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

Arul Rhik Mazumder
Co-mentor: Samson Wang

August 2, 2025

Motivation & Context

For a quantum system with Hamiltonian H, the time evolution of a state $|\psi(t)\rangle$ is governed by:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle, \quad \text{where} \quad U(t) = e^{-iHt/\hbar}$$

Motivation & Context

For a quantum system with Hamiltonian H, the time evolution of a state $|\psi(t)\rangle$ is governed by:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle, \quad \text{where} \quad U(t) = e^{-iHt/\hbar}$$

- Time evolution e^{-iHt} is a core building block of quantum algorithms (QPE, HHL, quantum simulation).

Motivation & Context

For a quantum system with Hamiltonian H, the time evolution of a state $|\psi(t)\rangle$ is governed by:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle, \quad \text{where} \quad U(t) = e^{-iHt/\hbar}$$

- Time evolution e^{-iHt} is a core building block of quantum algorithms (QPE, HHL, quantum simulation).
- **Trotterization** approximates e^{-iHt} using native gate sequences.

Motivation & Context

For a quantum system with Hamiltonian H, the time evolution of a state $|\psi(t)\rangle$ is governed by:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle, \quad \text{where} \quad U(t) = e^{-iHt/\hbar}$$

- Time evolution e^{-iHt} is a core building block of quantum algorithms (QPE, HHL, quantum simulation).
- **Trotterization** approximates e^{-iHt} using native gate sequences.
- **Challenge:** Error scales poorly with precision ε:

$$\text{Number of steps} \sim \frac{1}{\varepsilon} \quad \text{(first order)}, \quad \sim \varepsilon^{-\frac{1}{p}} \text{ for order } p$$

Motivation & Context

For a quantum system with Hamiltonian H, the time evolution of a state $|\psi(t)\rangle$ is governed by:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle, \quad \text{where} \quad U(t) = e^{-iHt/\hbar}$$

- Time evolution e^{-iHt} is a core building block of quantum algorithms (QPE, HHL, quantum simulation).
- **Trotterization** approximates e^{-iHt} using native gate sequences.
- **Challenge:** Error scales poorly with precision ε:

$$\text{Number of steps} \sim \frac{1}{\varepsilon} \quad \text{(first order)}, \quad \sim \varepsilon^{-\frac{1}{p}} \text{ for order } p$$

- **Goal:** Improve precision scaling via classical extrapolation, without increasing quantum circuit depth.

Product Formulae (Background)

Given non-commuting operators A, B, we want to approximate:

$$e^{(A+B)t}$$

Product Formulae (Background)

Given non-commuting operators A, B, we want to approximate:

$$e^{(A+B)t}$$

1st-order Trotter (Lie Product Formula):

$$e^{(A+B)t} \approx \left(e^{At/n} e^{Bt/n}\right)^n + \mathcal{O}(t^2/n)$$

Product Formulae (Background)

Given non-commuting operators A, B, we want to approximate:

$$e^{(A+B)t}$$

1st-order Trotter (Lie Product Formula):

$$e^{(A+B)t} \approx \left(e^{At/n} e^{Bt/n}\right)^n + \mathcal{O}(t^2/n)$$

2nd-order (Trotter–Suzuki):

$$e^{(A+B)t} \approx \left(e^{At/2n} e^{Bt/n} e^{At/2n}\right)^n + \mathcal{O}(t^3/n^2)$$

Product Formulae (Background)

Given non-commuting operators A, B, we want to approximate:

$$e^{(A+B)t}$$

1st-order Trotter (Lie Product Formula):

$$e^{(A+B)t} \approx \left(e^{At/n} e^{Bt/n}\right)^n + \mathcal{O}(t^2/n)$$

2nd-order (Trotter–Suzuki):

$$e^{(A+B)t} \approx \left(e^{At/2n} e^{Bt/n} e^{At/2n}\right)^n + \mathcal{O}(t^3/n^2)$$

$2k$**-th-order (Recursive Suzuki Form):**

$$S_{2k}(t) = S_{2k-2}(p_k t)^2 \, S_{2k-2}((1-4p_k)t) \, S_{2k-2}(p_k t)^2$$

where $p_k = 1/(4 - 4^{1/(2k-1)})$

Product Formulae (Implementation)

Given $H = \sum_{j=1}^{m} H_j$, Trotterize as:

$$U(t) \approx \left(\prod_{j=1}^{m} e^{-iH_j t/n} \right)^n$$

Product Formulae (Implementation)

Given $H = \sum_{j=1}^{m} H_j$, Trotterize as:

$$U(t) \approx \left(\prod_{j=1}^{m} e^{-iH_j t/n} \right)^n$$

- Each H_j is a simple term (e.g., Pauli string).

