Quantum Spectral Method for Gradient and Hessian Estimation

Changpeng Shao

Academy of Mathematics and Systems Science, Chinese Academy of Sciences
joint work with Yuxin Zhang arXiv:2407.03833

The 3rd CCF Quantum Computation Conference
3-4 August 2024

Two motivating quotes

Figure 1: Gian-Carlo Rota: Every mathematician has only a few tricks.

Figure 2: George Pólya: An idea which can be used once is a trick. If it can be used more than once it becomes a method.

Quantum computing

- Quantum Fourier Transform (QFT) plays an important in the design of quantum algorithms, for example,

  Deutsch-Jozsa algorithm, Bernstein-Vazirani algorithm, Simon's algorithm, Shor's algorithm, Quantum phase estimation algorithm, the HHL algorithm for linear systems, ...

Quantum computing

- Quantum Fourier Transform (QFT) plays an important in the design of quantum algorithms, for example,

  Deutsch-Jozsa algorithm, Bernstein-Vazirani algorithm, Simon's algorithm, Shor's algorithm, Quantum phase estimation algorithm, the HHL algorithm for linear systems, ...

- In this talk, we are considering the application of QFT at the estimation of gradient vector and Hessian matrix of multivariate functions.

Why estimate gradient and Hessian?

- Useful in many optimisation algorithms, such as gradient descent and Newton's algorithm. So corresponding results can be used to speed up optimisation problems. For example,
  - quantum speedup of convex optimisation
    [van Apeldoorn, Gilyén, Gribling, de Wolf (QIP 2019)]
    [Chakrabarti, Childs, Li, Wu (QIP 2019)]
  - quantum speedup of linear programming
    [Apers, Gribling (QIP 2024)]

Why estimate gradient and Hessian?

- Useful in many optimisation algorithms, such as gradient descent and Newton's algorithm. So corresponding results can be used to speed up optimisation problems. For example,
  - quantum speedup of convex optimisation
    [van Apeldoorn, Gilyén, Gribling, de Wolf (QIP 2019)]
    [Chakrabarti, Childs, Li, Wu (QIP 2019)]
  - quantum speedup of linear programming
    [Apers, Gribling (QIP 2024)]
- Subroutine of many other quantum algorithms. For example,
  - quantum advantage of shallow circuits
    [Bravyi, Gosset, Koenig (Science 2018)]
  - quantum tomography
    [van Apeldoorn, Cornelissen, Gilyén, Nannicini (SODA 2023)]
  - quantum algorithm for estimating multiple expectation values
    [Huggins, Wan, McClean, O'Brien, Wiebe, Babbush (PRL 129, 240501)]

Problem statement: quantum gradient estimation

Problem (Gradient estimation)

Let $\varepsilon \in (0,1)$ *be the accuracy. Given a differentiable function* $f : \mathbb{R}^d \to \mathbb{R}$*, and oracle to query* f*, compute an approximate gradient* $\boldsymbol{g} \in \mathbb{R}^d$ *such that* $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \le \varepsilon$*.*

Problem statement: quantum gradient estimation

Problem (Gradient estimation)

Let $\varepsilon \in (0,1)$ *be the accuracy. Given a differentiable function* $f : \mathbb{R}^d \to \mathbb{R}$*, and oracle to query* f*, compute an approximate gradient* $\boldsymbol{g} \in \mathbb{R}^d$ *such that* $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \le \varepsilon$*.*

- Oracle: $O_f : |x\rangle \to e^{if(x)}|x\rangle$.

Problem statement: quantum gradient estimation

Problem (Gradient estimation)

Let $\varepsilon \in (0,1)$ be the accuracy. Given a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, and oracle to query f, compute an approximate gradient $\boldsymbol{g} \in \mathbb{R}^d$ such that $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \leq \varepsilon$.

- Oracle: $O_f : |x\rangle \to e^{if(x)}|x\rangle$.
- Query complexity: the minimal number of calls to O_f to find $\boldsymbol{g}$.

Problem statement: quantum gradient estimation

Problem (Gradient estimation)

Let $\varepsilon \in (0,1)$ be the accuracy. Given a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, and oracle to query f, compute an approximate gradient $\boldsymbol{g} \in \mathbb{R}^d$ such that $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \leq \varepsilon$.

- Oracle: $O_f : |x\rangle \to e^{if(x)}|x\rangle$.
- Query complexity: the minimal number of calls to O_f to find $\boldsymbol{g}$.

A quantum query algorithm of complexity T has the form: $U_T O_f \cdots U_1 O_f U_0$. Usually, $U_0, U_1, \ldots, U_T$ are simple gates, so the gate complexity $\approx$ query complexity.

Problem statement: quantum gradient estimation

Problem (Gradient estimation)

Let $\varepsilon \in (0,1)$ be the accuracy. Given a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, and oracle to query f, compute an approximate gradient $\boldsymbol{g} \in \mathbb{R}^d$ such that $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \le \varepsilon$.

