Quantum algorithms for solving differential equations

Ashley Montanaro

School of Mathematics, University of Bristol, and Phasecraft Ltd

6 October 2022

Talk based on joint work with:

Noah Linden and Changpeng Shao

(Comm. Math. Phys. 395, pp. 601–641, 2022)

Dong An, Noah Linden, Jin-Peng Liu, Changpeng Shao, and Jiasu Wang

(Quantum 5, 481, 2021)

University of Bristol

EPSRC Engineering and Physical Sciences Research Council

European Research Council
Established by the European Commission

Solving differential equations with a quantum computer

One plausible problem domain where quantum computers could be applied is solving differential equations, for example linear PDEs:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x_1^2} + \cdots + \frac{\partial^2 u}{\partial x_d^2} \right)$$

Solving differential equations with a quantum computer

One plausible problem domain where quantum computers could be applied is solving differential equations, for example linear PDEs:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x_1^2} + \cdots + \frac{\partial^2 u}{\partial x_d^2} \right)$$

One reason this seems plausible is that:

- PDEs are often solved by discretisation to produce a system of linear equations;
- Quantum computers could have an exponential advantage over classical computers for linear equation problems.

Solving differential equations with a quantum computer

One plausible problem domain where quantum computers could be applied is solving differential equations, for example linear PDEs:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x_1^2} + \cdots + \frac{\partial^2 u}{\partial x_d^2} \right)$$

One reason this seems plausible is that:

- PDEs are often solved by discretisation to produce a system of linear equations;
- Quantum computers could have an exponential advantage over classical computers for linear equation problems.

Some indications there could be an advantage for PDEs:
e.g. [Leyton+Osborne 0812.4423] [Berry 1010.2745] [Cao et al 1207.2485] [Clader et al 1301.2340] [Childs et al 2002.07868] ...

"Solving" linear equations

A matrix is d-sparse if it has at most d non-zero elements in each row and column.

Solving linear equations

Given access to a d-sparse $N \times N$ matrix A, and $b \in \mathbb{R}^N$, output x such that $Ax = b$.

“Solving” linear equations

A matrix is d-sparse if it has at most d non-zero elements in each row and column.

Solving linear equations

Given access to a d-sparse $N \times N$ matrix A, and $b \in \mathbb{R}^N$, output x such that $Ax = b$.

One “quantum” way of thinking about the problem:

“Solving” linear equations

Given the ability to produce the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$, and access to A as above, produce the state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

"Solving" linear equations

A matrix is d-sparse if it has at most d non-zero elements in each row and column.

Solving linear equations

Given access to a d-sparse $N \times N$ matrix A, and $b \in \mathbb{R}^N$, output x such that $Ax = b$.

One "quantum" way of thinking about the problem:

"Solving" linear equations

Given the ability to produce the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$, and access to A as above, produce the state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

Theorem: If A has condition number κ $(= \|A^{-1}\| \|A\|)$, $|x\rangle$ can be approximately produced in time $\mathrm{poly}(\log N, d, \kappa)$ [Harrow et al 0811.3171] [many others]

Some challenges for the quantum algorithm

- How to produce the initial state $|b\rangle$?

Some challenges for the quantum algorithm

- How to produce the initial state $|b\rangle$?
- How to get information out of the final state $|x\rangle$?

Some challenges for the quantum algorithm

- How to produce the initial state $|b\rangle$?
- How to get information out of the final state $|x\rangle$?
- How to access the matrix A?

Some challenges for the quantum algorithm

- How to produce the initial state $|b\rangle$?
- How to get information out of the final state $|x\rangle$?
- How to access the matrix A?
- How to bound the condition number κ?

Some challenges for the quantum algorithm

- How to produce the initial state $|b\rangle$?
- How to get information out of the final state $|x\rangle$?
- How to access the matrix A?
- How to bound the condition number κ?
- How to bound the level of accuracy achieved?

Some challenges for the quantum algorithm

- How to produce the initial state $|b\rangle$?
- How to get information out of the final state $|x\rangle$?
- How to access the matrix A?
- How to bound the condition number κ?
- How to bound the level of accuracy achieved?

Taking these into account, and making some assumptions about the problem solved, in [AM+Pallister 1512.05903] it was shown that using the HHL algorithm to solve PDEs discretised with the finite element method (FEM) can achieve at most a polynomial speedup (in fixed "spatial" dimension).

