Classical Algorithms for Constant Approximation of the Ground State Energy of Local Hamiltonians

François Le Gall
Nagoya University

Proceedings of the 2025 European Symposium on Algorithms (ESA 2025)
arXiv:2410.21833

Shenzhen-Nagoya Workshop on Quantum Science 2025

Statement of our First Result

Consider a $O(1)$-local Hamiltonian H acting on n qubits

→ a Hermitian $2^n \times 2^n$ matrix H with nice "sparsity" properties (in particular, it can be described in $\mathrm{poly}(n)$ bits)

Let $\lambda_0(H)$ denote the smallest eigenvalue of H (the "ground energy")

Estimating $\lambda_0(H)$ is a central problem in quantum complexity theory and computational chemistry

First main result:

> For any constant $\varepsilon > 0$, there exists a classical algorithm that computes with high probability an ε-relative approximation of $\lambda_0(H)$ in $2^{O(n)}$ time and $\mathrm{poly}(n)$ space.

the notation $O^*(\cdot)$ removes $\mathrm{poly}(n)$ factors

Previously:
- ✓ classical algorithm with $O^*(2^n)$ time but $O(2^n)$ space (Lanczos method)
- ✓ classical algorithm with poly(n) space but $2^{O(n \log n)}$ time (recursive Feynman method [Aaronson and Chen 2017])

⇒ we get for the first time simultaneously $2^{O(n)}$ time and $\mathrm{poly}(n)$ space

- ✓ quantum algorithm with $2^{O(n)}$ time and $\mathrm{poly}(n)$ space (phase estimation)

⇒ our algorithm matches the performance of the best quantum algorithm

	Type	Precision	Time	Space
Our algorithm	classical	constant	$2^{O(n)}$	$\mathrm{poly}(n)$
Lanczos method	classical	$1/\mathrm{poly}(n)$	$O^*(2^n)$	$O(2^n)$
Feynman method [Aaronson and Chen, 2017]	classical	$1/\mathrm{poly}(n)$	$2^{O(n \log n)}$	$\mathrm{poly}(n)$
Phase estimation	quantum	$1/\mathrm{poly}(n)$	$2^{O(n)}$	$\mathrm{poly}(n)$

First main result:

> For any constant $\varepsilon > 0$, there exists a classical algorithm that computes with high probability an ε-relative approximation of $\lambda_0(H)$ in $2^{O(n)}$ time and $\mathrm{poly}(n)$ time.

Previously:
- ✓ classical algorithm with $O^*(2^n)$ time but $O(2^n)$ space (Lanczos method)
- ✓ classical algorithm with poly(n) space but $2^{O(n \log n)}$ time (recursive Feynman method [Aaronson and Chen 2017])

➡ we get for the first time simultaneously $2^{O(n)}$ time and $\mathrm{poly}(n)$ space

- ✓ quantum algorithm with $2^{O(n)}$ time and $\mathrm{poly}(n)$ space (phase estimation)

➡ our algorithm matches the performance of the best quantum algorithm

Statement of our Second Result

Assume that we additionally know a vector (a "guiding state") that has some overlap χ with the eigenspace corresponding to $\lambda_0(H)$

Main setting when considering applications to computational chemistry

Second main result:

For any constant $\varepsilon > 0$, there exists a classical algorithm that computes with high probability an ε-relative approximation of $\lambda_0(H)$ in $\mathrm{poly}(\chi^{-1}, n)$ time and $\mathrm{poly}(n)$ space.

