Quantum gradient estimation

A. J. Cornelissen[1,2]

[1]Applied Mathematics
Delft University of Technology

[2]QuSoft
Centrum Wiskunde & Informatica

April 24th, 2019

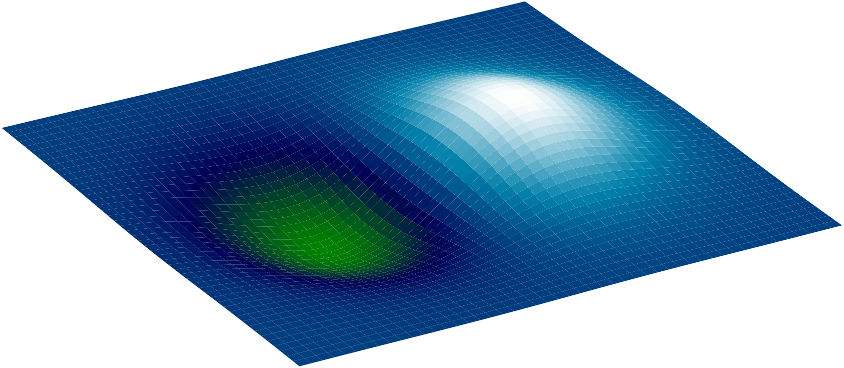
Context

Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.

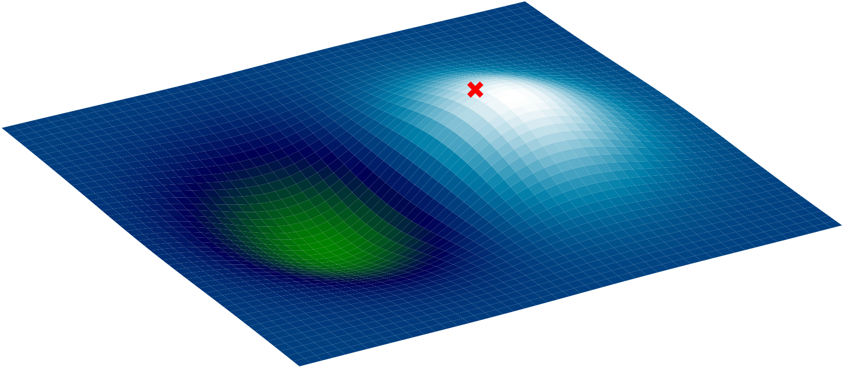
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



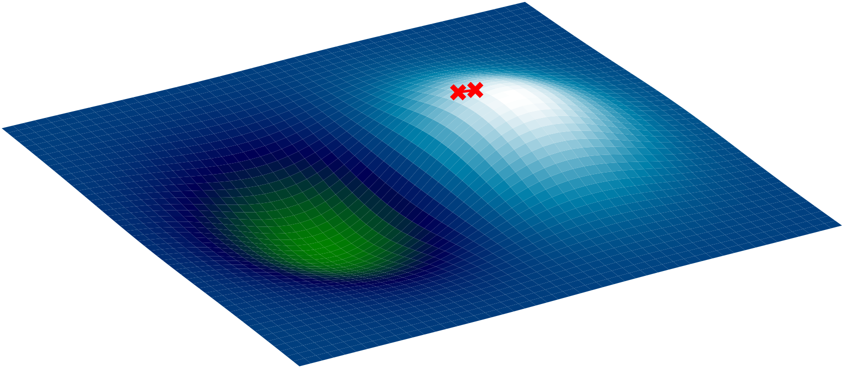
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



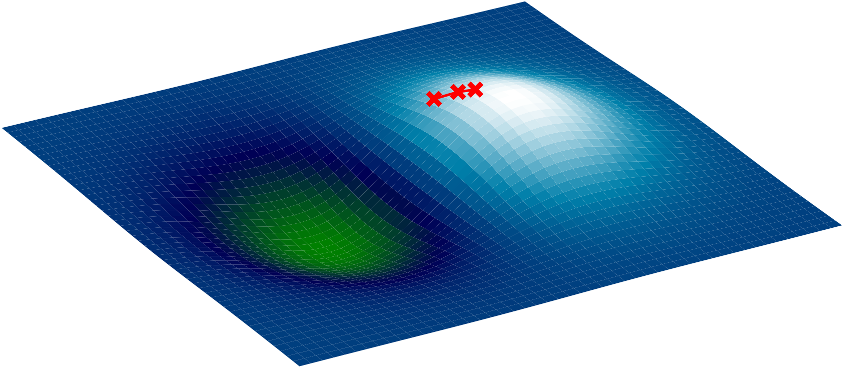
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



