7

Solving differential equations

The authors are grateful to Dong An, Di Fang, and Ashley Montanaro for reviewing this chapter.

Overview

Many applications in engineering and science rely on solving differential equations. Accordingly, this constitutes a large fraction of research-and-development high-performance computing (HPC) workloads across a wide variety of industries. There have been many proposals to speed up differential equation solving using a quantum computer. At this point, the consensus is that we lack compelling evidence for practical quantum speedup on industry-relevant problems. However, the field is progressing rapidly, and this conclusion is subject to change.

Some of the main application areas that have been considered are:

- **Computational fluid dynamics** (CFD), usually involving simulation of the Navier–Stokes equation. The main industries relying on CFD simulations are automotive, aerospace, civil engineering, wind energy, weather/climate modeling, and defense. While most simulations focus on air or fluid flow on solid objects, other processes, such as foaming, are also important to model. Large CFD calculations are routinely in the petaflop regime and are run on millions of CPU cores. Specific quantum proposals include [683, 965, 560, 584, 821, 406, 405, 268, 661].
- **Geophysical modeling**, involving simulation of the wave equation. The main industries are oil and gas, hydroelectric, geophysics. Large seismic imaging simulations can easily be in the petaflop regime. Quantum proposals for simulating the wave equation include [779, 518, 365].

- A wide variety of engineering problems involving the **finite element method** (FEM) for studying structural properties of solid objects. The main industries are civil engineering, manufacturing (including automotive), aerospace, and defense. The simulations are typically slightly smaller in scale than CFD, though still often requiring large HPC clusters. Quantum FEM proposals include [580, 777, 1002, 1082].
- **Maxwell's equations and the heat equation** have applications in chip design and other electronic component design, as well as for navigation and radar technology. Specific quantum proposals include [295, 580, 692].
- **Plasma physics** simulations, involving the simulation of the Vlasov equation, are widespread in nuclear fusion research. Quantum approaches include [803, 377, 354].
- **Risk modeling**, involving the simulation of stochastic differential equations, is extensively used in finance (especially derivatives pricing), insurance, and energy markets. Specific quantum proposals include [865, 33, 857, 393, 685].

Differential equations can be categorized according to a number of properties: (i) *ordinary vs. partial* depending on the number of differential variables, (ii) *stochastic vs. deterministic* depending on whether the function is a random variable or not, (iii) *linear vs. nonlinear*. We will focus mainly on linear ordinary and partial differential equations, which have received the most attention in the quantum computing literature, and only comment briefly on stochastic and nonlinear differential equations.

In order to solve a differential equation numerically, one typically specifies a discretization scheme. Two important classes of discretization schemes are: (i) *grid-based* schemes, including finite difference methods (FDMs), as well as the finite volume method (FVM) and the FEM combined with various choices of support grids and preconditioning (see [665, 996] for an introduction). For example, in the finite difference framework, the continuous space is discretized on a uniform grid and the continuous operators are replaced by finite difference operations on neighboring grid points. Alternatively, (ii) one can discretize space by expansion in a functional basis (Fourier, Hermite, etc.), and solve the discrete problem in this basis. This second class of methods is often referred to as *spectral methods* [930]. There is often a tradeoff between error convergence and regularity requirements, with higher-order grid-based methods and spectral methods offering faster error convergence with the number of grid points or basis functions utilized, but requiring more stringent assumptions on the smoothness of the solution of the differential equation, which are not always satisfied in the applications listed above.

Given a discretization scheme, solving linear differential equations can be accomplished by solving linear systems of equations. In cases where one is interested in very high precision, requiring very fine discretization, the linear system of equations can be too large for straightforward numerical solutions on a classical computer. In particular, if one wants high-precision results integrated over time, and/or systems with many continuous variables, then the simulations can be challenging both in time and memory. Quantum algorithms aim to offer a speedup over classical methods by leveraging the existence of quantum linear system solvers, or more generally, primitives in quantum linear algebra, which enable quantum algorithms to perform efficient manipulations of vectors that are exponentially large in the number of qubits and elementary operations involved. However, at a technical level, various complications arise, including the difficulty of reading out useful information at the end of the algorithm, and assumptions about the differential equation that must be true for the methods to work. Ultimately, polynomial speedups for end-to-end problems appear to be possible, but for differential equations in a fixed number of spatial dimensions, exponential speedups for real-world applications are not generally attainable.