By taking $\chi = 2^{-n}$ (e.g., taking a random vector as guiding state), we get the first result

Previously:

- ✓ classical algorithm with $n^{O(\log(\chi^{-1}))}$ time and $\mathrm{poly}(n)$ space (dequantization of the Quantum Singular Value Transformation [Gharibian and LG 2022])

  ➡ this improves the best classical algorithm

- ✓ quantum algorithm with $\mathrm{poly}(\chi^{-1}, n)$ time and $\mathrm{poly}(n)$ space (phase estimation)

  ➡ our algorithm matches the performance of the best quantum algorithm

	Type	Precision	Time	Space
Our algorithm	classical	constant	$\mathrm{poly}(\chi^{-1}, n)$	$\mathrm{poly}(n)$
Gharibian-LG	classical	constant	$n^{O(\log(\chi^{-1}))}$	$\mathrm{poly}(n)$
Phase estimation	quantum	$1/\mathrm{poly}(n)$	$\mathrm{poly}(\chi^{-1}, n)$	$\mathrm{poly}(n)$

Second main result:

> For any constant $\varepsilon > 0$, there exists a classical algorithm that computes with high probability an ε-relative approximation of $\lambda_0(H)$ in $\mathrm{poly}(\chi^{-1}, n)$ time and $\mathrm{poly}(n)$ space.

By taking $\chi = 2^{-n}$ (e.g., taking a random vector as guiding state), we get the first result.

Previously:

- ✓ classical algorithm with $n^{O(\log(\chi^{-1}))}$ time and $\mathrm{poly}(n)$ space (dequantization of the Quantum Singular Value Transformation [Gharibian and LG 2022])

  ➡ this improves the best classical algorithm

- ✓ quantum algorithm with $\mathrm{poly}(\chi^{-1}, n)$ time and $\mathrm{poly}(n)$ space (phase estimation)

  ➡ our algorithm matches the performance of the best quantum algorithm

A Few Details about the Setting and Notations

- ✓ We write the $O(1)$-local Hamiltonian H as

$$H = \sum_{i=1}^{m} H_i$$

where $m = \mathrm{poly}(n)$ and each H_i is a $2^n \times 2^n$ matrix containing at most $s = O(1)$ non-zero entries in each row and column

- ✓ We normalize the Hamiltonian so that $\|H\| \leq 1$ (all the eigenvalues are then in [-1,1])

- ✓ We discussing classical algorithms using the guiding state, we assume that we have "sample-and-query" access to it, as in all prior works on dequantization (e.g., [Tang 2019])

- ✓ Given a vector $u \in \mathbb{C}^{2^n}$, we write by $u^\dagger$ his conjugate transpose

  Given two vectors $u, v \in \mathbb{C}^{2^n}$, the quantity $u^\dagger v$ corresponds to their inner product

Proof Overview

Estimate $\lambda_0(H)$

⇓ Eigenvalue estimation via polynomial transformation

Compute $u^\dagger P(H) u$ for some vector u

⇓ Trivial

Compute $u^\dagger H^r u$ for each $r \in \{0, \ldots, d\}$

⇓ Sampling (our main technical contribution)

Compute $u^\dagger H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

⇓ [Tang 2019]

Compute one entry of $H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

Iterated matrix multiplication

Proof Overview

Estimate $\lambda_0(H)$

↓ Eigenvalue estimation via polynomial transformation

Compute $u^\dagger P(H) u$ for some vector u

↓ Trivial

Compute $u^\dagger H^r u$ for each $r \in \{0, \dots, d\}$

↓ Sampling (our main technical contribution)

Compute $u^\dagger H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \dots, d$)

↓ [Tang 2019]

Compute one entry of $H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \dots, d$)

Iterated matrix multiplication

Eigenvalues Estimation via Polynomial Transformation

(Standard technique in works on the Quantum Singular Transformation)

Consider the case of distinguishing if $\lambda_0 \leq \mathrm{a}$ or $\lambda_0 \geq \mathrm{b}$ for $-1 \leq \mathrm{a} < \mathrm{b} \leq 1$ $(\mathrm{b} - \mathrm{a} = \Omega(1))$