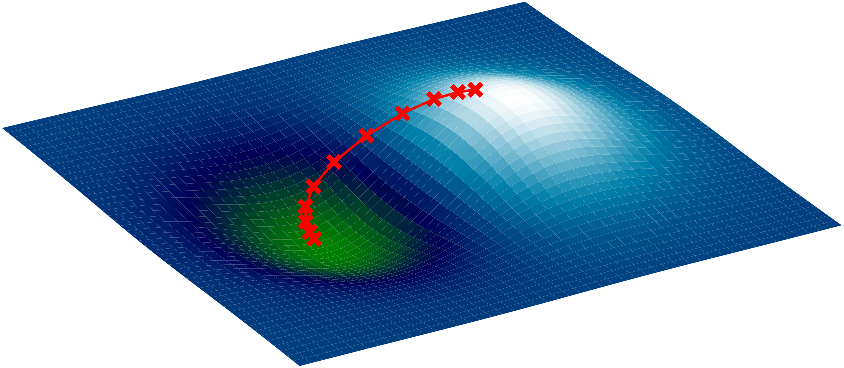
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



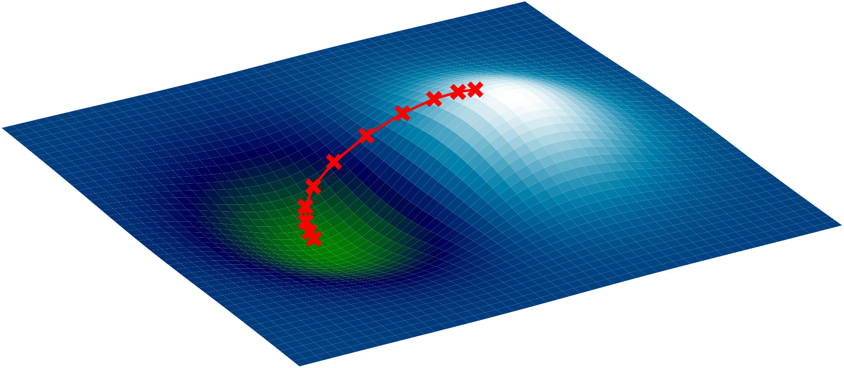
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



Can we speed up the gradient calculation step when d is large?

Classical gradient estimation

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

$$\begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(N)}) \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} a \\ g_1 \\ \vdots \\ g_d \end{bmatrix}$$

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

$$\begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(N)}) \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} a \\ g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- So, at least $d + 1$ function evaluations required classically.

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

$$\begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(N)}) \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} a \\ g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- So, at least $d+1$ function evaluations required classically.
- **Can we do better with a quantum computer?**

Contents

Contents

Contents

Contents

Contents

Contents

Visualization of quantum states

Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

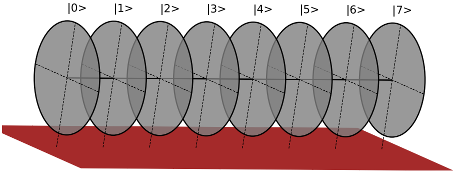
- For all j: $|\alpha_j| \leq 1$.

Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

- For all j: $|\alpha_j| \leq 1$.

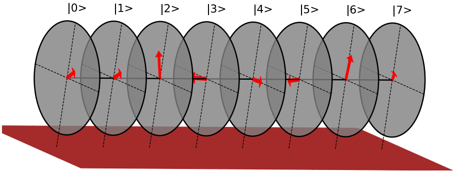


Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

- For all j: $|\alpha_j| \leq 1$.



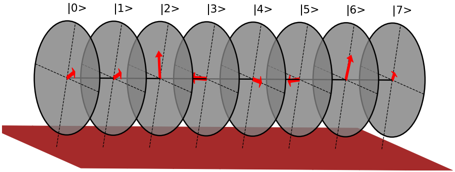
Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

- For all j: $|\alpha_j| \leq 1$.
- One can modify the state by applying unitary transformations $U \in \mathbb{C}^{2^n \times 2^n}$:

$$|\psi\rangle \mapsto U|\psi\rangle$$



Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

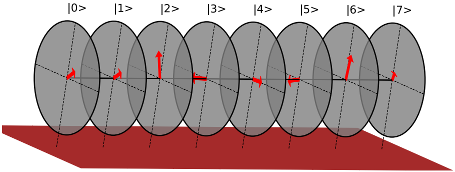
$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