Product Formulae (Implementation)

Given $H = \sum_{j=1}^{m} H_j$, Trotterize as:

$$U(t) \approx \left(\prod_{j=1}^{m} e^{-iH_j t/n} \right)^n$$

- Each H_j is a simple term (e.g., Pauli string).
- Implement $e^{-iH_j t/n}$ using native gates like R_z, CNOT.

Product Formulae (Implementation)

Given $H = \sum_{j=1}^{m} H_j$, Trotterize as:

$$U(t) \approx \left(\prod_{j=1}^{m} e^{-iH_j t/n} \right)^n$$

- Each H_j is a simple term (e.g., Pauli string).
- Implement $e^{-iH_j t/n}$ using native gates like R_z, CNOT.
- Higher-order Trotter reduces error:

$$r = \# \text{ of steps} = \mathcal{O}\left(\lambda_{\text{comm}} \frac{t^{1+1/p}}{\varepsilon^{1/p}} \right)$$

where λ_{comm} measures non-commutativity.

Product Formulae (Implementation)

Given $H = \sum_{j=1}^{m} H_j$, Trotterize as:

$$U(t) \approx \left(\prod_{j=1}^{m} e^{-iH_j t/n} \right)^n$$

- Each H_j is a simple term (e.g., Pauli string).
- Implement $e^{-iH_j t/n}$ using native gates like R_z, CNOT.
- Higher-order Trotter reduces error:

$$r = \#\text{ of steps} = \mathcal{O}\left(\lambda_{\text{comm}} \frac{t^{1+1/p}}{\varepsilon^{1/p}} \right)$$

  where λ_{comm} measures non-commutativity.
- Gate complexity: $\mathcal{O}(mr)$.

Modern Approaches (Qubitization)

Qubitization is an alternative to time evolution, achieving optimal asymptotic scaling with respect to error.

Modern Approaches (Qubitization)

Qubitization is an alternative to time evolution, achieving optimal asymptotic scaling with respect to error.

Basic Idea:

- Encode Hamiltonian $H = \sum_{j=1}^{m} a_j H_j$ using a block encoding.

$$U = \begin{bmatrix} H/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \text{where } \alpha = \sum_j |a_j|$$

- Use quantum signal processing (QSP) to implement e^{-iHt} with optimal gate complexity: $\mathcal{O}\left(m\alpha(t + \log(1/\epsilon))\right)$

Modern Approaches (Qubitization)

Qubitization is an alternative to time evolution, achieving optimal asymptotic scaling with respect to error.

Basic Idea:

- Encode Hamiltonian $H = \sum_{j=1}^{m} a_j H_j$ using a block encoding.

$$U = \begin{bmatrix} H/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \text{where } \alpha = \sum_j |a_j|$$

- Use quantum signal processing (QSP) to implement e^{-iHt} with optimal gate complexity: $\mathcal{O}\left(m\alpha(t + \log(1/\epsilon))\right)$

Advantages:

- Asymptotically optimal error scaling with fewer steps needed than Trotter at high precision.

Modern Approaches (Qubitization)

Qubitization is an alternative to time evolution, achieving optimal asymptotic scaling with respect to error.

Basic Idea:

- Encode Hamiltonian $H = \sum_{j=1}^{m} a_j H_j$ using a block encoding.

$$U = \begin{bmatrix} H/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \text{where } \alpha = \sum_j |a_j|$$

- Use quantum signal processing (QSP) to implement e^{-iHt} with optimal gate complexity: $\mathcal{O}\left(m\alpha(t + \log(1/\epsilon))\right)$

Advantages:

- Asymptotically optimal error scaling with fewer steps needed than Trotter at high precision.

Challenges:

- Requires ancillary qubits.
- Needs oracles for state preparation and more complex to compile and implement on near-term devices.

Why Trotterization?

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

Why Trotterization?

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

Why Trotterization?