- Oracle: $O_f : |x\rangle \to e^{if(x)}|x\rangle$.
- Query complexity: the minimal number of calls to O_f to find $\boldsymbol{g}$.

A quantum query algorithm of complexity T has the form: $U_T O_f \cdots U_1 O_f U_0$. Usually, $U_0, U_1, \ldots, U_T$ are simple gates, so the gate complexity $\approx$ query complexity.

In addition, for query complexity, we can prove nontrivial lower bounds. So it can be used to separate quantum and classical computing.

Previous results: Jordan's algorithm (PRL 2005)

In 2005, Jordan proposed a fast quantum algorithm for numerical gradient estimation.

Previous results: Jordan's algorithm (PRL 2005)

In 2005, Jordan proposed a fast quantum algorithm for numerical gradient estimation.

For illustration, consider the special case that $f(\boldsymbol{x}) = \boldsymbol{g} \cdot \boldsymbol{x}$ is a real-valued linear function for some unknown vector $\boldsymbol{g} = (g_1, \ldots, g_d) \in \mathbb{Z}_N^d$.

Previous results: Jordan's algorithm (PRL 2005)

In 2005, Jordan proposed a fast quantum algorithm for numerical gradient estimation.

For illustration, consider the special case that $f(\boldsymbol{x}) = \boldsymbol{g} \cdot \boldsymbol{x}$ is a real-valued linear function for some unknown vector $\boldsymbol{g} = (g_1, \ldots, g_d) \in \mathbb{Z}_N^d$.

Given a phase oracle $O_f : |\boldsymbol{x}\rangle \to e^{2\pi i f(\boldsymbol{x})/N}|\boldsymbol{x}\rangle$, we can generate the quantum state:

Previous results: Jordan's algorithm (PRL 2005)

In 2005, Jordan proposed a fast quantum algorithm for numerical gradient estimation.

For illustration, consider the special case that $f(\boldsymbol{x}) = \boldsymbol{g} \cdot \boldsymbol{x}$ is a real-valued linear function for some unknown vector $\boldsymbol{g} = (g_1, \ldots, g_d) \in \mathbb{Z}_N^d$.

Given a phase oracle $O_f : |\boldsymbol{x}\rangle \to e^{2\pi i f(\boldsymbol{x})/N}|\boldsymbol{x}\rangle$, we can generate the quantum state:

$$\frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{x} \in \mathbb{Z}_N^d} e^{\frac{2\pi i f(\boldsymbol{x})}{N}} |\boldsymbol{x}\rangle = \bigotimes_{j=1}^{d} \frac{1}{\sqrt{N}} \sum_{x_j \in \mathbb{Z}_N} e^{\frac{2\pi i g_j x_j}{N}} |x_j\rangle.$$

Previous results: Jordan's algorithm (PRL 2005)

In 2005, Jordan proposed a fast quantum algorithm for numerical gradient estimation.

For illustration, consider the special case that $f(\boldsymbol{x}) = \boldsymbol{g} \cdot \boldsymbol{x}$ is a real-valued linear function for some unknown vector $\boldsymbol{g} = (g_1, \ldots, g_d) \in \mathbb{Z}_N^d$.

Given a phase oracle $O_f : |\boldsymbol{x}\rangle \to e^{2\pi i f(\boldsymbol{x})/N} |\boldsymbol{x}\rangle$, we can generate the quantum state:

$$\frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{x} \in \mathbb{Z}_N^d} e^{\frac{2\pi i f(\boldsymbol{x})}{N}} |\boldsymbol{x}\rangle = \bigotimes_{j=1}^{d} \frac{1}{\sqrt{N}} \sum_{x_j \in \mathbb{Z}_N} e^{\frac{2\pi i g_j x_j}{N}} |x_j\rangle.$$

We will obtain $|g_1, \ldots, g_d\rangle$ by applying the inverse QFT to the above state.

Previous results: Jordan's algorithm (PRL 2005)

The query complexity of this algorithm is $O(1)$ when $f(\boldsymbol{x})$ is close to a linear function $\boldsymbol{g} \cdot \boldsymbol{x}$, which is exponentially faster than classical algorithms.

Previous results: Jordan's algorithm (PRL 2005)

The query complexity of this algorithm is $O(1)$ when $f(\boldsymbol{x})$ is close to a linear function $\boldsymbol{g} \cdot \boldsymbol{x}$, which is exponentially faster than classical algorithms.