Actual end-to-end problem(s) solved

We are interested in solving a general linear partial differential equation (PDE) of the form

$$\mathcal{L}(u(x)) = f(x) \quad \text{for} \quad x \in \mathbb{C}^d\,, \tag{7.1}$$

where $\mathcal{L}$ is a linear differential operator acting on the function $u(x)$, and $f(x) \in \mathbb{C}$ is the inhomogeneous term (which is 0 for homogeneous PDEs). In addition to Eq. (7.1), we are given boundary conditions on $u(x)$ and its derivatives, which ideally are sufficient to ensure a unique solution—for example, Dirichlet boundary conditions refer to a specification of a function $b(x)$ and a requirement that $u(x) = b(x)$ for x contained in some subset $\Omega \in \mathbb{C}^d$. As a canonical example of a linear PDE, consider the Poisson equation in d dimensions, given by

$$\frac{\partial^2 u}{\partial x_1^2} + \cdots + \frac{\partial^2 u}{\partial x_d^2} = f(x)\,. \tag{7.2}$$

As another example, we consider a linear ordinary differential equation (ODE) of the form

$$\frac{\mathrm{d}\bar{u}(t)}{\mathrm{d}t} = A(t)\bar{u}(t) + \bar{b}(t)\,, \tag{7.3}$$

where we refer to the variable $t \in \mathbb{R}$ as time (although it could represent a different quantity), $\bar{u}(t)$ and $\bar{b}(t)$ are N-dimensional vectors, and $A(t)$ is an $N \times N$ matrix. Boundary conditions on $\bar{u}(t)$ are also specified, often in the form of an initial condition at $t = 0$, with one seeking the solution at some final time T; this is known as an *initial value problem*. Higher-order linear ODEs can always be transformed into first-order linear ODEs. Note that if $A(t)$ is anti-Hermitian and $\bar{b}(t) = 0$, then Eq. (7.3) becomes the time-dependent Schrödinger equation, which is solved directly with Hamiltonian simulation. Equation (7.3) could be viewed within the framework of Eq. (7.1) with $d = 1$ and u a vector-valued rather than scalar-valued function. We separate these cases because the existing literature typically uses either Eq. (7.1) or Eq. (7.3) as its starting point, and the methods pursued in each case are distinct.

For nonlinear PDEs, the linear equations in Eq. (7.1) and Eq. (7.3) are replaced by nonlinear ones. For example, one can extend Eq. (7.3) to consider an ODE with a polynomial nonlinearity of the form

$$\frac{\mathrm{d}\bar{u}(t)}{\mathrm{d}t} = F_M(t)\bar{u}(t)^{\otimes M} + A(t)\bar{u}(t) + \bar{b}(t)\,, \tag{7.4}$$

where $F_M(t)$ is a tensor encoding the nonlinearity, although note that existing quantum algorithms have focused on the case where $F_M(t)$ and $A(t)$ are time independent. This class of differential equations includes important potential applications such as CFD, since the Navier–Stokes equation is nonlinear with a quadratic nonlinearity ($M = 2$).

What does it mean to "solve" the differential equation? In the most general sense, this refers to obtaining an expression for the solution $u(x)$ (for Eq. (7.1)) or $\bar{u}(t)$ (for Eq. (7.3)). While closed-form solutions can be derived for some simple differential equations, this is not possible in general, and the solution typically must be computed numerically. However, in specific applications, we often do not need complete information about the solution function $u(x)$ or $\bar{u}(t)$ to accomplish a certain goal. An end-to-end problem involving the solution of a differential equation boils down to estimating the value of some *property* of the solution, denoted by $\mathcal{P}[u] \in \mathbb{R}$, up to specified additive error parameter ϵ. For linear PDEs, a straightforward example is when the property $\mathcal{P}$ is simply the value of u at a specific point x^*, that is, $\mathcal{P}[u] = u(x^*)$. More generally, we restrict to the case where $\mathcal{P}[u]$ is a linear functional of u, that is, $\mathcal{P}[u] = \langle r, u\rangle := \int_{x\in\Omega} \mathrm{d}x\, r(x)u(x)$ for some subset $\Omega \subset \mathbb{R}^d$ and function $r : \Omega \to \mathbb{R}$ for which $\langle r, r\rangle = 1$ [777]. For example, in [295], a quantum algorithm for solving Maxwell's equations based on the FEM was given, where the quantity of interest was not the electric field itself at any specific point, but rather the

electromagnetic scattering cross section. In this case, the cross section was given by the square of a linear functional of u.

Properties of ODEs can be treated in the same framework, where we are interested in computing quantities of the form $\mathcal{P}[u] = \int_{t\in\Omega} \mathrm{d}t\ \bar{r}^\intercal(t)\bar{u}(t)$, which are linear functionals of the entries of $\bar{u}(t)$ over some subset Ω of the interval $[0, T]$. However, we note that for initial value problems, often of primary interest are properties at the final time T, in which case $\mathcal{P}[u]$ would reduce to an inner product $\bar{r}^\intercal\bar{u}(T)$. For example, in [832], the drag force on a ship hull was expressed as a linear functional of the solution to the lattice Boltzmann equation evolved forward in time.

Dominant resource cost/complexity

There are many distinct approaches to constructing a quantum algorithm for solving the end-to-end problem above. The exact complexity will of course depend on the method, but it will also depend on specific details related to how the differential equation and its boundary conditions are specified to the quantum algorithm (input model), as well as instance-specific factors such as how smooth the solution to the differential equation is. Here we overview some of the available choices and complexity considerations, focusing the bulk of the discussion on methods that leverage the quantum linear system solver (QLSS) as a primitive, as these have received the most attention in the literature.

