathcal{O}\left(\varepsilon^{-2}\right)$$

- Richardson-extrapolated Partial Randomization

$$C_{\text{gate}} = \mathcal{O}\left(L_D(\Upsilon \tilde{\lambda}_{\text{comm}} T)^{1+\frac{1}{p}} \log^2(1/\varepsilon) + \lambda_R^2 T^2\right)$$
$$C_{\text{sample}} = \mathcal{O}\left((\log\log(1/\varepsilon))^2 \varepsilon^{-2})\right)$$

Richardson Improvement to Partial Randomization

To estimate $\mathrm{Tr}[\rho e^{iHT}]$, our algorithm shows the exponential improvement in gate complexity with respect to error:

- Standard Partial Randomization

$$C_{\text{gate}} = \mathcal{O}\left(L_D(\tilde{\alpha}_{\text{comm}}^{(p+1)})^{\frac{1}{p}} T^{1+\frac{1}{p}} \varepsilon^{-1/p} + \lambda_R^2 T^2\right)$$
$$C_{\text{sample}} = \mathcal{O}\left(\varepsilon^{-2}\right)$$

- Richardson-extrapolated Partial Randomization

$$C_{\text{gate}} = \mathcal{O}\left(L_D(\Upsilon \tilde{\lambda}_{\text{comm}} T)^{1+\frac{1}{p}} \log^2(1/\varepsilon) + \lambda_R^2 T^2\right)$$
$$C_{\text{sample}} = \mathcal{O}\left((\log\log(1/\varepsilon))^2 \varepsilon^{-2})\right)$$

Note that $\tilde{\alpha}$ is commutator for partial randomized decomposition. We rigorously study this quantity to provide tighter bounds.

Bounding Commutator Factors

Lemma

Let $H = H_D + H_R$, where $H_D = \sum_{i \in D} H_i$ is the deterministic part and H_R is the randomized part. Let $\tilde{\alpha}_{comm}$ be the nested commutator norm computed over a grouped decomposition of $\{H_l\}_{l=1}^{L_D} \cup H_R$. Let $\tilde{\lambda}_{comm}$ be the resulting relevant commutator factor from applying its derivation using $\tilde{\alpha}_{comm}$. Then

$$\tilde{\lambda}_{comm} \leq \lambda_{comm}$$

Next Steps: Overview

1. Main algorithm:
   - **Develop complete algorithms**
   - **1D extrapolation?**
   - **Resource estimation?**

Next Steps: Overview

1. Main algorithm:
   - **Develop complete algorithms**
   - **1D extrapolation?**
   - **Resource estimation?**
2. Problem-specific questions
   - **Low Energy Subspace**: Can we tighten our error analysis for specific systems that only occupy the low-energy subspace of the Hilbert space?
   - **Fermionic Systems**: Can we tighten our error analysis for specific systems that preserve fermion number?
   - **Understanding** λ_{comm}: are there other examples where we can get strong bounds?

References I

[1] 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.

[2] 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.

[3] 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.

References II

[4] 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.

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