Theorem (Jordan)

Let $\boldsymbol{g} \in \mathbb{R}^d$ and $a, \varepsilon \in \mathbb{R}_+$. Suppose we have $\tilde{f}$ such that

$$|\tilde{f}(\boldsymbol{x}) - \boldsymbol{g} \cdot \boldsymbol{x}| \leq \frac{\varepsilon a}{8 \cdot 42\pi}, \tag{1}$$

and oracle $\mathrm{O} : |\boldsymbol{x}\rangle \to e^{2\pi i 2^{n_\varepsilon} \tilde{f}(\boldsymbol{x})}|\boldsymbol{x}\rangle$, where $2^{n_\varepsilon} = 4/a\varepsilon$. Then $\widetilde{O}(1)$ queries to O can gives us $\tilde{\boldsymbol{g}} \in \mathbb{R}^d$ such that

$$\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\|_\infty \leq \varepsilon.$$

Previous results: Jordan's algorithm (PRL 2005)

The query complexity of this algorithm is $O(1)$ when $f(\boldsymbol{x})$ is close to a linear function $\boldsymbol{g} \cdot \boldsymbol{x}$, which is exponentially faster than classical algorithms.

Theorem (Jordan)

Let $\boldsymbol{g} \in \mathbb{R}^d$ and $a, \varepsilon \in \mathbb{R}_+$. Suppose we have $\tilde{f}$ such that

$$|\tilde{f}(\boldsymbol{x}) - \boldsymbol{g} \cdot \boldsymbol{x}| \leq \frac{\varepsilon a}{8 \cdot 42\pi}, \tag{1}$$

and oracle $\mathrm{O} : |\boldsymbol{x}\rangle \to e^{2\pi i 2^{n_\varepsilon} \tilde{f}(\boldsymbol{x})}|\boldsymbol{x}\rangle$, where $2^{n_\varepsilon} = 4/a\varepsilon$. Then $\widetilde{O}(1)$ queries to O can gives us $\tilde{\boldsymbol{g}} \in \mathbb{R}^d$ such that

$$\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\|_\infty \leq \varepsilon.$$

Consequently, for any function f, as long as we can construct $\tilde{f}$ such that the condition (1) holds for $\boldsymbol{g} = \nabla f(\mathbf{0})$, then we can approximate $\nabla f(\mathbf{0})$.

Previous results: Gilyén-Arunachalam-Wiebe's algorithm (SODA 2019)

In 2019, Gilyén, Arunachalam, and Wiebe developed Jordan's algorithm and improved the efficiency using higher-order finite difference methods:

$$\nabla f(\mathbf{0}) \cdot \boldsymbol{x} \approx \sum_{\ell=1}^{m} \frac{(-1)^{\ell-1}}{\ell} \frac{\binom{m}{\ell}}{\binom{m+\ell}{\ell}} \Big(f(\ell \boldsymbol{x}) - f(-\ell \boldsymbol{x}) \Big).$$

Previous results: Gilyén-Arunachalam-Wiebe's algorithm (SODA 2019)

In 2019, Gilyén, Arunachalam, and Wiebe developed Jordan's algorithm and improved the efficiency using higher-order finite difference methods:

$$\nabla f(\mathbf{0}) \cdot \boldsymbol{x} \approx \sum_{\ell=1}^{m} \frac{(-1)^{\ell-1}}{\ell} \frac{\binom{m}{\ell}}{\binom{m+\ell}{\ell}} \Big(f(\ell \boldsymbol{x}) - f(-\ell \boldsymbol{x}) \Big).$$

If $|\partial^{\alpha} f| \leq c^k k^{k/2}$ for all $k \in \mathbb{N}$ and $\alpha \in \mathbb{N}_0^d$ with $|\alpha| = k$, then error is bounded by

$$\sum_{k=2m+1}^{\infty} \left(8acm\sqrt{d} \right)^k$$

for most $\boldsymbol{x} \in aG_n^d$. Here G_n is the grid point that discretises $(-1/2, 1/2)$.

Previous results: Gilyén-Arunachalam-Wiebe's algorithm (SODA 2019)

In 2019, Gilyén, Arunachalam, and Wiebe developed Jordan's algorithm and improved the efficiency using higher-order finite difference methods:

$$\nabla f(\mathbf{0}) \cdot \boldsymbol{x} \approx \sum_{\ell=1}^{m} \frac{(-1)^{\ell-1}}{\ell} \frac{\binom{m}{\ell}}{\binom{m+\ell}{\ell}} \Big(f(\ell \boldsymbol{x}) - f(-\ell \boldsymbol{x}) \Big).$$

If $|\partial^\alpha f| \leq c^k k^{k/2}$ for all $k \in \mathbb{N}$ and $\alpha \in \mathbb{N}_0^d$ with $|\alpha| = k$, then error is bounded by

$$\sum_{k=2m+1}^{\infty} \left(8acm\sqrt{d} \right)^k$$

for most $\boldsymbol{x} \in aG_n^d$. Here G_n is the grid point that discretises $(-1/2, 1/2)$. To satisfy condition (1), it suffices to choose a such that $a^{-1} = \widetilde{O}(\sqrt{d})$. So the query complexity is $\widetilde{O}(\sqrt{d}/\varepsilon)$.