Discretization of linear PDEs: Any numerical method must perform some form of discretization. First, we focus on linear PDEs such as Eq. (7.2) where there is no time variable. The choice of discretization will depend sensitively on the problem at hand. In the case of the Poisson equation in Eq. (7.2) with Dirichlet boundary conditions, quantum algorithms leveraging discretization schemes based on FDM, FEM, and spectral methods were discussed in [228], [777], and [285], respectively. The key goal is to minimize the number N of grid points or basis functions while achieving discretization error $O(\epsilon)$. Using low-order grid-based methods, a problem in d spatial dimensions requires taking $N = (1/\epsilon)^{\Omega(d)}$ grid points, with some caveats on solution norm and regularity [777]. Alternative sparse-grid or spectral methods can improve the $1/\epsilon$ dependence to logarithmic, but still scale exponentially with d [285]; however, these generally require stricter regularity requirements on the solution to the differential equation, which may not be satisfied in applications.

After appropriate discretization, the linear differential equation in Eq. (7.1) (along with its boundary conditions) reduces to a matrix equation:

$$L|u\rangle = |f\rangle\,. \tag{7.5}$$

This is the same linear equation that would be obtained for a classical method using the same discretization scheme. Information about the solution function $u(x)$ is encoded into the N components of the vector $|u\rangle$.[1] Classical methods typically manipulate a full description of all components of the vector $|u\rangle$, whereas quantum methods can create the normalized quantum state $|u\rangle/\||u\rangle\|$ encoding these N components into its amplitudes with $O(\log(N))$ qubits.

If the linear PDE has a time variable, one option is to treat it equally as the other $d-1$ variables (see, e.g., [692]), but it is often treated separately. First, discretization of the other $d-1$ variables using N total grid points or basis functions is performed as above, which reduces the linear PDE in Eq. (7.1) to an ODE with N variables as in Eq. (7.3). Time is then discretized and propagated as discussed for ODEs below.

Discretization of time in linear ODEs: To solve the linear ODE on N variables in Eq. (7.3)—whether it came about via discretization of a PDE or otherwise—the time interval $[0, T]$ is discretized into grid points, and the solution at one grid point is related to the solution at the prior grid point by a time-ordered exponential of the matrix A. If A is time independent, this exponential can be approximated by a truncated Taylor series [138, 646, 571], and if A is time dependent, it can be approximated by a truncated Dyson series [132]. The number of grid points needed scales linearly with T, and the series is truncated at order $\text{polylog}(T/\epsilon)$. An alternative approach when $\bar{u}(t)$ is sufficiently smooth in time uses spectral methods, which approximate $\bar{u}(t)$ as a truncated series over a complete set of basis functions [279]. In any case, the relation between the solution at different time grid points or basis functions leads to a linear system of equations, now of size roughly $N' = O(NT)$, but again of the form $L|u\rangle = |f\rangle$ as in Eq. (7.5).

Here the solution $|u\rangle$ is a "history state" meaning that it is given by a *superposition* of states $|t\rangle|\bar{u}(t)\rangle$ for different discrete values of t across the entire interval $[0, T]$. Since one is often interested only in $\bar{u}(T)$, additional trivial time steps can be included at the end to boost the portion of the history state amplitude on the $t = T$ branch [131].

It is important to emphasize that classical methods for solving ODEs do not solve the same linear equation $L|u\rangle = |f\rangle$ arrived at by these methods. Rather, they typically handle time in a time-marching fashion where the value of $\bar{u}(t)$ at one time step is directly computed from its value at one or more previous time steps. In [379], a quantum time-marching strategy was proposed

[1] In this chapter, we adopt a convention where quantum states like $|u\rangle$ need not be normalized states. In fact, the norm, denoted by $\||u\rangle\|$ will be important for reasoning about the complexity.

for time-dependent homogeneous linear ODEs, which generates a sequence of quantum states $|\bar{u}(0)\rangle, |\bar{u}(t_1)\rangle, |\bar{u}(t_2)\rangle, \ldots, |\bar{u}(T)\rangle$ (rather than a superposition of these states). This method avoids the need to solve a linear system, but it does utilize primitives from quantum linear algebra. Similarly, the methods of [312, 579, 80, 581, 36, 37] avoid the need to solve linear systems by mapping ODEs to equations that can be simulated with Hamiltonian simulation [312, 579, 80, 581], or linear combination of Hamiltonian simulation [36, 37].