- For all j: $|\alpha_j| \leq 1$.
- One can modify the state by applying unitary transformations $U \in \mathbb{C}^{2^n \times 2^n}$:

$$|\psi\rangle \mapsto U|\psi\rangle$$

- A measurement returns $j \in \{0, 1, \dots, 2^n - 1\}$ with probability:

$$\mathbb{P}(j) = |\alpha_j|^2$$



Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

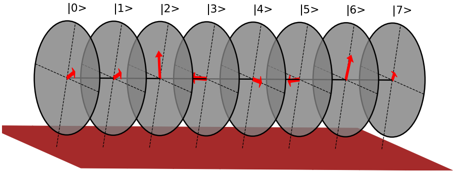
$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

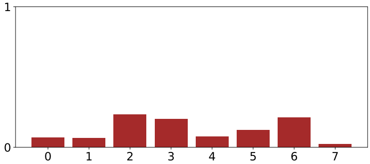
- For all j: $|\alpha_j| \leq 1$.
- One can modify the state by applying unitary transformations $U \in \mathbb{C}^{2^n \times 2^n}$:

$$|\psi\rangle \mapsto U|\psi\rangle$$

- A measurement returns $j \in \{0, 1, \ldots, 2^n - 1\}$ with probability:

$$\mathbb{P}(j) = |\alpha_j|^2$$





Quantum Fourier transform

Quantum Fourier transform

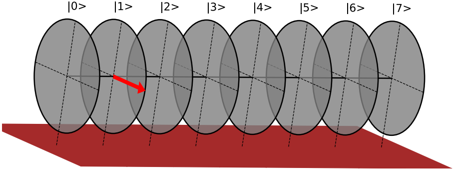
- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

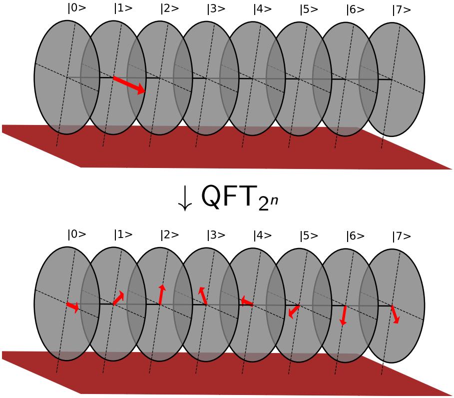
$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$



Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

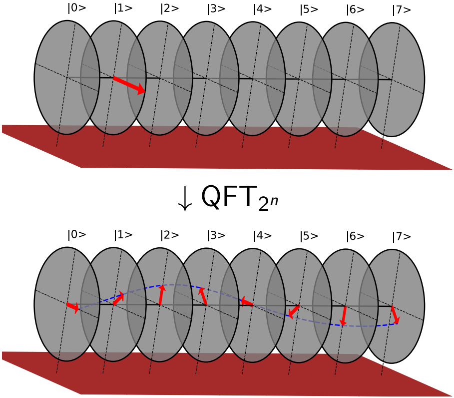
$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$



Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

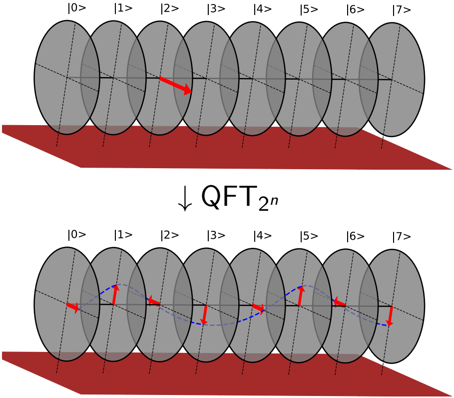
$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$



Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

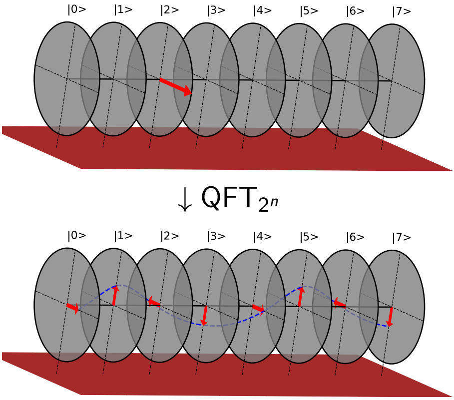


Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.