Previous results: Gilyén-Arunachalam-Wiebe's algorithm (SODA 2019)

In 2019, Gilyén, Arunachalam, and Wiebe developed Jordan's algorithm and improved the efficiency using higher-order finite difference methods:

$$\nabla f(\mathbf{0}) \cdot \boldsymbol{x} \approx \sum_{\ell=1}^{m} \frac{(-1)^{\ell-1}}{\ell} \frac{\binom{m}{\ell}}{\binom{m+\ell}{\ell}} \Big(f(\ell \boldsymbol{x}) - f(-\ell \boldsymbol{x}) \Big).$$

If $|\partial^\alpha f| \leq c^k k^{k/2}$ for all $k \in \mathbb{N}$ and $\alpha \in \mathbb{N}_0^d$ with $|\alpha| = k$, then error is bounded by

$$\sum_{k=2m+1}^{\infty} \left(8acm\sqrt{d} \right)^k$$

for most $\boldsymbol{x} \in aG_n^d$. Here G_n is the grid point that discretises $(-1/2, 1/2)$. To satisfy condition (1), it suffices to choose a such that $a^{-1} = \widetilde{O}(\sqrt{d})$. So the query complexity is $\widetilde{O}(\sqrt{d}/\varepsilon)$. They also proved the optimality of this result.

Main results

We propose a new quantum algorithm for gradient estimation for another class of functions, achieving exponential speedups over classical algorithms.

Main results

We propose a new quantum algorithm for gradient estimation for another class of functions, achieving exponential speedups over classical algorithms.

Theorem (Our result, informal)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be analytic and well defined at a neighborhood of $\mathbf{0}$ in the complex field. Then there exists a quantum algorithm that computes an approximate $\boldsymbol{g} \in \mathbb{R}^d$ such that $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \leq \varepsilon$, using $\widetilde{O}(1/\varepsilon)$ queries to phase oracles.

Main results

We propose a new quantum algorithm for gradient estimation for another class of functions, achieving exponential speedups over classical algorithms.

Theorem (Our result, informal)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be analytic and well defined at a neighborhood of $\mathbf{0}$ in the complex field. Then there exists a quantum algorithm that computes an approximate $\boldsymbol{g} \in \mathbb{R}^d$ such that $\|\boldsymbol{g} - \nabla f(\mathbf{0})\|_\infty \leq \varepsilon$, using $\widetilde{O}(1/\varepsilon)$ queries to phase oracles.

Oracle settings:

Phase oracles O_{f_1}, O_{f_2} for real and imaginary parts of $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + i f_2(\boldsymbol{x})$:

$$O_{f_1} : |\boldsymbol{x}\rangle \to e^{i f_1(\boldsymbol{x})}|\boldsymbol{x}\rangle, \quad O_{f_2} : |\boldsymbol{x}\rangle \to e^{i f_2(\boldsymbol{x})}|\boldsymbol{x}\rangle.$$

Key ideas

Different from previous algorithms, our algorithm employs the spectral method to approximate gradient $\nabla f(\mathbf{0})$.

Key ideas

Different from previous algorithms, our algorithm employs the spectral method to approximate gradient $\nabla f(\mathbf{0})$.

Consider a univariate function $f(x)$ that is sufficiently smooth such that it can be represented by the Taylor series in $\overline{B}(x_0, r) \subset \mathbb{C}$, i.e.

$$f(x) = \sum_{n=0}^{\infty} a_n (x - x_0)^n, \quad a_n = \frac{f^{(n)}(x_0)}{n!}, \quad |a_n| \leq \max_{x \in \overline{B}(x_0, r)} |f(x)| r^{-n}.$$

Key ideas

Different from previous algorithms, our algorithm employs the spectral method to approximate gradient $\nabla f(\mathbf{0})$.

Consider a univariate function $f(x)$ that is sufficiently smooth such that it can be represented by the Taylor series in $\overline{B}(x_0, r) \subset \mathbb{C}$, i.e.

$$f(x) = \sum_{n=0}^{\infty} a_n (x - x_0)^n, \quad a_n = \frac{f^{(n)}(x_0)}{n!}, \quad |a_n| \leq \max_{x \in \overline{B}(x_0, r)} |f(x)| r^{-n}.$$

Then for $\delta \in (0, r), N \in \mathbb{N}$, and $\omega = e^{-2\pi i/N}$, we have

$$f(x_0 + \delta\omega^k) = \sum_{n=0}^{\infty} a_n (\delta\omega^k)^n = \sum_{n=0}^{N-1} \omega^{kn} c_n,$$

where $c_n = \sum_{m=0}^{\infty} a_{n+mN} \delta^{n+mN}$.