This talk

Today I will discuss two recent works applying quantum algorithms to differential equations.

First, solving the heat equation in d dimensions in the region $[0, L]^d \times [0, T]$ with periodic spatial boundary conditions:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x_1^2} + \cdots + \frac{\partial^2 u}{\partial x_d^2} \right)$$

Problem

Let $u(\mathbf{x}, t)$ be a solution to the heat equation. Given an initial condition $u(\mathbf{x}, 0) = u_0(\mathbf{x})$, a time t, and a subset $S \subseteq [0, L]^d$, compute $\int_S u(\mathbf{x}, t) d\mathbf{x} \pm \epsilon$.

Will quantum algorithms outperform classical ones for this problem?

This talk

Second, speeding up the solution of general stochastic differential equations:

$$\mathrm{d}X_t = \mu(X_t, t)\mathrm{d}t + \sigma(X_t, t)\mathrm{d}W_t$$

for $t \in [0, T]$, where X_t is an Itô process, and W_t is Brownian motion.

Problem

Given an initial distribution π_0, and a payoff function $\mathcal{P}(X)$, compute $\mathbb{E}[\mathcal{P}(X_T) \mid X_0 \in \pi_0] \pm \epsilon$.

A fundamental problem in mathematical finance, where we think of X_t as the price of some asset at time t: allows computing option prices, risks, ...

Heat equation: summary of results

We compared various classical and quantum methods for solving the heat equation:

Method	$\mathbf{d = 1}$	$\mathbf{d = 2}$	$\mathbf{d = 3}$	$\mathbf{d > 3}$
* Classical linear equations	$\widetilde{O}(\epsilon^{-2})$	$\widetilde{O}(\epsilon^{-2.5})$	$\widetilde{O}(\epsilon^{-3})$	$\widetilde{O}(\epsilon^{-d/2-1.5})$
* Classical time-stepping	$\widetilde{O}(\epsilon^{-1.5})$	$\widetilde{O}(\epsilon^{-2})$	$\widetilde{O}(\epsilon^{-2.5})$	$\widetilde{O}(\epsilon^{-d/2-1})$
* Classical FFT	$\widetilde{O}(\epsilon^{-0.5})$	$\widetilde{O}(\epsilon^{-1})$	$\widetilde{O}(\epsilon^{-1.5})$	$\widetilde{O}(\epsilon^{-d/2})$
Classical random walk	$\widetilde{O}(\epsilon^{-3})$	$\widetilde{O}(\epsilon^{-3})$	$\widetilde{O}(\epsilon^{-3})$	$\widetilde{O}(\epsilon^{-3})$
HHL	$\widetilde{O}(\epsilon^{-2.5})$	$\widetilde{O}(\epsilon^{-2.5})$	$\widetilde{O}(\epsilon^{-2.75})$	$\widetilde{O}(\epsilon^{-d/4-2})$
Diagonalisation	$\widetilde{O}(\epsilon^{-1.25})$	$\widetilde{O}(\epsilon^{-1.5})$	$\widetilde{O}(\epsilon^{-1.75})$	$\widetilde{O}(\epsilon^{-d/4-1})$
Coherent rw acceleration	$\widetilde{O}(\epsilon^{-1.75})$	$\widetilde{O}(\epsilon^{-2})$	$\widetilde{O}(\epsilon^{-2.25})$	$\widetilde{O}(\epsilon^{-d/4-1.5})$
Rw amplitude estimation	$\widetilde{O}(\epsilon^{-2})$	$\widetilde{O}(\epsilon^{-2})$	$\widetilde{O}(\epsilon^{-2})$	$\widetilde{O}(\epsilon^{-2})$

Only the dependence on the accuracy ϵ is shown.

Starred methods use space $\mathsf{poly}(1/\epsilon)$, others use space $\mathsf{poly}(\log 1/\epsilon)$. $\widetilde{O}$ notation hides log factors.