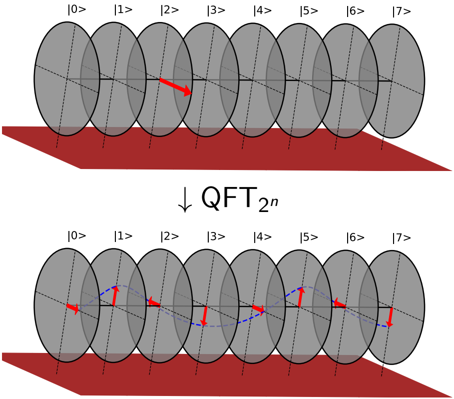
Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

$$\mathrm{QFT}_{2^n}^{\dagger} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$



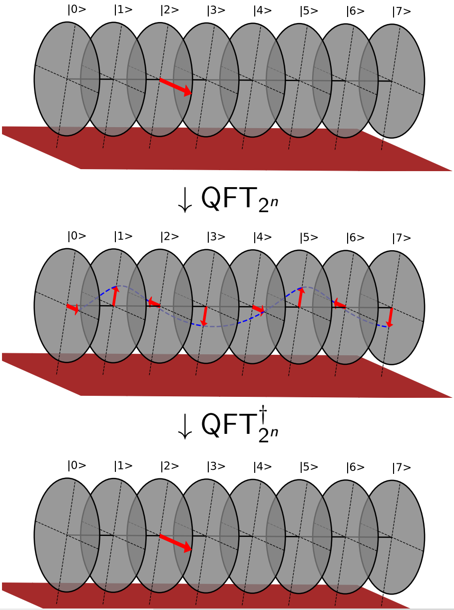
Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

$$\mathrm{QFT}_{2^n}^{\dagger} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$



Quantum Fourier transform

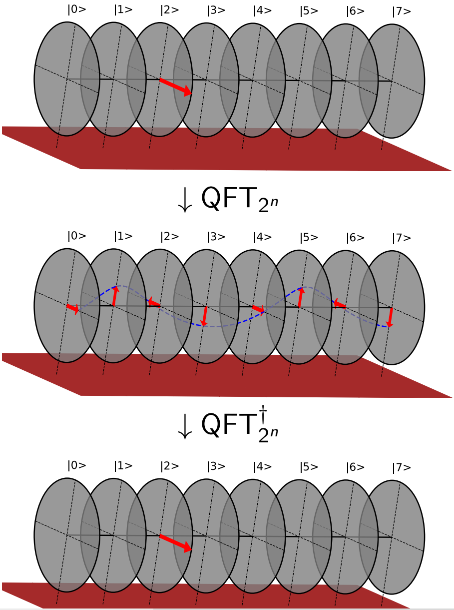
- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

$$\mathrm{QFT}_{2^n}^{\dagger} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$

- Efficient implementations available.



Quantum Fourier transform

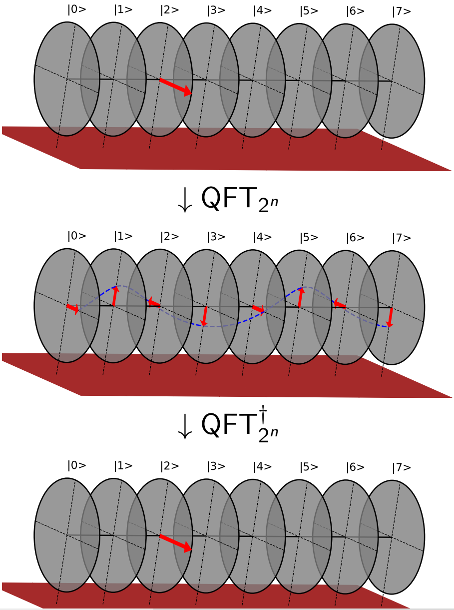
- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

$$\mathrm{QFT}^{\dagger}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$

- Efficient implementations available.
- Also works for non-integer revolutions.



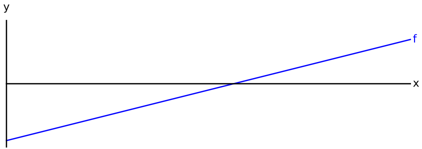
Quantum function evaluations

Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.

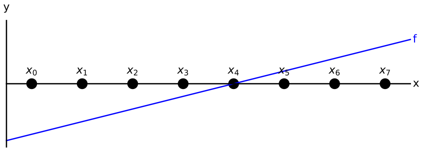
Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.



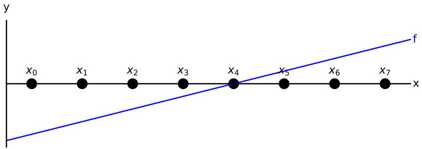
Quantum function evaluations

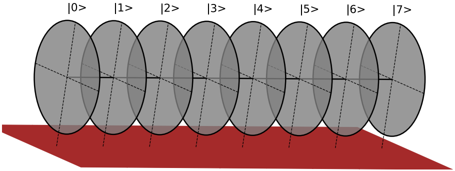
- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n-1}\} \subseteq \mathbb{R}$.



Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \dots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.