Key ideas

To understand how to obtain derivative information from c_n, let us consider c_1 and c_2 as examples.

$$c_1 = a_1\delta + a_{1+N}\delta^{1+N} + a_{1+2N}\delta^{1+2N} + \cdots$$

and there is a constant $\kappa > 0$ such that $|a_n| \leq \kappa r^{-n}$ for all $n \in \mathbb{N}$, which implies that series $\{a_n\}$ decreases faster than r^{-n}.

Key ideas

To understand how to obtain derivative information from c_n, let us consider c_1 and c_2 as examples.

$$c_1 = a_1\delta + a_{1+N}\delta^{1+N} + a_{1+2N}\delta^{1+2N} + \cdots$$

and there is a constant $\kappa > 0$ such that $|a_n| \leq \kappa r^{-n}$ for all $n \in \mathbb{N}$, which implies that series $\{a_n\}$ decreases faster than r^{-n}.

Adding the fact that $\delta < r$, we can conclude that c_1/δ is close to $a_1 = f'(x_0)$.

Key ideas

To understand how to obtain derivative information from c_n, let us consider c_1 and c_2 as examples.

$$c_1 = a_1\delta + a_{1+N}\delta^{1+N} + a_{1+2N}\delta^{1+2N} + \cdots$$

and there is a constant $\kappa > 0$ such that $|a_n| \leq \kappa r^{-n}$ for all $n \in \mathbb{N}$, which implies that series $\{a_n\}$ decreases faster than r^{-n}.

Adding the fact that $\delta < r$, we can conclude that c_1/δ is close to $a_1 = f'(x_0)$.

The same holds for c_2 as

$$c_2 = a_2\delta^2 + a_{2+N}\delta^{2+N} + a_{2+2N}\delta^{2+2N} + \cdots$$

Similarly, c_2/δ^2 is close to $a_2 = f''(x_0)/2$.

Key ideas

Now we have $f_k := f(x_0 + \delta\omega^k) = \sum_{n=0}^{N-1} \omega^{kn} c_n$. The inverse discrete Fourier transform gives us

$$c_n = \frac{1}{N} \sum_{k=0}^{N-1} \omega^{-kn} f_k,$$

which means that we obtain c_n from function values f_k.

Key ideas

Now we have $f_k := f(x_0 + \delta\omega^k) = \sum_{n=0}^{N-1} \omega^{kn} c_n$. The inverse discrete Fourier transform gives us

$$c_n = \frac{1}{N} \sum_{k=0}^{N-1} \omega^{-kn} f_k,$$

which means that we obtain c_n from function values f_k.

The error is bounded by

$$\left| a_n - \frac{c_n}{\delta^n} \right| \leq \kappa r^{-n} \sum_{m=1}^{\infty} (\delta/r)^{mN} = \kappa r^{-n} \frac{(\delta/r)^N}{1 - (\delta/r)^N}$$

for some constant κ.

Key ideas

Now we have $f_k := f(x_0 + \delta\omega^k) = \sum_{n=0}^{N-1} \omega^{kn} c_n$. The inverse discrete Fourier transform gives us

$$c_n = \frac{1}{N} \sum_{k=0}^{N-1} \omega^{-kn} f_k,$$

which means that we obtain c_n from function values f_k.

The error is bounded by

$$\left| a_n - \frac{c_n}{\delta^n} \right| \leq \kappa r^{-n} \sum_{m=1}^{\infty} (\delta/r)^{mN} = \kappa r^{-n} \frac{(\delta/r)^N}{1 - (\delta/r)^N}$$

for some constant κ. Particularly, when $n = 1$,

$$\left| f'(0) - \frac{c_1}{\delta} \right| \leq \kappa r^{-1} \frac{(\delta/r)^N}{1 - (\delta/r)^N}.$$

Key ideas

For multivariable function $f(\boldsymbol{x})$, we consider $h(\tau) = f(\tau\boldsymbol{x})$. Then $h'(0) = \nabla f(0) \cdot \boldsymbol{x}$. Using the above analysis, we derive a real-valued function $F(\boldsymbol{x})$

$$F(\boldsymbol{x}) = \frac{1}{N\delta} \sum_{k=0}^{N-1} \omega^{-k} f(\delta\omega^k \boldsymbol{x})$$

that gives an approximation for $\nabla f(\mathbf{0}) \cdot \boldsymbol{x}$.

Key ideas

For multivariable function $f(\boldsymbol{x})$, we consider $h(\tau) = f(\tau\boldsymbol{x})$. Then $h'(0) = \nabla f(0) \cdot \boldsymbol{x}$. Using the above analysis, we derive a real-valued function $F(\boldsymbol{x})$

$$F(\boldsymbol{x}) = \frac{1}{N\delta} \sum_{k=0}^{N-1} \omega^{-k} f(\delta\omega^k \boldsymbol{x})$$

that gives an approximation for $\nabla f(\mathbf{0}) \cdot \boldsymbol{x}$.