Methods

All of the classical and quantum algorithms are based on discretising space and time via the finite difference method (FTCS):

$$\frac{du}{dx} = \frac{u(x+h) - u(x)}{h} + O(h)$$

$$\frac{d^2u}{dx^2} = \frac{u(x+h) + u(x-h) - 2u(x)}{h^2} + O(h^2)$$

Methods

All of the classical and quantum algorithms are based on discretising space and time via the finite difference method (FTCS):

$$\frac{du}{dx} = \frac{u(x+h) - u(x)}{h} + O(h)$$

$$\frac{d^2u}{dx^2} = \frac{u(x+h) + u(x-h) - 2u(x)}{h^2} + O(h^2)$$

Leads to the set of linear constraints

$$\frac{\widetilde{u}(\mathbf{x},t+\Delta t) - \widetilde{u}(\mathbf{x},t)}{\Delta t} = \frac{\alpha}{\Delta x^2} \sum_{i=1}^{d} \widetilde{u}(\ldots, x_i + \Delta x, \ldots, t) + \widetilde{u}(\ldots, x_i - \Delta x, \ldots, t) - 2\widetilde{u}(\mathbf{x}, t)$$

Methods

All of the classical and quantum algorithms are based on discretising space and time via the finite difference method (FTCS):

$$\frac{du}{dx} = \frac{u(x+h) - u(x)}{h} + O(h)$$

$$\frac{d^2u}{dx^2} = \frac{u(x+h) + u(x-h) - 2u(x)}{h^2} + O(h^2)$$

Leads to the set of linear constraints

$$\frac{\widetilde{u}(\mathbf{x},t+\Delta t) - \widetilde{u}(\mathbf{x},t)}{\Delta t} = \frac{\alpha}{\Delta x^2} \sum_{i=1}^{d} \widetilde{u}(\dots, x_i + \Delta x, \dots, t) + \widetilde{u}(\dots, x_i - \Delta x, \dots, t) - 2\widetilde{u}(\mathbf{x}, t)$$

To achieve final accuracy ϵ, we can take $\Delta t = O(\epsilon)$, $\Delta x = O(\sqrt{\epsilon})$ (assuming u is sufficiently smooth, $\partial^4 u / \partial x_i^2 \partial x_j^2 = O(L^{-d})$).

Linear equation methods

We have a system of $N = O(\epsilon^{-d/2-1})$ linear equations to solve.

Linear equation methods

We have a system of $N = O(\epsilon^{-d/2-1})$ linear equations to solve.

- Condition number: $\kappa = O(\epsilon^{-1})$.

Linear equation methods

We have a system of $N = O(\epsilon^{-d/2-1})$ linear equations to solve.

- Condition number: $\kappa = O(\epsilon^{-1})$.
- Classical complexity: $\widetilde{O}(\sqrt{\kappa}N) = O(\epsilon^{-d/2-1.5})$.

Linear equation methods

We have a system of $N = O(\epsilon^{-d/2-1})$ linear equations to solve.

- Condition number: $\kappa = O(\epsilon^{-1})$.
- Classical complexity: $\widetilde{O}(\sqrt{\kappa}N) = O(\epsilon^{-d/2-1.5})$.
- Quantum complexity: $\widetilde{O}(\kappa) = \widetilde{O}(\epsilon^{-1})\ldots$

Linear equation methods

We have a system of $N = O(\epsilon^{-d/2-1})$ linear equations to solve.

- Condition number: $\kappa = O(\epsilon^{-1})$.
- Classical complexity: $\widetilde{O}(\sqrt{\kappa}N) = O(\epsilon^{-d/2-1.5})$.
- Quantum complexity: $\widetilde{O}(\kappa) = \widetilde{O}(\epsilon^{-1})$...

...but this algorithm only produces a quantum state which is equal to $\widetilde{u}/\|\widetilde{u}\|_2$.

Linear equation methods

We have a system of $N = O(\epsilon^{-d/2-1})$ linear equations to solve.

- Condition number: $\kappa = O(\epsilon^{-1})$.
- Classical complexity: $\widetilde{O}(\sqrt{\kappa}N) = O(\epsilon^{-d/2-1.5})$.
- Quantum complexity: $\widetilde{O}(\kappa) = \widetilde{O}(\epsilon^{-1})\dots$

$\dots$but this algorithm only produces a quantum state which is equal to $\widetilde{u}/\|\widetilde{u}\|_2$.

To approximate $\int_S u(\mathbf{x}, t)d\mathbf{x}$, we need to know $\|\widetilde{u}\|_2$; achieving high enough accuracy takes time $\widetilde{O}(\epsilon^{-d/4-2})$.