Solving the linear system: Once the linear PDE or ODE has been reduced to a linear system $L|u\rangle = |f\rangle$, it can be solved on a quantum computer by applying the QLSS. The QLSS subroutine prepares a quantum state approximating the normalized solution vector $|u\rangle/\||u\rangle\|$ up to some specified precision ξ in ℓ_2 norm, where $\||u\rangle\| = \sqrt{\langle u|u\rangle}$ is the norm of the quantum state encoding the solution to the linear system. To do so, the QLSS assumes access to oracles that (coherently) query the matrix elements of L and prepare the normalized state $|f\rangle/\||f\rangle\|$. Optimal QLSSs [313, 327] (see also alternative near-optimal methods in [26, 282, 248, 964, 31, 571]) make $O(s\kappa \log(1/\xi))$ queries to these oracles, where κ is the condition number of the matrix L (i.e., the ratio of the largest and smallest singular values), and s is the maximum number of nonzero elements in any row or column of L ("sparsity"). Additionally, one can compute an estimate for the norm $\||u\rangle\|$ up to relative error ξ using $\widetilde{O}(s\kappa/\xi)$ queries (note the worse ξ-dependence) [327, 248]. For simplicity, we assume that to achieve ϵ overall error on the end-to-end problem, it will suffice to take $\xi = O(\epsilon)$, although there can also be factors that depend on the choice of discretization and norms of the solution u (see, e.g., [777]).

Henceforth, let N' refer to the size of the linear system being solved, so $N' = N$ for the PDE example described above, and $N' = O(NT)$ for the ODE example (with N reserved for the size of the matrix A).

The oracles for querying the matrix elements of the s-sparse $N' \times N'$ matrix L and for preparing the N'-dimensional state $|f\rangle/\||f\rangle\|$ are assumed to have cost polylog(N'). This is valid if the matrix elements of L can be efficiently computed "on the fly," which is plausible when they are given by succinct expressions, for example, based on a simple finite difference formula. However, if entries of L and $|f\rangle/\||f\rangle\|$ depend on arbitrary, classically stored data related to, for instance, object geometries, boundary conditions, or grid point locations, then the assumption of polylog(N') cost per query requires access to a log-depth quantum random access memory (QRAM). This requirement is necessary in many practical applications of PDEs involving highly complex geometries in three spatial dimensions, such as CFD and seismic modeling. The assumption of log-depth QRAM brings significant caveats—for example,

while the quantum circuits for implementing the QRAM operation can be parallelized to have depth polylog(N), they cannot avoid having size poly(N) for accessing a database with poly(N) entries; see Chapter 17 on loading classical data for more information.

With these assumptions, the QLSS portion of the quantum algorithm can be performed exponentially faster in the parameter N, and with exponential saving in memory, compared to any classical method that manipulates vectors of size N, which includes Gaussian elimination and iterative methods like conjugate gradient. Specifically, the quantum complexity to obtain the state $|u\rangle/\||u\rangle\|$ is given by

$$s\kappa \cdot \text{polylog}(N', 1/\epsilon)$$

and the cost to obtain $\||u\rangle\|$ is $s\kappa\epsilon^{-1} \cdot \text{polylog}(N', 1/\epsilon)$. The number of qubits is $O(\log(N'))$, although if a log-depth QRAM is necessary, this may require poly(N') ancilla qubits.

Reading out estimates for properties of the solution to the differential equation: Preparing the $\log(N')$-qubit state $|u\rangle/\||u\rangle\|$ does not immediately yield an estimate for the property $\mathcal{P}[u]$. Indeed, reading out useful information from $|u\rangle/\||u\rangle\|$ represents a major bottleneck of the algorithm. Consider the case that $\mathcal{P}[u]$ corresponds to the value $u(x^*)$ at a specific point x^* (for PDEs), or the amplitude $\langle x^*|\bar{u}(T)\rangle$ on one of the basis states (for ODEs). Then, the estimation of $\mathcal{P}[u]$ to precision ϵ is performed with amplitude estimation, which introduces multiplicative overhead $O(\||u\rangle\|/\epsilon)$ into the complexity. To read out all N amplitudes of the state $|u\rangle$ in this fashion, a linear factor of N would be reintroduced, although more advanced methods of pure state tomography can reduce this to $\sqrt{N}$ [49]. In the more general case that $\mathcal{P}[u]$ is a linear functional, the value of $\mathcal{P}[u]$ can be expressed (up to discretization error) as an overlap $\mathcal{P}[u] = \langle r|u\rangle$ between some preparable normalized state $|r\rangle$ and the solution vector $|u\rangle$ that solves $L|u\rangle = |f\rangle$. Thus, to compute $\mathcal{P}[u]$, one computes the overlap between $|r\rangle$ and $|u\rangle/\||u\rangle\|$, and then multiplies by $\||u\rangle\|$. Overlap estimation [637] is a straightforward application of amplitude estimation, and achieving precision $\epsilon/\||u\rangle\|$ introduces $O(\||u\rangle\|/\epsilon)$ multiplicative overhead. Thus, the overall scaling of the complexity is

$$\frac{s\kappa\,\||u\rangle\|}{\epsilon} \cdot \text{polylog}(N', 1/\epsilon)\,. \tag{7.6}$$