When $N = O(\log(\kappa/\varepsilon))$, we can get an additive error ε, which is significantly smaller than the one obtained by finite difference formulas. This is a crucial point in our work.

Key ideas

For multivariable function $f(\boldsymbol{x})$, we consider $h(\tau) = f(\tau\boldsymbol{x})$. Then $h'(0) = \nabla f(0) \cdot \boldsymbol{x}$. Using the above analysis, we derive a real-valued function $F(\boldsymbol{x})$

$$F(\boldsymbol{x}) = \frac{1}{N\delta} \sum_{k=0}^{N-1} \omega^{-k} f(\delta\omega^k \boldsymbol{x})$$

that gives an approximation for $\nabla f(\mathbf{0}) \cdot \boldsymbol{x}$.

When $N = O(\log(\kappa/\varepsilon))$, we can get an additive error ε, which is significantly smaller than the one obtained by finite difference formulas. This is a crucial point in our work.

Combining this with previous results, we obtain a quantum algorithm that estimates gradient $\nabla f(\mathbf{0})$ with query complexity $\widetilde{O}(1/\varepsilon)$.

Differences

Difference 1. The functions used to approximate $\nabla f(\mathbf{0}) \cdot \boldsymbol{x}$.

- GAW deal with analytic real-valued functions $f : \mathbb{R}^d \to \mathbb{R}$ with specific smoothness conditions. They use the degree-$2m$ central difference approximation.
- We deal with analytic complex-valued functions $f : \mathbb{C}^d \to \mathbb{C}$ such that $f(\mathbb{R}^d) \subset \mathbb{R}$. We derive the approximating formula using the spectral method.

Differences

Difference 1. The functions used to approximate $\nabla f(\mathbf{0}) \cdot \boldsymbol{x}$.

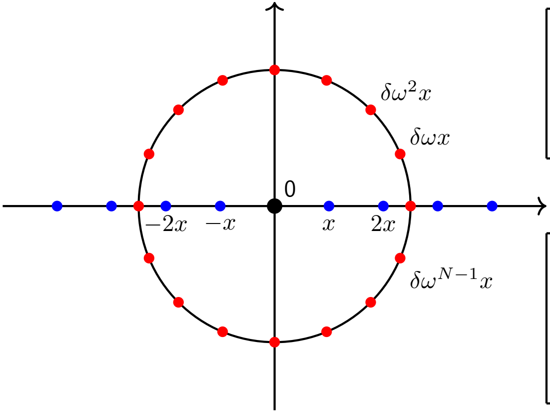
- GAW deal with analytic real-valued functions $f : \mathbb{R}^d \to \mathbb{R}$ with specific smoothness conditions. They use the degree-$2m$ central difference approximation.
- We deal with analytic complex-valued functions $f : \mathbb{C}^d \to \mathbb{C}$ such that $f(\mathbb{R}^d) \subset \mathbb{R}$. We derive the approximating formula using the spectral method.

Remark

The spectral method also outperforms finite difference method for solving ODEs on a quantum computer. [Childs, Liu, CMP 2019], [Berry J. Phys. A 2010]

Differences

Our method offers better error performance in approximating derivatives. In our opinion, one of the key reasons is the way sampling points are selected. As illustrated for 1-dimensional functions.



Spectral method (our):

$$\frac{1}{N\delta}\sum_{k=0}^{N-1}\omega^{-k}f(\delta\omega^k x) \approx f'(0)x$$

Finite difference method [GAW]:

$$\sum_{\substack{\ell=-m \\ \ell\neq 0}}^{m}\frac{(-1)^{\ell-1}}{\ell}\,\frac{\binom{m}{|\ell|}}{\binom{m+|\ell|}{|\ell|}}f(\ell x) \approx f'(0)x$$

Differences

Difference 2. The oracle settings.

- GAW assumes access to phase oracle O_f: $|\boldsymbol{x}\rangle \to e^{if(\boldsymbol{x})}|\boldsymbol{x}\rangle$ for $f : \mathbb{R}^d \to \mathbb{R}$.
- We assume phase oracles for the real and imaginary parts of $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + if_2(\boldsymbol{x})$, denoted as O_{f_1}, O_{f_2}, respectively.

While our oracle assumptions differ from GAW, they are still standard and natural, resembling how data is stored in classical computing.

Quantum Hessian estimation

As a generalization, we present four quantum algorithms for estimating the Hessian: using either finite difference method or spectral for estimating dense or sparse Hessians.

Quantum Hessian estimation

As a generalization, we present four quantum algorithms for estimating the Hessian: using either finite difference method or spectral for estimating dense or sparse Hessians.