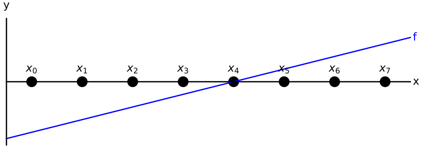


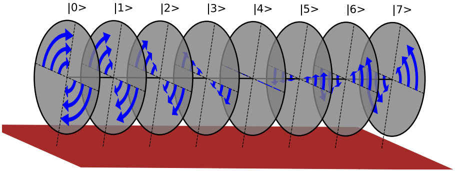


Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.
- We can evaluate f as follows:

$$O_f : |j\rangle \mapsto e^{if(x_j)} |j\rangle$$



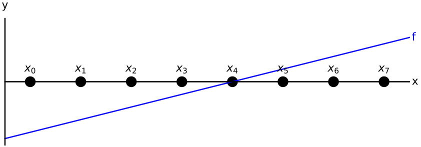


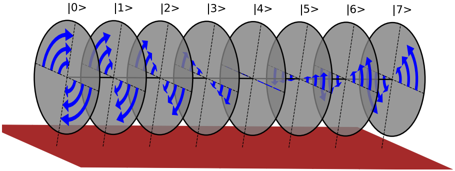
Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \dots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.
- We can evaluate f as follows:

$$O_f : |j\rangle \mapsto e^{if(x_j)} |j\rangle$$

- This is called the phase oracle of f on G.



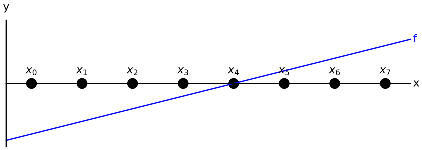


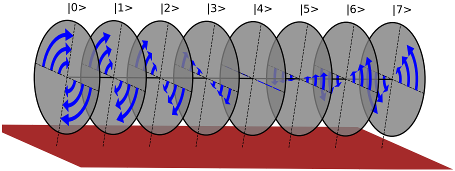
Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.
- We can evaluate f as follows:

$$O_f : |j\rangle \mapsto e^{if(x_j)} |j\rangle$$

- This is called the phase oracle of f on G.
- One application of this phase oracle is one *quantum function evaluation*.





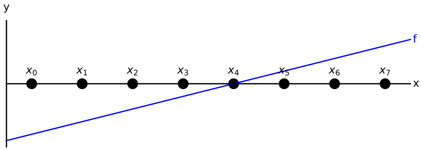
Quantum derivative estimation algorithm for linear functions

Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

Quantum derivative estimation algorithm for linear functions

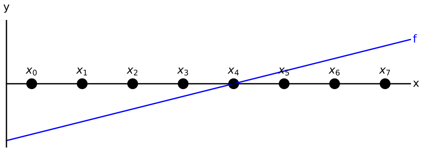
Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.



Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

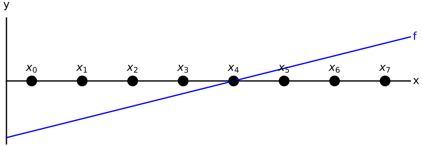
1. Create a uniform superposition over the grid.

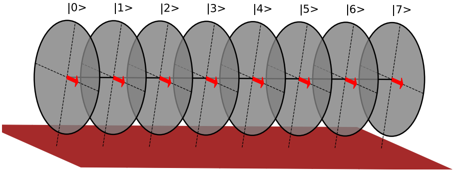


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.

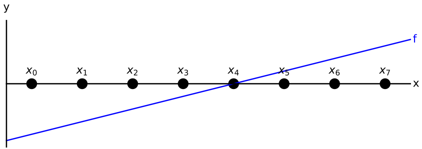


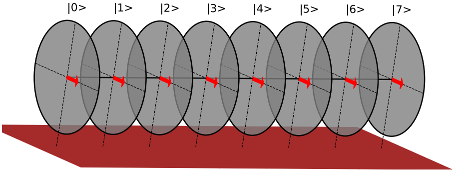


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.

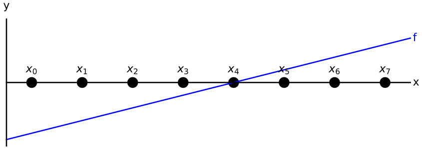


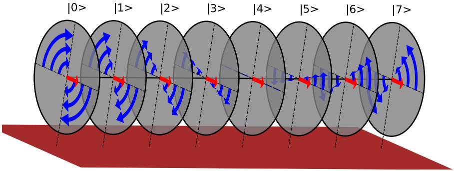


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.