For initial value problems governed by the ODE in Eq. (7.3), one is often interested in properties $\mathcal{P}[u]$ that depend only on the solution $\bar{u}(T)$ at the final time T. When $|u\rangle = \sum_i |t_i\rangle|\bar{u}(t_i)\rangle$ is a history state encoding of the solution $\bar{u}(t)$

over the whole interval, the complexity of computing useful information will grow with the ratio

$$q = \frac{\sup_{t\in[0,1]} \| \, |\bar{u}(t)\rangle \, \|}{\| \, |\bar{u}(T)\rangle \, \|} \approx O\left(\frac{\| \, |u\rangle \, \|}{\| \, |\bar{u}(T)\rangle \, \|}\right),$$

that is, the factor by which the norm of the solution has decayed compared to its maximum on the interval $[0, T]$ (the approximation is correct assuming $\sup_{t\in[0,T]} \| \, |\bar{u}(t)\rangle \, \|$ accounts for a constant fraction of the total norm $\| \, |u\rangle \, \|$). This arises from the fact that $|T\rangle|\bar{u}(T)\rangle$ contributes at most $\| \, |\bar{u}(T)\rangle \, \| \approx O(\| \, |u\rangle \, \|/q)$ amplitude to the history state $|u\rangle$ and thus the additive precision ϵ will need to be on the order of $\| \, |u\rangle \, \|/q$, or smaller, to learn something useful about $\bar{u}(T)$. Since the complexity scales linearly with $\| \, |u\rangle \, \|/\epsilon$, the complexity grows with q. Unfortunately, q can be large and growing with T if the solution to the ODE is decaying. Furthermore, the dependence of the complexity on q is necessary since otherwise the algorithm would be able to perform postselection on low-probability events, a power ruled out by widely believed complexity-theoretic conjectures; see [138, 35].

The persisting polylog(N) dependence in Eq. (7.6) suggests an exponential speedup in the parameter N compared to classical methods, but this conclusion depends on the scaling of the parameters s, κ, and $\| \, |u\rangle \, \|/\epsilon$ with N.

Condition number: The sparsity s and condition number κ depend on the differential equation and the choice of discretization, but can often be controlled. For example, for PDEs discretized by the FEM, in many instances we have $s = O(1)$ and $\kappa \leq O(N^{2/d})$ (see, e.g., [198, Theorem 9.7.1]). Additionally, preconditioning of linear systems to reduce κ is an effective technique in classical iterative approaches to solving linear systems such as the conjugate gradient method, and several studies have examined the possibility of integrating these into the QLSS in some circumstances [295, 925, 990, 83]. In the best case scenario, these could reduce κ to $O(1)$.

In the setting of ODEs like Eq. (7.3), upper bounds on κ can be derived. These bounds can have the form $\kappa \leq CT$, where C is a factor that depends on the spectral properties of A.[2] The upper bounds on κ also require an assumption that the ODE is "dissipative" [131, 138, 646, 132]; otherwise, the value of C in the bound can grow exponentially with T [646], consistent with the observation that the norm of the history state $|u\rangle$ can be exponentially larger than the norm of the initial state $|\bar{u}(0)\rangle$. A sufficient condition for the ODE to be dissipative is that A is diagonalizable and the real parts of its eigenvalues

[2] Generally, the bound on κ scales with the number of grid points in time. The linear-in-T bound is achieved when using the Dyson series method with high-order truncation [132].

are non-positive [138] so that the solutions are stable and do not grow with time, although this technical definition was relaxed slightly in [646] and now includes nondiagonalizable A. The requirement of dissipation is not as relevant for classical solvers based on time-marching strategies, which can renormalize growing solutions at each step and do not generally require solving linear systems.

The linear dependence on T in the complexity cannot be improved in general since this factor appears in the complexity of optimal Hamiltonian simulation, which corresponds to the special case that A is anti-Hermitian and $\bar{b} = 0$ [134]. However, it has been shown that many ODEs admit "fast forwarding" and the dependence on T can be reduced. For example, for stable ODEs (when all eigenvalues of A have a negative real part), a bound of the form $\kappa \leq \widetilde{O}(\sqrt{T})$ was derived in [571]; see also [35].

Final complexity: For dissipative ODEs on N variables, propagated to time T, the final complexity inherits a linear dependence on CT via the condition number, and existing literature typically includes the factor q defined above directly in the complexity statements. These statements are phrased to say that the state $|\bar{u}(T)\rangle$ can be obtained to error ξ in complexity roughly $TsqC \cdot \text{polylog}(N, 1/\xi)$ [138, 646, 571, 132], which accounts for postselecting the time register to $t = T$ but not yet the complexity to read out a property of interest.[3] Defining $\epsilon' = \epsilon q / \| |u\rangle \|$ to be the normalized precision parameter that should be small in order to learn something interesting, and taking $\xi = O(\epsilon')$, the total end-to-end complexity including readout could then be expressed as

$$\frac{TsqC}{\epsilon'} \cdot \text{polylog}(N, q/\| |u\rangle \| \epsilon') \,. \tag{7.7}$$