Theorem (Hessian estimation based on spectral method)

Let $f : \mathbb{C}^d \to \mathbb{C}$ be analytic and maps $\mathbb{R}^d$ to $\mathbb{R}$. Then there is a quantum algorithm that computes $\widetilde{\mathbf{H}}$ such that $\|\widetilde{\mathbf{H}} - \mathbf{H}_f(\mathbf{0})\|_{\max} \leq \varepsilon$, using $\widetilde{O}(d/\varepsilon)$ queries to O_{f_1}, O_{f_2}, where $\mathbf{H}_f(\mathbf{0})$ is the Hessian of f at $\mathbf{0}$.

Quantum Hessian estimation

As a generalization, we present four quantum algorithms for estimating the Hessian: using either finite difference method or spectral for estimating dense or sparse Hessians.

Theorem (Hessian estimation based on spectral method)

Let $f : \mathbb{C}^d \to \mathbb{C}$ be analytic and maps $\mathbb{R}^d$ to $\mathbb{R}$. Then there is a quantum algorithm that computes $\widetilde{\mathbf{H}}$ such that $\|\widetilde{\mathbf{H}} - \mathbf{H}_f(\mathbf{0})\|_{\max} \leq \varepsilon$, using $\widetilde{O}(d/\varepsilon)$ queries to O_{f_1}, O_{f_2}, where $\mathbf{H}_f(\mathbf{0})$ is the Hessian of f at $\mathbf{0}$.

- We obtain a lower bound of $\widetilde{\Omega}(d)$.

Quantum Hessian estimation

As a generalization, we present four quantum algorithms for estimating the Hessian: using either finite difference method or spectral for estimating dense or sparse Hessians.

Theorem (Hessian estimation based on spectral method)

Let $f : \mathbb{C}^d \to \mathbb{C}$ be analytic and maps $\mathbb{R}^d$ to $\mathbb{R}$. Then there is a quantum algorithm that computes $\widetilde{\mathbf{H}}$ such that $\|\widetilde{\mathbf{H}} - \mathbf{H}_f(\mathbf{0})\|_{\max} \leq \varepsilon$, using $\widetilde{O}(d/\varepsilon)$ queries to O_{f_1}, O_{f_2}, where $\mathbf{H}_f(\mathbf{0})$ is the Hessian of f at $\mathbf{0}$.

- We obtain a lower bound of $\widetilde{\Omega}(d)$.
- If $\mathbf{H}_f(\mathbf{0})$ is promised to be s-sparse, then the complexity can be reduced to $\widetilde{O}(s/\varepsilon)$.

Quantum Hessian estimation

To illustrate how to derive a quantum algorithm for Hessian estimation from a gradient estimation algorithm, we will provide a brief explanation using the simple case of $f(\boldsymbol{x}) = \langle \boldsymbol{x}|H|\boldsymbol{x}\rangle$.

Quantum Hessian estimation

To illustrate how to derive a quantum algorithm for Hessian estimation from a gradient estimation algorithm, we will provide a brief explanation using the simple case of $f(\boldsymbol{x}) = \langle \boldsymbol{x}|H|\boldsymbol{x}\rangle$.

For fixed $\boldsymbol{y} \in \mathbb{R}^d$, we can get the following linear function h using 2 queries to f

$$h_{\boldsymbol{y}}(\boldsymbol{x}) = \frac{1}{2}\left(f(\boldsymbol{x}+\boldsymbol{y}) - f(\boldsymbol{x})\right) = \langle \boldsymbol{x}|H|\boldsymbol{y}\rangle + \frac{1}{2}\langle \boldsymbol{y}|H|\boldsymbol{y}\rangle.$$

Quantum Hessian estimation

To illustrate how to derive a quantum algorithm for Hessian estimation from a gradient estimation algorithm, we will provide a brief explanation using the simple case of $f(\boldsymbol{x}) = \langle \boldsymbol{x}|H|\boldsymbol{x}\rangle$.

For fixed $\boldsymbol{y} \in \mathbb{R}^d$, we can get the following linear function h using 2 queries to f

$$h_{\boldsymbol{y}}(\boldsymbol{x}) = \frac{1}{2}\left(f(\boldsymbol{x}+\boldsymbol{y}) - f(\boldsymbol{x})\right) = \langle \boldsymbol{x}|H|\boldsymbol{y}\rangle + \frac{1}{2}\langle \boldsymbol{y}|H|\boldsymbol{y}\rangle.$$

Hence, using Jordan's algorithm, we can compute $H|\boldsymbol{y}\rangle$ for $h_{\boldsymbol{y}}(\boldsymbol{x})$. Let $\boldsymbol{y} = \boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ be the computational basis, then we can recover H using $O(d)$ times quantum gradient estimation algorithm.

Quantum Hessian estimation

To illustrate how to derive a quantum algorithm for Hessian estimation from a gradient estimation algorithm, we will provide a brief explanation using the simple case of $f(\boldsymbol{x}) = \langle \boldsymbol{x}|H|\boldsymbol{x}\rangle$.