Other classical methods

We can rewrite the discretised heat equation as

$$\widetilde{u}(\mathbf{x}, t+\Delta t) = \left(1 - \frac{2d\alpha\Delta t}{\Delta x^2}\widetilde{u}(\mathbf{x}, t)\right) + \frac{\alpha\Delta t}{\Delta x^2}\sum_{i=1}^{d}\widetilde{u}(\dots, x_i + \Delta x, \dots, t) + \widetilde{u}(\dots, x_i - \Delta x, \dots, t).$$

Other classical methods

We can rewrite the discretised heat equation as

$$\widetilde{u}(\mathbf{x}, t+\Delta t) = \left(1 - \tfrac{2d\alpha\Delta t}{\Delta x^2}\widetilde{u}(\mathbf{x}, t)\right) + \tfrac{\alpha\Delta t}{\Delta x^2}\textstyle\sum_{i=1}^{d} \widetilde{u}(\dots, x_i + \Delta x, \dots, t) + \widetilde{u}(\dots, x_i - \Delta x, \dots, t).$$

- We can simply step forward in time using sparse matrix multiplication: time $\widetilde{O}(\epsilon^{-d/2} \cdot \epsilon^{-1})$.

Other classical methods

We can rewrite the discretised heat equation as

$$\widetilde{u}(\mathbf{x}, t+\Delta t) = \left(1 - \frac{2d\alpha\Delta t}{\Delta x^2}\widetilde{u}(\mathbf{x}, t)\right) + \frac{\alpha\Delta t}{\Delta x^2}\sum_{i=1}^{d}\widetilde{u}(\dots, x_i + \Delta x, \dots, t) + \widetilde{u}(\dots, x_i - \Delta x, \dots, t).$$

- We can simply step forward in time using sparse matrix multiplication: time $\widetilde{O}(\epsilon^{-d/2} \cdot \epsilon^{-1})$.
- We can diagonalise the discretised linear system with the FFT: time $\widetilde{O}(\epsilon^{-d/2})$.

Other classical methods

We can rewrite the discretised heat equation as

$$\widetilde{u}(\mathbf{x}, t+\Delta t) = \left(1 - \frac{2d\alpha\Delta t}{\Delta x^2}\widetilde{u}(\mathbf{x}, t)\right) + \frac{\alpha\Delta t}{\Delta x^2}\sum_{i=1}^{d}\widetilde{u}(\dots, x_i+\Delta x, \dots, t) + \widetilde{u}(\dots, x_i-\Delta x, \dots, t).$$

- We can simply step forward in time using sparse matrix multiplication: time $\widetilde{O}(\epsilon^{-d/2} \cdot \epsilon^{-1})$.
- We can diagonalise the discretised linear system with the FFT: time $\widetilde{O}(\epsilon^{-d/2})$.
- We can observe that this corresponds to a random walk and sample from the output distribution in time $\widetilde{O}(\epsilon^{-1})$.

Other classical methods

We can rewrite the discretised heat equation as

$$\widetilde{u}(\mathbf{x}, t+\Delta t) = \left(1 - \frac{2d\alpha\Delta t}{\Delta x^2}\widetilde{u}(\mathbf{x}, t)\right) + \frac{\alpha\Delta t}{\Delta x^2}\sum_{i=1}^{d}\widetilde{u}(\dots, x_i+\Delta x, \dots, t) + \widetilde{u}(\dots, x_i-\Delta x, \dots, t).$$

- We can simply step forward in time using sparse matrix multiplication: time $\widetilde{O}(\epsilon^{-d/2} \cdot \epsilon^{-1})$.
- We can diagonalise the discretised linear system with the FFT: time $\widetilde{O}(\epsilon^{-d/2})$.
- We can observe that this corresponds to a random walk and sample from the output distribution in time $\widetilde{O}(\epsilon^{-1})$.
- Gives an algorithm for approximating $\int_S u(\mathbf{x}, t)d\mathbf{x} \pm \epsilon$ in time $\widetilde{O}(\epsilon^{-1} \cdot \epsilon^{-2})$.

Other quantum methods

Analogous to the classical ones:

Other quantum methods

Analogous to the classical ones:

- We can start with an initial state $|u_0\rangle$ and try to produce the state $|u_\tau\rangle$ corresponding to τ steps.

Other quantum methods

Analogous to the classical ones:

- We can start with an initial state $|u_0\rangle$ and try to produce the state $|u_\tau\rangle$ corresponding to τ steps.
- We can use a coherent acceleration method for random walks due to [Apers+Sarlette '18], [Gilyen et al '18] which gives a square-root improvement in τ: time $\widetilde{O}(\epsilon^{-d/4-1.5})$.