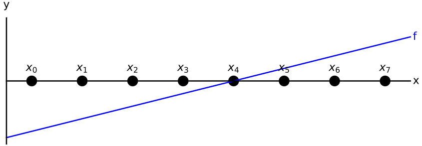


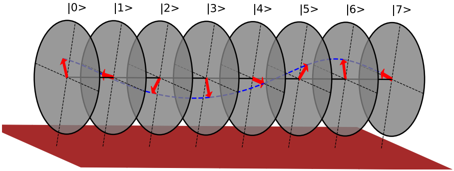


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.

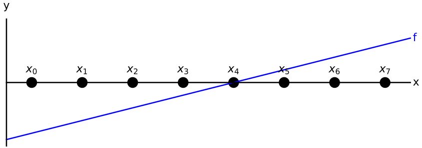


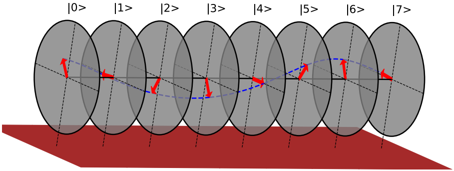


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.
3. Apply the inverse QFT.
4. Measure.

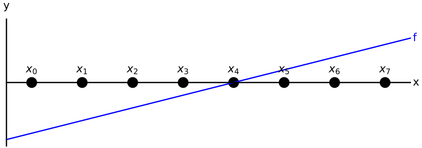


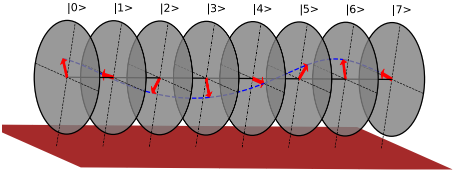


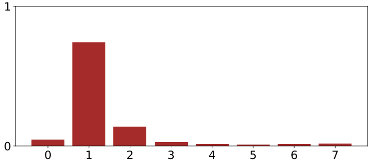
Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.
3. Apply the inverse QFT.
4. Measure.





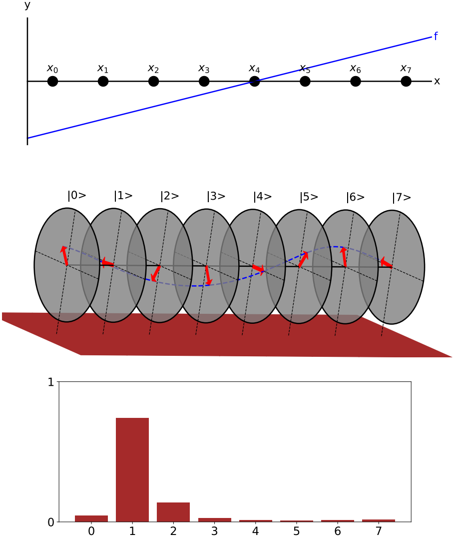


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.
3. Apply the inverse QFT.
4. Measure.

Generalizes to $f : \mathbb{R}^d \to \mathbb{R}$.
(Jordan, 2004)



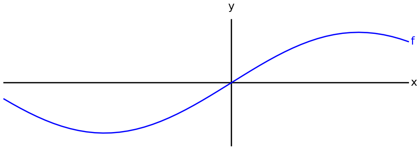
Modifications for non-linear functions

Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.

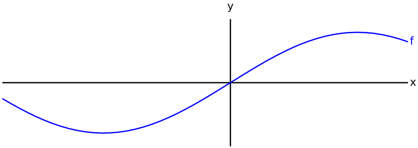
Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.



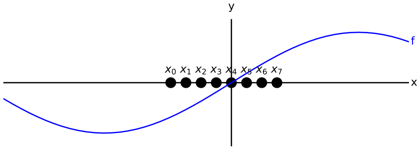
Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.



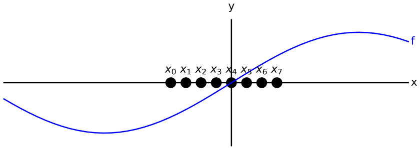
Modifications for non-linear functions

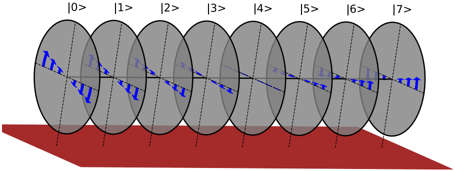
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.



Modifications for non-linear functions

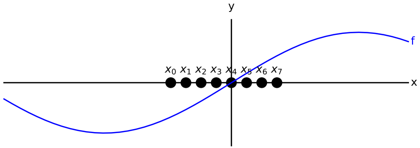
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.