- **Low qubit overhead:** Requires no ancillas or block-encoding circuits.
- **Simple to compile:** Operators decompose naturally into native gate sets.
- **Structure-preserving:** Tends to maintain conserved quantities, symmetries, and locality.

Why Trotterization?

- **Low qubit overhead:** Requires no ancillas or block-encoding circuits.
- **Simple to compile:** Operators decompose naturally into native gate sets.
- **Structure-preserving:** Tends to maintain conserved quantities, symmetries, and locality.
- **Commutator scaling:** Errors scale with nested commutators, which are often small in realistic systems. Performs substantially better when $\lambda_{\text{comm}} << \|H\|_1$.

Why Trotterization?

- **Low qubit overhead:** Requires no ancillas or block-encoding circuits.
- **Simple to compile:** Operators decompose naturally into native gate sets.
- **Structure-preserving:** Tends to maintain conserved quantities, symmetries, and locality.
- **Commutator scaling:** Errors scale with nested commutators, which are often small in realistic systems. Performs substantially better when $\lambda_{\text{comm}} << \|H\|_1$.
- **Theoretically intriguing:** Observed error often far below worst-case bounds, suggesting gaps in our theoretical understanding.

Richardson Extrapolation: Concept

Goal: Improve accuracy of quantum simulations without deeper circuits when used as an algorithmic primitive

Richardson Extrapolation: Concept

Goal: Improve accuracy of quantum simulations without deeper circuits when used as an algorithmic primitive

Trotterized observable:

$$O(\delta) = O_{\text{exact}} + c_1\delta^p + c_2\delta^{p+1} + \cdots$$

Richardson Extrapolation: Concept

Goal: Improve accuracy of quantum simulations without deeper circuits when used as an algorithmic primitive

Trotterized observable:

$$O(\delta) = O_{\text{exact}} + c_1\delta^p + c_2\delta^{p+1} + \cdots$$

Idea: Simulate at step sizes $\delta_1, \delta_2, \ldots, \delta_k$, then cancel leading errors via a linear combination:

$$O_{\text{extrap}} = \sum_{i=1}^{k} \alpha_i O(\delta_i), \quad \sum \alpha_i = 1$$

Richardson Extrapolation: Concept

Goal: Improve accuracy of quantum simulations without deeper circuits when used as an algorithmic primitive

Trotterized observable:

$$O(\delta) = O_{\text{exact}} + c_1\delta^p + c_2\delta^{p+1} + \cdots$$

Idea: Simulate at step sizes $\delta_1, \delta_2, \ldots, \delta_k$, then cancel leading errors via a linear combination:

$$O_{\text{extrap}} = \sum_{i=1}^{k} \alpha_i O(\delta_i), \quad \sum \alpha_i = 1$$

- Choose α_i to cancel terms $\delta^p, \delta^{p+1}, \ldots$
- Only classical postprocessing — no circuit depth increase!

Richardson Extrapolation: Example (1st Order)

Suppose the Trotter error scales as:

$$f(\delta) = f(0) + c\delta + \mathcal{O}(\delta^2)$$

Richardson Extrapolation: Example (1st Order)

Suppose the Trotter error scales as:

$$f(\delta) = f(0) + c\delta + \mathcal{O}(\delta^2)$$

Simulate at two step sizes: δ and $\delta/2$

$$f(\delta/2) = f(0) + c\frac{\delta}{2} + \mathcal{O}(\delta^2)$$

Richardson Extrapolation: Example (1st Order)

Suppose the Trotter error scales as:

$$f(\delta) = f(0) + c\delta + \mathcal{O}(\delta^2)$$

Simulate at two step sizes: δ and $\delta/2$

$$f(\delta/2) = f(0) + c\frac{\delta}{2} + \mathcal{O}(\delta^2)$$

Construct extrapolated estimate:

$$F^{(1)}(\delta) = \frac{f(\delta/2) - \frac{1}{2}f(\delta)}{1 - \frac{1}{2}} = 2f(\delta/2) - f(\delta)$$

Richardson Extrapolation: Example (1st Order)

Suppose the Trotter error scales as:

$$f(\delta) = f(0) + c\delta + \mathcal{O}(\delta^2)$$

Simulate at two step sizes: δ and $\delta/2$

$$f(\delta/2) = f(0) + c\frac{\delta}{2} + \mathcal{O}(\delta^2)$$

Construct extrapolated estimate:

$$F^{(1)}(\delta) = \frac{f(\delta/2) - \frac{1}{2}f(\delta)}{1 - \frac{1}{2}} = 2f(\delta/2) - f(\delta)$$

This cancels the $\mathcal{O}(\delta)$ term, improving error to:

$$F^{(1)}(\delta) = f(0) + \mathcal{O}(\delta^2)$$

Richardson Extrapolation: Benefits

- **Improves accuracy:** Cancels Trotter error up to order $\mathcal{O}(\delta^{m+1})$ using m samples.