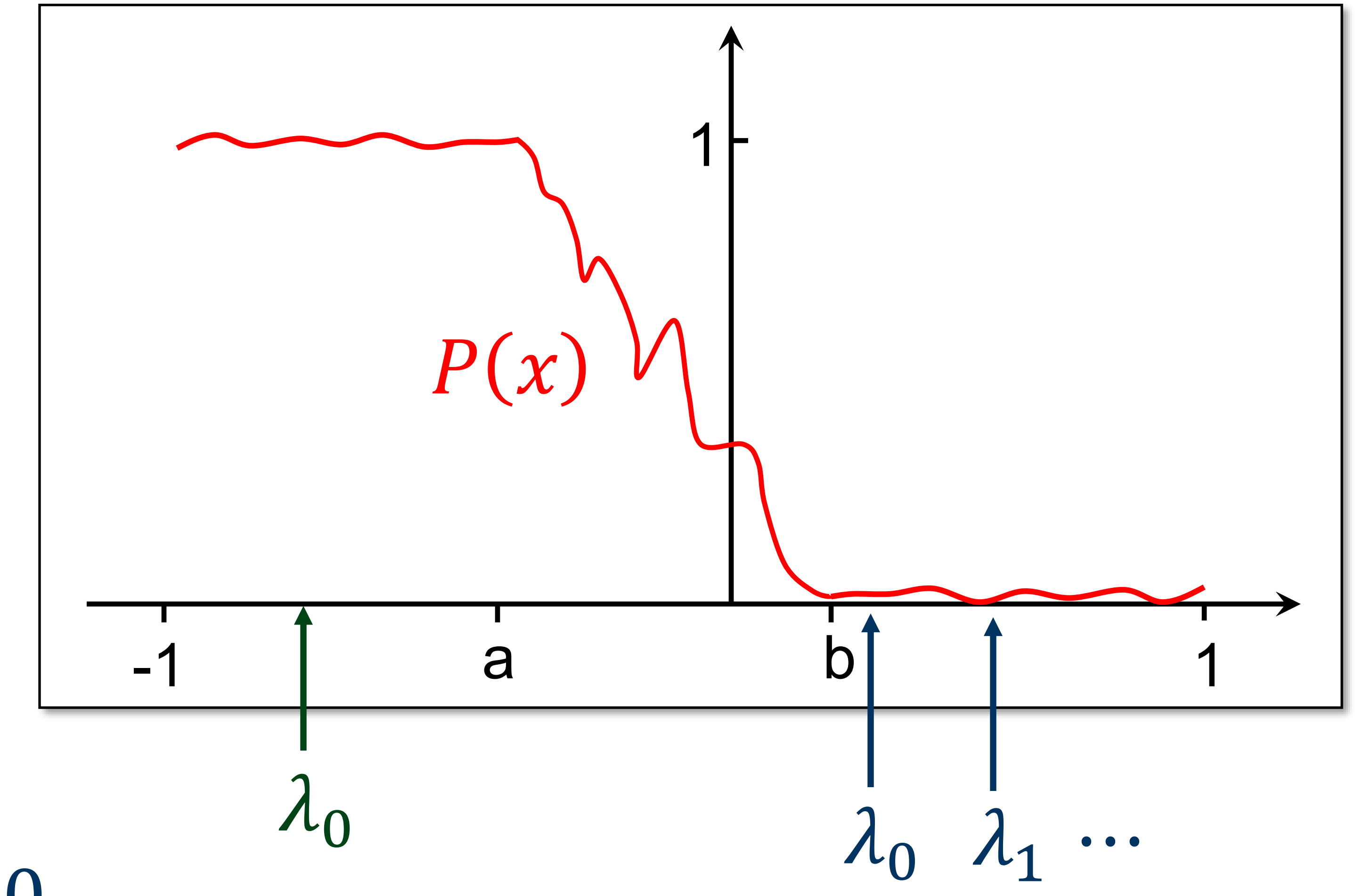
Consider the (unknown) spectral decomposition of H :

$H \equiv \mathrm{diag}(\lambda_0, \lambda_1, \dots, \lambda_{2^n-1})$ where $-1 \leq \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{2^n-1} \leq 1$ are the eigenvalues of H

The idea is to take a (low degree) polynomial $P \in \mathbb{R}$[x] such that $P(x) \in [0,1]$ for all $x \in [-1,1]$ and

$$\begin{cases} P(x) \approx 1 \text{ if } x \in [-1,a] \\ P(x) \approx 0 \text{ if } x \in [b,1] \end{cases}$$

"approximation of the step function"