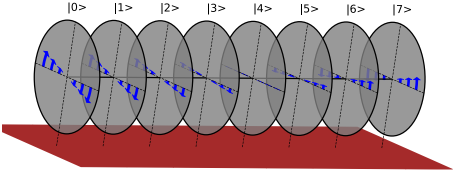




Modifications for non-linear functions

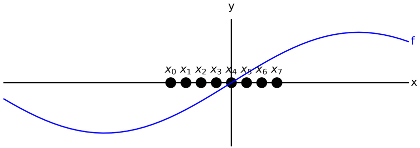
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:

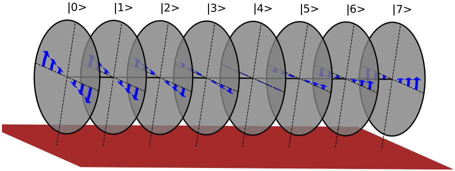




Modifications for non-linear functions

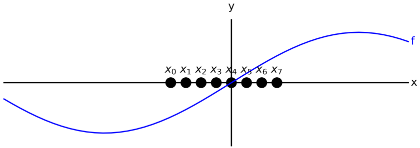
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:
  - Rotations become very small.

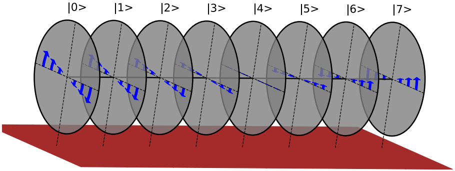




Modifications for non-linear functions

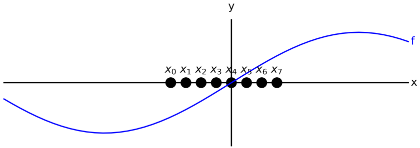
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:
  - Rotations become very small.
  - Function evaluations must be very precise.

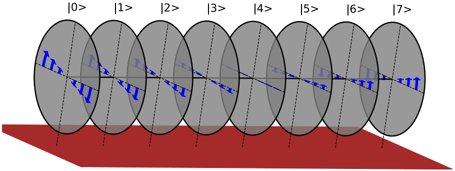




Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:
  - Rotations become very small.
  - Function evaluations must be very precise.
- Key idea: central difference method to extend region of approximate linearity.





Central difference method

Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

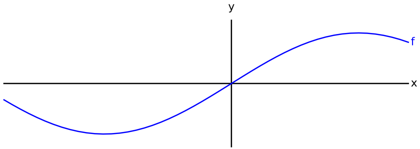
- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$

Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell\mathbf{x})$$



- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$

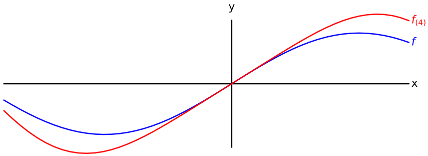
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



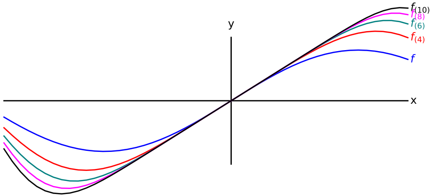
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



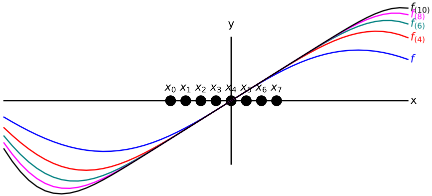
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



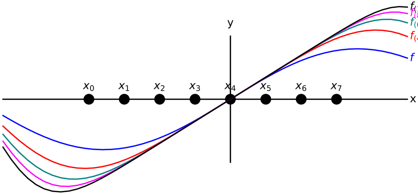
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



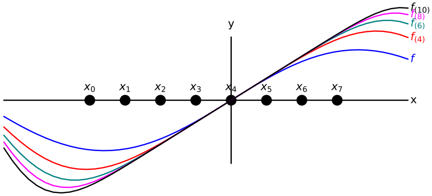
Central difference method

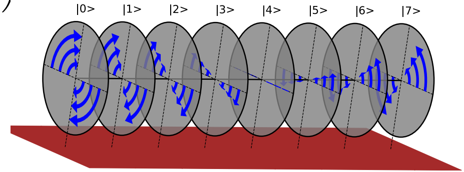
- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1} \right).$$





Central difference method

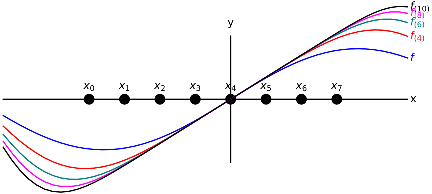
- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

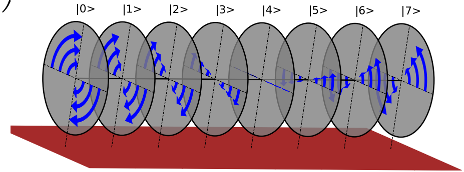
$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right).$$