Richardson Extrapolation: Benefits

- **Improves accuracy:** Cancels Trotter error up to order $\mathcal{O}(\delta^{m+1})$ using m samples.
- **Efficient postprocessing:** Achieved purely classically; no increase in circuit depth.

Richardson Extrapolation: Benefits

- **Improves accuracy:** Cancels Trotter error up to order $\mathcal{O}(\delta^{m+1})$ using m samples.
- **Efficient postprocessing:** Achieved purely classically; no increase in circuit depth.
- **Improves precision scaling:**

$$|F^{(m)}(\delta) - \langle O(T) \rangle| = \mathcal{O}(s^{2m} T^{2m(1+1/p)})$$

for symmetric order-p Trotter formulas.

Richardson Extrapolation: Benefits

- **Improves accuracy:** Cancels Trotter error up to order $\mathcal{O}(\delta^{m+1})$ using m samples.
- **Efficient postprocessing:** Achieved purely classically; no increase in circuit depth.
- **Improves precision scaling:**

$$|F^{(m)}(\delta) - \langle O(T)\rangle| = \mathcal{O}(s^{2m} T^{2m(1+1/p)})$$

  for symmetric order-p Trotter formulas.
- **Hardware-friendly:** Well-suited to NISQ-era devices — short circuits + more measurements.

Richardson Extrapolation: Benefits

- **Improves accuracy:** Cancels Trotter error up to order $\mathcal{O}(\delta^{m+1})$ using m samples.
- **Efficient postprocessing:** Achieved purely classically; no increase in circuit depth.
- **Improves precision scaling:**

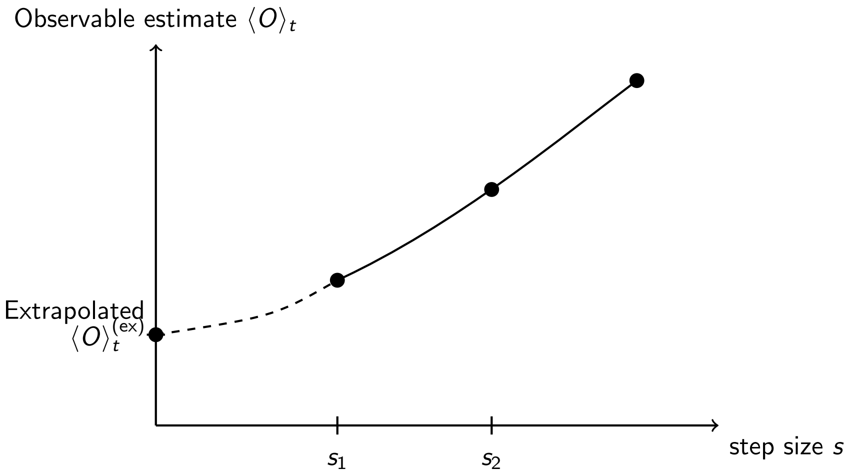
$$|F^{(m)}(\delta) - \langle O(T)\rangle| = \mathcal{O}(s^{2m} T^{2m(1+1/p)})$$

  for symmetric order-p Trotter formulas.
- **Hardware-friendly:** Well-suited to NISQ-era devices — short circuits + more measurements.

Trade-off:

- Requires multiple simulations at different δ_i.
- Sensitive to sampling noise — averaging must be precise.

Richardson Extrapolation of Observable Estimates



Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

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

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

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

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

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

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

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

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

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

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

Estimating General Matrix Functions

Goal: Estimate physical observables of the form:

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

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

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

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

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

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

$\mathrm{Tr}[Ze^{iAT}]$ has a bounded power series

We first need to show that we can apply Richardson extrapolation on each of the $\mathrm{Tr}[Ze^{iAt_k}]$ terms, which requires that our desired quantities can be written as a power series.