Other quantum methods

Analogous to the classical ones:

- We can start with an initial state $|u_0\rangle$ and try to produce the state $|u_\tau\rangle$ corresponding to τ steps.
- We can use a coherent acceleration method for random walks due to [Apers+Sarlette '18], [Gilyen et al '18] which gives a square-root improvement in τ: time $\widetilde{O}(\epsilon^{-d/4-1.5})$.
- We can diagonalise the discretised linear system and solve the diagonalised system (postselect): time $\widetilde{O}(\epsilon^{-d/4-1})$.

Other quantum methods

Analogous to the classical ones:

- We can start with an initial state $|u_0\rangle$ and try to produce the state $|u_\tau\rangle$ corresponding to τ steps.
- We can use a coherent acceleration method for random walks due to [Apers+Sarlette '18], [Gilyen et al '18] which gives a square-root improvement in τ: time $\widetilde{O}(\epsilon^{-d/4-1.5})$.
- We can diagonalise the discretised linear system and solve the diagonalised system (postselect): time $\widetilde{O}(\epsilon^{-d/4-1})$.
- We can speed up the classical random walk using amplitude estimation.

Other quantum methods

Analogous to the classical ones:

- We can start with an initial state $|u_0\rangle$ and try to produce the state $|u_\tau\rangle$ corresponding to τ steps.
- We can use a coherent acceleration method for random walks due to [Apers+Sarlette '18], [Gilyen et al '18] which gives a square-root improvement in τ: time $\widetilde{O}(\epsilon^{-d/4-1.5})$.
- We can diagonalise the discretised linear system and solve the diagonalised system (postselect): time $\widetilde{O}(\epsilon^{-d/4-1})$.
- We can speed up the classical random walk using amplitude estimation.
- Gives an algorithm for approximating $\int_S u(\mathbf{x}, t) d\mathbf{x} \pm \epsilon$ in time $\widetilde{O}(\epsilon^{-1} \cdot \epsilon^{-1})$.

Solving stochastic differential equations

Recall that our goal is to solve a stochastic differential equation:

$$\mathrm{d}X_t = \mu(X_t, t)\mathrm{d}t + \sigma(X_t, t)\mathrm{d}W_t$$

for $t \in [0, T]$, where X_t is an Itô process, W_t is Brownian motion.

Problem

Given an initial distribution π_0, and a payoff function $\mathcal{P}(X)$, compute $\mathbb{E}[\mathcal{P}(X_T) \mid X_0 \in \pi_0] \pm \epsilon$.

A fundamental problem in mathematical finance, where we think of X_t as the price of some asset at time t: allows computing option prices, risks, ...

The Monte Carlo method

- A simple way to solve an SDE is to discretise time into $n = T/h$ steps and step the price of the asset forward in time using a discrete random walk.

The Monte Carlo method

- A simple way to solve an SDE is to discretise time into $n = T/h$ steps and step the price of the asset forward in time using a discrete random walk.

- For example, Milstein discretisation takes $h = O(\epsilon)$, leading to a cost of $O(1/\epsilon)$ per sample of X_T ($T = O(1)$).

The Monte Carlo method

- A simple way to solve an SDE is to discretise time into $n = T/h$ steps and step the price of the asset forward in time using a discrete random walk.

- For example, Milstein discretisation takes $h = O(\epsilon)$, leading to a cost of $O(1/\epsilon)$ per sample of X_T ($T = O(1)$).

- The expectation of a random variable with variance σ^2 can be estimated up to ϵ using $O(\sigma^2/\epsilon^2)$ samples.

The Monte Carlo method

- A simple way to solve an SDE is to discretise time into $n = T/h$ steps and step the price of the asset forward in time using a discrete random walk.

- For example, Milstein discretisation takes $h = O(\epsilon)$, leading to a cost of $O(1/\epsilon)$ per sample of X_T ($T = O(1)$).

- The expectation of a random variable with variance σ^2 can be estimated up to ϵ using $O(\sigma^2/\epsilon^2)$ samples.

- So we obtain a total runtime of $O(1/\epsilon^3)$ if $\sigma = O(1)$.

The Monte Carlo method

- A simple way to solve an SDE is to discretise time into $n = T/h$ steps and step the price of the asset forward in time using a discrete random walk.

- For example, Milstein discretisation takes $h = O(\epsilon)$, leading to a cost of $O(1/\epsilon)$ per sample of X_T ($T = O(1)$).