For PDEs in d dimensions, we recall that N and ϵ are not independent parameters: in general, we are interested in simulating the PDE to a fixed precision, and adapt N to reach the desired precision. As discussed, depending on the discretization method, N scales either as $(1/\epsilon)^{O(d)}$ or $(\text{polylog}(1/\epsilon))^d$, but either way, we have that $\text{polylog}(N) \leq d^{O(1)} \cdot \text{polylog}(1/\epsilon)$. For PDEs with a time variable and initial conditions specified at $t = 0$, where $d - 1$ dimensions are discretized and time is integrated via mapping to an ODE, we substitute

[3] We briefly mention the complexity of alternative approaches to ODEs that avoid solving linear systems. The quantum time-marching method of [379] has a different complexity form, but has similar features, growing with time (in this case, as T^2) and depending on an "amplification ratio" $Q > q$, but offering other potential benefits, such as minimal regularity requirements (even allowing $A(t)$ to have jump discontinuities) and needing fewer queries to the initial condition $\bar{u}(0)$. Meanwhile, the linear combination of Hamiltonian simulation method [37] shares the feature of needing minimal queries to $\bar{u}(0)$ while matching the complexity stated above.

this value of N into Eq. (7.7). This gives a total complexity of

$$d^{O(1)} \cdot \widetilde{O}(TCq/\epsilon')$$

for reading out a property of the (renormalized) solution to the PDE at time T to precision ϵ'.

For static PDEs where all d dimensions are discretized by a grid-based method like the FEM, we instead substitute $\kappa = O(N^{2/d})$, $s = O(1)$, and $N = (1/\epsilon)^{O(d)}$ into Eq. (7.6), yielding overall end-to-end complexity

$$\frac{\| |u\rangle \| \, d^{O(1)}}{\epsilon^{1+O(1)}}$$

to compute a global property of the PDE, given conditions on the boundary. If preconditioning improves κ to $O(1)$, the ϵ dependence is improved to essentially linear in $1/\epsilon$—see [777] for a more careful analysis specific to the FEM that arrives at similar expressions, but better accounts for impact of solution norm and smoothness.

Generally, the main conclusion is that—irrespective of the discretization scheme—the quantum complexity is polynomial in the desired precision $1/\epsilon$, although for d-dimensional PDEs the complexity may scale as $\text{poly}(d) \cdot \text{poly}(1/\epsilon)$ rather than $\text{poly}((1/\epsilon)^d)$. Thus, for fixed dimension d there is potentially room for a polynomial-in-$1/\epsilon$ quantum speedup, the size of which grows exponentially in the dimension d. Ultimately, the necessity of the $O(1/\epsilon)$ scaling is traced back to the fact that the quantum solver produces a quantum state encoding the normalized solution to the differential equation, potentially exponentially faster than leading classical methods such as conjugate gradient, but the exponential speedup is lost in the readout step, where one must learn an observable of interest to error ϵ. Moreover, this conclusion holds not just for "bad" observables (like full state tomography), but for any observable, due to the $\Omega(1/\epsilon)$ cost of quantum readout.

Nonlinear differential equations: The immediate issue with applying the above methods to nonlinear differential equations such as the nonlinear ODE of Eq. (7.4) is that discretization no longer leads to a system of linear equations. Early work on quantum algorithms for nonlinear ODEs handled this issue by dividing time into short time steps and preparing the quantum state encoding the solution at one time step using multiple copies of the solution at the previous time step [679]. Since quantum states cannot be cloned, the complexity of this strategy necessarily grows exponentially in the number of time steps. More recent quantum algorithms for nonlinear differential equations instead proceed by first linearizing the differential equation and then applying

the methods sketched above [710, 702, 1061, 703, 646, 315]. Specifically, the most heavily studied approach has been Carleman linearization, where one exactly maps a nonlinear ODE with polynomial nonlinearity such as Eq. (7.4) to a linear ODE on an infinite-dimensional variable $(\bar{u}, \bar{u}^{\otimes 2}, \bar{u}^{\otimes 3}, \ldots)$, and then truncates to form an approximate finite-dimensional linear ODE. Quantum algorithms based on this method were first studied in [702] and further developed in [703, 646, 315, 207], and it has also been analyzed in the context of specific differential equations such as reaction-diffusion equations [703, 315, 32], the Navier–Stokes equation (via the lattice Boltzmann equation) [683, 832], and differential equations related to training classical neural networks [701]. The complexity of the quantum algorithm has a similar scaling as that for linear ODEs quoted in Eq. (7.7), growing linearly in T, q, and $1/\epsilon'$. However, this complexity scaling requires an additional assumption: a quantity R capturing the nonlinearity-to-dissipation ratio of the differential equation must satisfy $R < 1$ for the errors to be controlled (see [702, 703, 315, 1055]), and it is not always clear when this condition holds.