$\mathrm{Tr}[Ze^{iAT}]$ has a bounded power series

We first need to show that we can apply Richardson extrapolation on each of the $\mathrm{Tr}[Ze^{iAt_k}]$ terms, which requires that our desired quantities can be written as a power series.

Lemma (Observable Error Expansion)

For a staged p-th order product formula $\mathcal{P}$ of symmetry class σ, the observable satisfies:

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

- *s is the step size, so $r := 1/s$ is the number of steps*
- *$\tilde{E}, \tilde{F}$: operator-valued error terms*
- *Enables structured Richardson extrapolation of expectation values*

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

Algorithm:

1. Approximate $f(A)$ by truncated Fourier expansion: $f(A) \approx \sum_{k=1}^{K} c_k e^{iAt_k}$

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

Algorithm:

1. Approximate $f(A)$ by truncated Fourier expansion: $f(A) \approx \sum_{k=1}^{K} c_k e^{iAt_k}$
2. For each t_k, estimate $\mathrm{Tr}[Ze^{iAt_k}]$ via Richardson-extrapolated circuits.

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

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

Algorithm:

1. Approximate $f(A)$ by truncated Fourier expansion: $f(A) \approx \sum_{k=1}^{K} c_k e^{iAt_k}$
2. For each t_k, estimate $\mathrm{Tr}[Ze^{iAt_k}]$ via Richardson-extrapolated circuits.

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

3. Combine estimates using weighted sum:

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

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

Algorithm:

1. Approximate $f(A)$ by truncated Fourier expansion: $f(A) \approx \sum_{k=1}^{K} c_k e^{iAt_k}$
2. For each t_k, estimate $\mathrm{Tr}[Ze^{iAt_k}]$ via Richardson-extrapolated circuits.

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

3. Combine estimates using weighted sum:

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

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 $\mathcal{O}\left(\frac{\|\vec{c}\|_2^2 \|\vec{b}\|_2^2 \|Z\|_1^2 K^2 m^2}{\varepsilon^2} \cdot \log\left(\frac{1}{\delta}\right)\right)$

Algorithm:

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

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

Randomized 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 $\mathcal{Z} = \sum_{k,j} |c_k b_j|$

Randomized 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

Randomized 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

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

Why is Randomized Better

Goal: Estimate weighted sum of traces:

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

Why is Randomized Better

Goal: Estimate weighted sum of traces:

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

Deterministic Method:

- Computes all $K \cdot m$ terms equally
- Wastes effort on small or negligible terms
- Sample cost scales with $\|c\|_2^2 \|b\|_2^2$

Why is Randomized Better

Goal: Estimate weighted sum of traces:

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

Deterministic Method:

- Computes all $K \cdot m$ terms equally
- Wastes effort on small or negligible terms
- Sample cost scales with $\|c\|_2^2 \|b\|_2^2$

Randomized (Using Importance Sampling)

- Sample (k, j) with probability $\propto |c_k b_j|$
- Focuses effort on largest contributors
- Cost scales with $c^2 = \left(\sum_{k,j} |c_k b_j|\right)^2$

Why is Randomized Better

Goal: Estimate weighted sum of traces:

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

Deterministic Method:

- Computes all $K \cdot m$ terms equally
- Wastes effort on small or negligible terms
- Sample cost scales with $\|c\|_2^2 \|b\|_2^2$

Randomized (Using Importance Sampling)

- Sample (k, j) with probability $\propto |c_k b_j|$
- Focuses effort on largest contributors
- Cost scales with $c^2 = \left(\sum_{k,j} |c_k b_j|\right)^2$

Why Faster? By Cauchy-Schwarz:

$$c^2 \leq Km \cdot \|c\|_2^2 \|b\|_2^2 \Rightarrow \text{Speedup up to factor of } Km$$

Richardson Extrapolation works for $\mathrm{Tr}[e^{iAt}\rho e^{-iAt'}O]$

Once again, we first need to show that we can use Richardson.