- The expectation of a random variable with variance σ^2 can be estimated up to ϵ using $O(\sigma^2/\epsilon^2)$ samples.

- So we obtain a total runtime of $O(1/\epsilon^3)$ if $\sigma = O(1)$.

- We can improve this using a technique known as Multilevel Monte Carlo [Giles '08].

The Multilevel Monte Carlo method

The general idea:

- We have a sequence of random variables $P_0, \ldots, P_L$ that approximates a random variable P with increasing accuracy and cost.
- We write $\mathbb{E}[P_L] = \sum_{i=0}^{L} \mathbb{E}[P_i - P_{i-1}]$.
- We approximate $\mathbb{E}[P_i - P_{i-1}] \pm \epsilon/L$ and take the sum.

The Multilevel Monte Carlo method

The general idea:

- We have a sequence of random variables $P_0, \ldots, P_L$ that approximates a random variable P with increasing accuracy and cost.
- We write $\mathbb{E}[P_L] = \sum_{i=0}^{L} \mathbb{E}[P_i - P_{i-1}]$.
- We approximate $\mathbb{E}[P_i - P_{i-1}] \pm \epsilon/L$ and take the sum.

For example, in the Milstein discretisation scheme, we might let P_i be the payoff when discretising with step length $h = 2^{-i}$.

The Multilevel Monte Carlo method

The general idea:

- We have a sequence of random variables $P_0, \ldots, P_L$ that approximates a random variable P with increasing accuracy and cost.
- We write $\mathbb{E}[P_L] = \sum_{i=0}^{L} \mathbb{E}[P_i - P_{i-1}]$.
- We approximate $\mathbb{E}[P_i - P_{i-1}] \pm \epsilon/L$ and take the sum.

For example, in the Milstein discretisation scheme, we might let P_i be the payoff when discretising with step length $h = 2^{-i}$.

- Gives cost $O(2^i)$ for each sample at level i

The Multilevel Monte Carlo method

The general idea:

- We have a sequence of random variables $P_0, \ldots, P_L$ that approximates a random variable P with increasing accuracy and cost.
- We write $\mathbb{E}[P_L] = \sum_{i=0}^{L} \mathbb{E}[P_i - P_{i-1}]$.
- We approximate $\mathbb{E}[P_i - P_{i-1}] \pm \epsilon/L$ and take the sum.

For example, in the Milstein discretisation scheme, we might let P_i be the payoff when discretising with step length $h = 2^{-i}$.

- Gives cost $O(2^i)$ for each sample at level i
- We have the variance bound $\mathsf{Var}(P_i - P_{i-1}) = O(2^{-i})$

The Multilevel Monte Carlo method

The general idea:

- We have a sequence of random variables $P_0, \ldots, P_L$ that approximates a random variable P with increasing accuracy and cost.
- We write $\mathbb{E}[P_L] = \sum_{i=0}^{L} \mathbb{E}[P_i - P_{i-1}]$.
- We approximate $\mathbb{E}[P_i - P_{i-1}] \pm \epsilon/L$ and take the sum.

For example, in the Milstein discretisation scheme, we might let P_i be the payoff when discretising with step length $h = 2^{-i}$.

- Gives cost $O(2^i)$ for each sample at level i
- We have the variance bound $\mathsf{Var}(P_i - P_{i-1}) = O(2^{-i})$
- Overall cost at the i'th level is $O(2^{-i}/(\epsilon/L)^2 \times 2^i) = O(L^2/\epsilon^2)$

The Multilevel Monte Carlo method

The general idea:

- We have a sequence of random variables $P_0, \ldots, P_L$ that approximates a random variable P with increasing accuracy and cost.
- We write $\mathbb{E}[P_L] = \sum_{i=0}^{L} \mathbb{E}[P_i - P_{i-1}]$.
- We approximate $\mathbb{E}[P_i - P_{i-1}] \pm \epsilon/L$ and take the sum.

For example, in the Milstein discretisation scheme, we might let P_i be the payoff when discretising with step length $h = 2^{-i}$.

- Gives cost $O(2^i)$ for each sample at level i
- We have the variance bound $\mathsf{Var}(P_i - P_{i-1}) = O(2^{-i})$
- Overall cost at the i'th level is $O(2^{-i}/(\epsilon/L)^2 \times 2^i) = O(L^2/\epsilon^2)$
- Overall cost is $O(L^3/\epsilon^2) = \widetilde{O}(1/\epsilon^2)$.