For fixed $\boldsymbol{y} \in \mathbb{R}^d$, we can get the following linear function h using 2 queries to f

$$h_{\boldsymbol{y}}(\boldsymbol{x}) = \frac{1}{2}\left(f(\boldsymbol{x}+\boldsymbol{y}) - f(\boldsymbol{x})\right) = \langle \boldsymbol{x}|H|\boldsymbol{y}\rangle + \frac{1}{2}\langle \boldsymbol{y}|H|\boldsymbol{y}\rangle.$$

Hence, using Jordan's algorithm, we can compute $H|\boldsymbol{y}\rangle$ for $h_{\boldsymbol{y}}(\boldsymbol{x})$. Let $\boldsymbol{y} = \boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ be the computational basis, then we can recover H using $O(d)$ times quantum gradient estimation algorithm. $\Omega(d)$ queries are required even for $f(\boldsymbol{x}) = \langle \boldsymbol{x}|H|\boldsymbol{x}\rangle$.

Quantum Hessian estimation

To illustrate how to derive a quantum algorithm for Hessian estimation from a gradient estimation algorithm, we will provide a brief explanation using the simple case of $f(\boldsymbol{x}) = \langle \boldsymbol{x}|H|\boldsymbol{x}\rangle$.

For fixed $\boldsymbol{y} \in \mathbb{R}^d$, we can get the following linear function h using 2 queries to f

$$h_{\boldsymbol{y}}(\boldsymbol{x}) = \frac{1}{2}\left(f(\boldsymbol{x}+\boldsymbol{y}) - f(\boldsymbol{x})\right) = \langle \boldsymbol{x}|H|\boldsymbol{y}\rangle + \frac{1}{2}\langle \boldsymbol{y}|H|\boldsymbol{y}\rangle.$$

Hence, using Jordan's algorithm, we can compute $H|\boldsymbol{y}\rangle$ for $h_{\boldsymbol{y}}(\boldsymbol{x})$. Let $\boldsymbol{y} = \boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ be the computational basis, then we can recover H using $O(d)$ times quantum gradient estimation algorithm. $\Omega(d)$ queries are required even for $f(\boldsymbol{x}) = \langle \boldsymbol{x}|H|\boldsymbol{x}\rangle$.

When H is sparse, then randomly generate $k = O(s \log d)$ vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ are enough to recover H from $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ and $H\boldsymbol{y}_1, \ldots, H\boldsymbol{y}_k$.

Conclusions and future works

- We propose a new quantum algorithm for gradient estimation for analytic functions that can take values from $\mathbb{C}$.

Conclusions and future works

- We propose a new quantum algorithm for gradient estimation for analytic functions that can take values from $\mathbb{C}$.
- The algorithm achieves exponential speedup over classical algorithms.

Conclusions and future works

- We propose a new quantum algorithm for gradient estimation for analytic functions that can take values from $\mathbb{C}$.

- The algorithm achieves exponential speedup over classical algorithms.

- We also propose quantum algorithms for Hessian estimation, which achieve polynomial speedups over classical algorithms. Exponential speedup exist for sparse Hessians.

Conclusions and future works

- We propose a new quantum algorithm for gradient estimation for analytic functions that can take values from $\mathbb{C}$.
- The algorithm achieves exponential speedup over classical algorithms.
- We also propose quantum algorithms for Hessian estimation, which achieve polynomial speedups over classical algorithms. Exponential speedup exist for sparse Hessians.
- Applications of the gradient and Hessian estimation algorithms? E.g., sparse graph coloring problems [Gebremedhin, Manne, Pothen, SIAM Review 2005].

Conclusions and future works

- We propose a new quantum algorithm for gradient estimation for analytic functions that can take values from $\mathbb{C}$.
- The algorithm achieves exponential speedup over classical algorithms.
- We also propose quantum algorithms for Hessian estimation, which achieve polynomial speedups over classical algorithms. Exponential speedup exist for sparse Hessians.
- Applications of the gradient and Hessian estimation algorithms? E.g., sparse graph coloring problems [Gebremedhin, Manne, Pothen, SIAM Review 2005].
- Our analysis shows that finite difference method performs quite worse for Hessian estimation. Is there any way to fix this?

Conclusions and future works

- We propose a new quantum algorithm for gradient estimation for analytic functions that can take values from $\mathbb{C}$.

- The algorithm achieves exponential speedup over classical algorithms.

- We also propose quantum algorithms for Hessian estimation, which achieve polynomial speedups over classical algorithms. Exponential speedup exist for sparse Hessians.

- Applications of the gradient and Hessian estimation algorithms? E.g., sparse graph coloring problems [Gebremedhin, Manne, Pothen, SIAM Review 2005].

- Our analysis shows that finite difference method performs quite worse for Hessian estimation. Is there any way to fix this?

Thanks very much for your time!