Richardson Extrapolation works for $\mathrm{Tr}[e^{iAt}\rho e^{-iAt'}O]$

Once again, we first need to show that we can use Richardson.

$$\left|\mathrm{Tr}\left[e^{iHt_1}\rho e^{-iHt_2}O\right] - \sum_{j=1}^{m}\sum_{r=1}^{m} b_j b_r \mathrm{Tr}\left[\mathcal{P}^{1/s_j}(s_j t_k)\rho\left(\mathcal{P}^{1/s_r}(s_r t_l)\right)^{\dagger} O\right]\right| =$$

$$\left|\mathrm{Tr}\left[Z_1\left(e^{-iHt_2} - \sum_{r=1}^{m}\mathcal{P}^{1/s_r}(s_r t_l)^{\dagger}\right)\right]\right| + \left|\mathrm{Tr}\left[\left(\sum_{j=1}^{m}\mathcal{P}^{1/s_j}(s_j t_k) - e^{iHt_1}\right) Z_2\right]\right.$$

Richardson Extrapolation works for $\mathrm{Tr}[e^{iAt}\rho e^{-iAt'}O]$

Once again, we first need to show that we can use Richardson.

$$\left|\mathrm{Tr}\left[e^{iHt_1}\rho e^{-iHt_2}O\right] - \sum_{j=1}^{m}\sum_{r=1}^{m} b_j b_r \mathrm{Tr}\left[\mathcal{P}^{1/s_j}(s_j t_k)\rho\left(\mathcal{P}^{1/s_r}(s_r t_l)\right)^{\dagger} O\right]\right| =$$

$$\left|\mathrm{Tr}\left[Z_1\left(e^{-iHt_2} - \sum_{r=1}^{m}\mathcal{P}^{1/s_r}(s_r t_l)^{\dagger}\right)\right]\right| + \left|\mathrm{Tr}\left[\left(\sum_{j=1}^{m}\mathcal{P}^{1/s_j}(s_j t_k) - e^{iHt_1}\right)Z_2\right]\right.$$

Lemma (Gate Complexity to estimate $\mathrm{Tr}(e^{iAt_k}\rho e^{-iAt_l}O)$)

We can estimate $\mathrm{Tr}(e^{iAt_k}\rho e^{-iAt_l}O)$ with Richardson error $\varepsilon_R \leq \varepsilon$ with gate complexity

$$C_{gate} = \mathcal{O}(\Gamma \cdot (\log(1/\varepsilon))(\log(c/\varepsilon)) \cdot (a_{\max}\Upsilon\lambda_{\mathrm{comm}} t_{max})^{1+\frac{1}{p}})$$

Estimating $\mathrm{Tr}(e^{iAt_k}\rho e^{-iAt_l}O)$

Use the same techniques as before (Hoeffding + Importance Sampling)

Estimating $\mathrm{Tr}(e^{iAt_k}\rho e^{-iAt_l}O)$

Use the same techniques as before (Hoeffding + Importance Sampling)

Lemma (Sample Complexity to estimate $\mathrm{Tr}(f(A)\rho f(A)^\dagger O)$)

Suppose $\|O\| \leq 1$. Then, to estimate

$$\mathsf{Tr}[f(A)\rho f(A)^\dagger O]$$

to additive error ε with failure probability at most δ, the number of samples required satisfies

$$C_{\mathrm{sample}} = \mathcal{O}\left(\frac{c^4 \cdot (\log\log(1/\varepsilon))^4}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right),$$

Putting it all together

Theorem (Estimating $\mathrm{Tr}(f(A)\rho(f(A))^\dagger O)$ with Richardson)

To estimate $\mathrm{Tr}[f(A)\rho(f(A))^\dagger O]$ *with error* $\leq \varepsilon$ *and success probability at least* $1-\delta$ *using the randomized Richardson-extrapolated method, the resource costs are:*

- **Gate complexity (per sample):**

$$C_{gate} = \mathcal{O}\left(\Gamma \cdot (\log(1/\varepsilon))^2 (\log(c(\varepsilon/3)/\varepsilon)) \cdot (a_{\max} \Upsilon \lambda_{\mathrm{comm}} t_{max}(\varepsilon/3))^{1+\frac{1}{p}}\right)$$

- **Sample complexity:**

$$C_{\mathrm{sample}} = \mathcal{O}\left(\frac{c(\varepsilon/3)^4 (\log\log(1/\varepsilon))^4}{\varepsilon^2} \log\left(\frac{1}{\delta}\right)\right)$$