- One can implement $O_{\widetilde{f}_{(2m)}}$ using $\widetilde{\mathcal{O}}(m)$ queries to O_f. (Gilyén, Arunachalam, Wiebe, 2018)





Smoothness conditions (Gilyén et al. 2018)

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
    - Reinforcement learning

Smoothness conditions (Gilyén et al. 2018)

Case 2: Gevrey

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
    - Reinforcement learning

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
  - Reinforcement learning

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
  - Reinforcement learning

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
  - Reinforcement learning

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

- $\widetilde{\mathcal{O}}(d^\sigma)$ function evaluations.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
  - Reinforcement learning

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

- $\widetilde{\mathcal{O}}(d^\sigma)$ function evaluations.
- Speed-up when $\sigma < 1$!

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
  - Reinforcement learning

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

- $\widetilde{\mathcal{O}}(d^\sigma)$ function evaluations.
- Speed-up when $\sigma < 1$!
- Suitable when objective functions are intrinsically smooth:

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
  - Reinforcement learning

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

- $\widetilde{\mathcal{O}}(d^\sigma)$ function evaluations.
- Speed-up when $\sigma < 1$!
- Suitable when objective functions are intrinsically smooth:
  - Quantum variational circuits.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- $\widetilde{\mathcal{O}}(k)$ function evaluations suffice.
- Does not depend on d!
- Suitable when good polynomial approximations are available.
  - Reinforcement learning

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

- $\widetilde{\mathcal{O}}(d^\sigma)$ function evaluations.
- Speed-up when $\sigma < 1$!
- Suitable when objective functions are intrinsically smooth:
  - Quantum variational circuits.
  - Quantum approximate optimization algorithms.

Summary and conclusions

Summary and conclusions

- Current overview for ℓ^∞-approximate gradient estimation algorithms:

Summary and conclusions

- Current overview for ℓ^∞-approximate gradient estimation algorithms:

	Polynomial	Gevrey with $\sigma \in [0, \frac{1}{2}]$	Gevrey with $\sigma \in [\frac{1}{2}, 1]$
Query complexity	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}\left(d^{\frac{1}{2}}\right)$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Lower bound	$\Omega(1)$	$\Omega\left(d^{\frac{1}{2}}\right)$	$\Omega\left(d^{\frac{1}{2}}\right)$

Summary and conclusions

- Current overview for ℓ^∞-approximate gradient estimation algorithms:

	Polynomial	Gevrey with $\sigma \in [0, \frac{1}{2}]$	Gevrey with $\sigma \in [\frac{1}{2}, 1]$
Query complexity	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}\left(d^{\frac{1}{2}}\right)$	$\widetilde{\mathcal{O}}(d^\sigma)$
Lower bound	$\Omega(1)$	$\Omega\left(d^{\frac{1}{2}}\right)$	$\Omega\left(d^{\frac{1}{2}}\right)$

- For ℓ^p-approximate gradient estimation algorithms: multiply lower and upper bound by $\Theta(d^{\frac{1}{p}})$.

Summary and conclusions

- Current overview for ℓ^∞-approximate gradient estimation algorithms:

	Polynomial	Gevrey with $\sigma \in [0, \frac{1}{2}]$	Gevrey with $\sigma \in [\frac{1}{2}, 1]$
Query complexity	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}\left(d^{\frac{1}{2}}\right)$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Lower bound	$\Omega(1)$	$\Omega\left(d^{\frac{1}{2}}\right)$	$\Omega\left(d^{\frac{1}{2}}\right)$

- For ℓ^p-approximate gradient estimation algorithms: multiply lower and upper bound by $\Theta(d^{\frac{1}{p}})$.
- Open problem: can we improve on $\widetilde{\mathcal{O}}(d)$ in the Gevrey case where $\sigma = 1$?

Summary and conclusions

- Current overview for ℓ^∞-approximate gradient estimation algorithms:

	Polynomial	Gevrey with $\sigma \in [0, \frac{1}{2}]$	Gevrey with $\sigma \in [\frac{1}{2}, 1]$
Query complexity	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}\left(d^{\frac{1}{2}}\right)$	$\widetilde{\mathcal{O}}(d^\sigma)$
Lower bound	$\Omega(1)$	$\Omega\left(d^{\frac{1}{2}}\right)$	$\Omega\left(d^{\frac{1}{2}}\right)$

- For ℓ^p-approximate gradient estimation algorithms: multiply lower and upper bound by $\Theta(d^{\frac{1}{p}})$.
- Open problem: can we improve on $\widetilde{\mathcal{O}}(d)$ in the Gevrey case where $\sigma = 1$?

Thanks for your attention!

`arjan@cwi.nl`