If $\lambda_0 \geq \mathrm{b}$, then $P(H) \equiv \mathrm{diag}(P(\lambda_0), P(\lambda_1), \dots, P(\lambda_{2^n-1})) \approx 0$

If $\lambda_0 \leq \mathrm{a}$, then we have $P(\lambda_0) \approx 1$ and thus $P(H) \cong \mathrm{diag}(1, P(\lambda_1), \dots, P(\lambda_{2^n-1}))$

Eigenvalues Estimation via Polynomial Transformation

More generally, for any vector $u \in \mathbb{C}^{2^n}$ we have

$$\begin{cases} u^\dagger P(H) u \approx 0 & \text{if } \lambda_0 \in [\mathrm{b}, 1] \\ u^\dagger P(H) u \geq \chi & \text{if } \lambda_0 \in [-1, \mathrm{a}] \end{cases}$$

where χ is the overlap between u and the eigenspace corresponding to λ_0

Goal: compute $u^\dagger P(H) u$ for some vector u

inner product of u and $P(H)u$

Write $P(x) = a_0 + a_1 x + \cdots a_d\, x^d$

Then $u^\dagger P(H) u = \sum_{r=0}^{d} a_r\, u^\dagger H^r u$

Goal: compute $u^\dagger H^r u$ for each $r \in \{0, \ldots, d\}$

Which vector u?

✓ for our first result this will be a random vector

✓ for our second result this will be the guiding state

If $\lambda_0 \geq \mathrm{b}$, then $P(H) \equiv \mathrm{diag}(P(\lambda_0), P(\lambda_1), \ldots, P(\lambda_{2^n-1})) \approx 0$

If $\lambda_0 \leq \mathrm{a}$, then we have $P(\lambda_0) \approx 1$ and thus $P(H) \cong \mathrm{diag}(1, P(\lambda_1), \ldots, P(\lambda_{2^n-1}))$

Proof Overview

Estimate $\lambda_0(H)$

⇓ Eigenvalue estimation via polynomial transformation

Compute $u^\dagger P(H) u$ for some vector u

⇓ Trivial

Compute $u^\dagger H^r u$ for each $r \in \{0, \ldots, d\}$

⇓ Sampling (our main technical contribution)

Compute $u^\dagger H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

⇓ [Tang 2019]

Compute one entry of $H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

Iterated matrix multiplication

Computing $u^\dagger H^r u$ (the inner product of u and $H^r u$)

We have $u^\dagger H^r u = u^\dagger \left(\sum_{i=1}^{m} H_i \right)^r u$

$$H = \sum_{i=1}^{m} H_i$$

Consider the probability distribution $q\colon \{1, \dots, m\}^r \to [0,1]$ defined as

$$q(x) = \|H_{x_1}\| \cdots \|H_{x_r}\| \quad \text{for each } x = (x_1, \dots, x_\mathrm{r}) \in \{1, \dots, m\}^r$$

Consider the random variable $\dfrac{u^\dagger H_{x_1} \cdots H_{x_r} u}{q(x)}$ (here x is sampled from q)

Expectation: $\sum_x q(x) \dfrac{u^\dagger H_{x_1} \cdots H_{x_r} u}{q(x)} = \sum_x u^\dagger H_{x_1} \cdots H_{x_r} u = u^\dagger H^r u$

Variance: small

Taking the mean of a small number of samples gives a good estimate

Goal: compute $u^\dagger H_{x_1} \cdots H_{x_r} u$

Proof Overview

Estimate $\lambda_0(H)$

⇓ Eigenvalue estimation via polynomial transformation

Compute $u^\dagger P(H) u$ for some vector u

⇓ Trivial

Compute $u^\dagger H^r u$ for each $r \in \{0, \ldots, d\}$

⇓ Sampling (our main technical contribution)

Compute $u^\dagger H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

⇓ [Tang 2019]

Compute one entry of $H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

Iterated matrix multiplication

Computing $u^\dagger H_{x_1} \cdots H_{x_r} u$

Theorem (Tang 2019): For any vectors $u, v \in \mathbb{R}^n$, a good estimate of $u^\dagger v$ can be efficiently computed given sample-and-query access to u and query-access to v