For example, in the case of the Navier–Stokes equation, the size of the nonlinearity—and hence the value of R—grows with the "Reynolds number" of the fluid, and the condition $R < 1$ would be violated when simulating high-Reynolds-number turbulent flows. Turbulent flows are potentially handled by applying the Carleman linearization method to the lattice Boltzmann equation rather than the Navier–Stokes equation directly [683, 832], in which case the size of the nonlinearity does not scale with the Reynolds number. Indeed, generally speaking, for this approach to nonlinear differential equations, there is a delicate interplay between the size of the input state at time $t = 0$, the form of the nonlinearity, and the amount of dissipation in the linear term; see [315] for a discussion.

Separate from these approaches, methods have been proposed that map nonlinear classical dynamics to linear phase-space dynamics that can be simulated with Hamiltonian simulation [590, 354, 580, 578, 581].

Comments on the complexity of alternative methods and problems: We briefly comment on two further classes of applications involving PDEs, but which typically have very different characteristics. The first is stochastic differential equations (SDEs), which are simulated extensively in computational finance and more generally in risk modeling. There, one typically samples trajectories of the SDE (via Monte Carlo methods), and evaluates observables stochastically. Quantum-accelerated Monte Carlo has been worked on extensively (see Section 8.2 on pricing financial options). Where classical methods require sampling $O(1/\epsilon^2)$ trajectories to obtain an ϵ-estimate of a certain quan-

tity, the quantum method can create a superposition of trajectories and read out the relevant amplitude at $O(1/\epsilon)$ cost—a quadratic quantum advantage. On the classical side, a key advantage of these methods is that they avoid an exponential scaling in the number of continuous dimensions d (i.e., the "curse of dimensionality"), unlike the "Eulerian" approaches discussed above that discretize the d-dimensional space into $N \geq \exp(\Omega(d))$ grid points or basis functions. Thus, they are relatively effective when d is large, and less preferred when d is small due the fact that their ϵ dependence cannot be better than $O(1/\epsilon^2)$. In fact, in some cases, classical and quantum trajectory-based methods can be applied not only to SDEs but also to (deterministic) PDEs, and thus they compete against QLSS-based PDE and ODE methods we have discussed. For example, the quantum and classical complexity of a Monte Carlo approach for the heat equation was studied in [692] and compared against alternative approaches—for the end-to-end problem considered, the classical Monte Carlo approach outperformed all classical alternatives when $d > 4$, and the quantum-accelerated Monte Carlo approach outperformed all (quantum or classical) alternatives when $d > 2$. For SDEs, an alternative to Monte Carlo is to map the SDE to a Fokker–Planck equation via the Itô calculus and solve the Fokker–Planck PDE. This has been proposed in [441]. However, for most SDEs of interest in risk analysis, Monte Carlo simulation converges in a number of samples scaling linearly in the number of variables, leaving very little room for a quantum speedup in these applications given our current understanding.

The last class of problems to be mentioned are multi-particle Schrödinger equations. They are (i) high dimensional, (ii) complex, and (iii) require high-precision solutions for practical applications. Hence, they match all of the criteria under which a quantum advantage might be expected. The second-quantized approach to solving the full configuration interaction molecular Schrödinger equation is a specific case of the spectral method, although here one must solve an eigenvalue equation rather than a linear system. Unsurprisingly, this case has already gathered a lot of attention (see Chapter 2 on quantum chemistry).

Existing resource estimates

An explicit resource estimate for linear PDEs was reported in [902] for solving Maxwell's equations to estimate an electromagnetic scattering cross section in 2D. Following the asymptotic analysis of [295], it employed an FEM-based discretization scheme to form a linear system of size $N = 3 \times 10^8$, targeting accuracy $\epsilon = 0.01$. The estimates did not incorporate preconditioning methods and assumed a value for the condition number $\kappa \approx 10^4$, ultimately finding

that 10^{29} T gates would be needed to complete the computation. However, this work predated asymptotic and practical advancements to the complexity of the QLSS [313, 327], and modern estimates for the same problem would likely lead to more reasonable resource counts. Note also that much of the art in classical PDE solvers is to find appropriate preconditioning schemes to control the condition number. In [295], it was shown that one common class of preconditioners works within the framework of the quantum algorithm, but it is as yet unclear if this is the case more generally.

For ODEs, [571] gave a detailed performance analysis of the Taylor series truncation approach developed in [138, 646] applied to general time-independent dissipative ODEs. They gave explicit upper bounds on the condition number κ of the linear system in terms of the total evolution time T and the "log-norm" of the matrix A. They considered the task of outputting the history state $|u\rangle$ or the final-time state $|\bar{u}(T)\rangle$. By combining the bound on κ with explicit upper bounds from [572] on the query complexity of the QLSS, they determined an upper bound on the number of times the algorithm needs to query a block-encoding of the ODE matrix A to accomplish this task. The estimated number of queries per unit time varied from 10 to 10^5 over the parameter regime considered, and these figures would be reduced with subsequent improvements to the QLSS complexity, such as those reported in [327].