Quantum speedup of the multilevel Monte Carlo method

Theorem [AM 1504.06987] (informal)

Given the ability to generate samples from a random variable X with variance σ^2, there is a quantum algorithm which approximates $\mathbb{E}[X] \pm \epsilon$ using $\widetilde{O}(\sigma/\epsilon)$ samples from X.

Quantum speedup of the multilevel Monte Carlo method

Theorem [AM 1504.06987] (informal)

Given the ability to generate samples from a random variable X with variance σ^2, there is a quantum algorithm which approximates $\mathbb{E}[X] \pm \epsilon$ using $\widetilde{O}(\sigma/\epsilon)$ samples from X.

We apply this algorithm to the sequence of random variables $P_i - P_{i-1}$, as in the classical case.

Quantum speedup of the multilevel Monte Carlo method

Theorem [AM 1504.06987] (informal)

Given the ability to generate samples from a random variable X with variance σ^2, there is a quantum algorithm which approximates $\mathbb{E}[X] \pm \epsilon$ using $\widetilde{O}(\sigma/\epsilon)$ samples from X.

We apply this algorithm to the sequence of random variables $P_i - P_{i-1}$, as in the classical case.

- Cost is still $O(2^i)$ to produce each sample at level i
- We have the variance bound $\mathsf{Var}(P_i - P_{i-1}) = O(2^{-i})$
- Overall cost at the i'th level is $\widetilde{O}(2^{-i/2}/(\epsilon/L) \times 2^i) = \widetilde{O}(2^{i/2}L/\epsilon)$
- Overall cost is $\widetilde{O}(1/\epsilon^{1.5})$.

Quantum speedup of the multilevel Monte Carlo method

Theorem [AM 1504.06987] (informal)

Given the ability to generate samples from a random variable X with variance σ^2, there is a quantum algorithm which approximates $\mathbb{E}[X] \pm \epsilon$ using $\widetilde{O}(\sigma/\epsilon)$ samples from X.

We apply this algorithm to the sequence of random variables $P_i - P_{i-1}$, as in the classical case.

- Cost is still $O(2^i)$ to produce each sample at level i
- We have the variance bound $\mathsf{Var}(P_i - P_{i-1}) = O(2^{-i})$
- Overall cost at the i'th level is $\widetilde{O}(2^{-i/2}/(\epsilon/L) \times 2^i) = \widetilde{O}(2^{i/2}L/\epsilon)$
- Overall cost is $\widetilde{O}(1/\epsilon^{1.5})$.

Can be improved to $\widetilde{O}(1/\epsilon)$ for a Lipschitz continuous payoff.

Conclusions

Some intuition we gained from this work:

- Quantum computers might achieve a speedup over classical algorithms for solving the heat equation, but this speedup is likely to be only polynomial.

Conclusions

Some intuition we gained from this work:

- Quantum computers might achieve a speedup over classical algorithms for solving the heat equation, but this speedup is likely to be only polynomial.
- Quantum algorithms might still offer an advantage in terms of flexibility or space usage over their classical counterparts.

Conclusions

Some intuition we gained from this work:

- Quantum computers might achieve a speedup over classical algorithms for solving the heat equation, but this speedup is likely to be only polynomial.
- Quantum algorithms might still offer an advantage in terms of flexibility or space usage over their classical counterparts.
- The best quantum algorithms for solving PDEs might not be based on solving a system of linear equations.

Conclusions

Some intuition we gained from this work:

- Quantum computers might achieve a speedup over classical algorithms for solving the heat equation, but this speedup is likely to be only polynomial.
- Quantum algorithms might still offer an advantage in terms of flexibility or space usage over their classical counterparts.
- The best quantum algorithms for solving PDEs might not be based on solving a system of linear equations.
- Quantum algorithms can achieve a modest speedup over classical algorithms for solving stochastic differential equations, as used in mathematical finance.

Conclusions

Some intuition we gained from this work:

- Quantum computers might achieve a speedup over classical algorithms for solving the heat equation, but this speedup is likely to be only polynomial.
- Quantum algorithms might still offer an advantage in terms of flexibility or space usage over their classical counterparts.
- The best quantum algorithms for solving PDEs might not be based on solving a system of linear equations.
- Quantum algorithms can achieve a modest speedup over classical algorithms for solving stochastic differential equations, as used in mathematical finance.

Thanks!