We do have sample-and-query access to u (by assumption)

Goal: implement query-access to $H_{x_1} \cdots H_{x_r} u$, i.e.,
given $i \in 1, \ldots, 2^n$, compute the i-th entry of $H_{x_1} \cdots H_{x_r} u$

This is iterated matrix multiplication

The key property we can use is that each matrix $H_{x_1}, \cdots, H_{x_r}$ has at most $s = O(1)$ non-zero entries in each row/column

A careful recursive implementation then leads to time complexity $O^*(s^r)$ and space complexity $\mathrm{poly}(n)$

Proof Overview

H: $2^n \times 2^n$ matrix
s: sparsity of each H_i
d: degree of P
ε : precision

Estimate $\lambda_0(H)$

⇓ Eigenvalue estimation via polynomial transformation

Compute $u^\dagger P(H) u$ for some vector u

⇓ Trivial

Compute $u^\dagger H^r u$ for each $r \in \{0, \ldots, d\}$

⇓ Sampling (our main technical contribution)

Compute $u^\dagger H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

⇓ [Tang 2019]

Compute one entry of $H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

Iterated matrix multiplication: $O^*(s^0 + s^1 + \cdots + s^d)$ time and $\mathrm{poly}(n)$ space

(s^0: r=0, s^1: r=1, s^d: r=d)

Total complexity: $O^*(s^d \cdot d)$ time and $\mathrm{poly}(n)$ space

Deriving the Second Result

H: $2^n \times 2^n$ matrix
s: sparsity of each H_i
d: degree of P
ε : precision

χ : overlap between the guiding state u and the eigenspace corresponding to λ_0

Estimate $\lambda_0(H)$

⇩ Eigenvalue estimation via polynomial transformation

Taking $d = O(\log(1/\chi)/\varepsilon))$ is enough

Compute $u^\dagger P(H) u$ for ~~some vector~~ the guiding state u

⇩ Trivial

Compute $u^\dagger H^r u$ for each $r \in \{0, \dots, d\}$

⇩ Sampling (our main technical contribution)

Compute $u^\dagger H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \dots, d$)

⇩ [Tang 2019]

Compute one entry of $H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \dots, d$)

Iterated matrix multiplication

Total complexity: $O^*(s^d \cdot d)$ time and $\mathrm{poly}(n)$ space

$\mathrm{poly}(\chi^{-1}, n)$ time when s and ε are constant

Deriving the First Result

H: $2^n \times 2^n$ matrix
s: sparsity of each H_i
d: degree of P
ε : precision

χ : overlap with eigenspace corresponding to λ_0

Taking $d = O(n/\varepsilon)$ is enough nough

Estimate $\lambda_0(H)$

⇓ Eigenvalue estimation via polynomial transformation

Compute $u^\dagger P(H) u$ for ~~some vector~~ a random vector u

$\chi \gtrsim 2^{-n}$

⇓ Trivial

Compute $u^\dagger H^r u$ for each $r \in \{0, \ldots, d\}$

⇓ Sampling (our main technical contribution)

Compute $u^\dagger H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

⇓ [Tang 2019]

Compute one entry of $H_{x_1} \cdots H_{x_r} u$ (for $r = 0, \ldots, d$)

Iterated matrix multiplication

Total complexity: $O^*(s^d \cdot d)$ time and $\mathrm{poly}(n)$ space

$2^{O(n)}$ time when s and ε are constant

Conclusion

- ✓ We constructed classical algorithms approximating the ground-energy of a local Hamiltonian for two settings
  - without guiding state: exponential time but polynomial space complexity
  - with guiding state: time complexity depends on the overlap parameter χ
- ✓ In both settings, for constant precision, our algorithms improve previous classical algorithms and match the performance of quantum algorithms
- ✓ Our main insight is to use sampling, exploiting the fact that a local Hamiltonian is a sum of extremely sparse matrices

Main open question: Are our algorithms practical?

Can they be used in computational chemistry or computational physics when only a rough approximation is needed?