In [832], the query bounds of [571] were applied to the specific end-to-end CFD problem of computing the drag force on a ship hull in the incompressible (and potentially turbulent) parameter regime, by solving the nonlinear lattice Boltzmann equation (linearized via Carleman linearization). They considered a simplified version of the problem where the ship hull is modeled as a sphere. They estimated that the quantum algorithm would need 10^{20}–10^{24} T gates and roughly 10^3–10^5 logical qubits (depending on the value of the Reynolds number) to compute the drag force on the sphere. Classically, the lattice Boltzmann method is a high-accuracy method and would be computationally intractable for this instance in the high-Reynolds-number regime. In practice, classical methods resort to lower accuracy methods, which for this instance can be completed within several minutes on a laptop. Computing the drag force on actual ship hulls with the quantum method is expected to be significantly more resource intensive compared to the flow-past-a-sphere instance due to the need to coherently load the boundary conditions describing the ship hull each time the block-encoding is queried.

Caveats

A key caveat is that many analyses in the literature do not consider the full end-to-end problem that needs to be solved for applications. Often, these works only consider the cost of preparing the quantum state $|u\rangle$ that encodes the solution to the differential equation into its amplitudes, and they report this cost in terms of the number of queries to oracles of the input data. Thus, these works study the task of solving differential equations as a primitive—similar to Hamiltonian simulation, that is, simulation of the time-dependent Schrödinger ODE—rather than as an end-to-end application. As discussed above, reading out useful information to precision ϵ introduces a $\Omega(1/\epsilon)$ multiplicative overhead and dramatically changes the complexity scaling, precluding exponential end-to-end speedups. Furthermore, whereas a full classical description of the solution could be computed just once, and subsequently many properties read out from that description, the state $|u\rangle$ is consumed during readout, and the number of times $|u\rangle$ needs to be prepared grows with the number of properties one wants to learn. In some cases, one may only seek to learn a few observables, but in other cases, extracting the desired information might require near full tomography of the quantum state $|u\rangle$, which in certain situations removes all quantum advantage [692].

The readout caveat can potentially be avoided if a small number of *samples* from the state $|u\rangle$ measured in the computational basis, as opposed to properties $\mathcal{P}[u]$ as defined above, would be useful for the end-to-end application. However, in such cases one must also be careful to compare against classical methods for the same task, where quantum-inspired methods can be competitive [977]. In [80], it was shown that BQP-hard problems can be encoded into an ODE describing coupled oscillators and solved by sampling from $|u\rangle$, but this situation would be unlikely to arise naturally in applications. In [701], it was suggested that samples from $|u\rangle$ encoding the solution to certain nonlinear differential equations could be useful for training neural networks.

Another caveat is that complexity statements often report only the number of times the algorithm queries oracles for the input data. In applications where the input data encoding complex boundary conditions or object geometries is not efficiently computable but rather stored in a large classical database, one must assume access to a log-depth QRAM in order to implement these oracles efficiently, an assumption that has its own caveats.

For simulating time evolution of ODEs, it is important to emphasize the dependence of the complexity in Eq. (7.7) on the parameter q, which for dissipative systems grows with time, potentially exponentially, as the size of the solution decays. Furthermore, for nonlinear differential equations, we reiterate

that existing quantum algorithms are often based on the assumption that the nonlinearity-to-dissipation ratio is sufficiently small, which may not be satisfied in practical instances. Generally speaking, strong nonlinearities can cause the solution to develop discontinuities, and linearization schemes might break down for problems of interest if the solutions lack sufficient regularity, as can be the case for simulations of turbulence in CFD.

Finally, we note that due to the large number of methods available to classical solvers of differential equations, an important caveat is that any claim of quantum advantage must be sure to compare against the best possible classical method, and consider the possibility that this classical method might benefit from parallelization.

Comparable classical complexity and challenging instance sizes

Classical algorithms for linear PDEs can compute a classical description of the solution u by solving the same linear equation $L|u\rangle = |f\rangle$ solved by the quantum algorithm. For an s-sparse $N \times N$ linear system, the complexity of an exact Gaussian elimination approach is $O(N^\omega)$, where $\omega < 2.37$ is the matrix multiplication exponent. However, in practice, approximate iterative methods such as the conjugate gradient method are more common. The complexity of conjugate gradient scales as $\widetilde{O}(Ns\sqrt{\kappa}\log(1/\epsilon))$ when L is positive semidefinite [481, Chapter 10.2]. For a discretization scheme like the FEM where $N = (1/\epsilon)^{\Omega(d)}$, and using the aforementioned bound $\kappa \leq O(N^{2/d})$, the complexity of the conjugate gradient approach is $s(1/\epsilon)^{\Omega(d)}$, which has exponential dependence on the spatial dimension d but for fixed d scales as $\text{poly}(1/\epsilon)$. Additionally, in practice, the conjugate gradient method benefits from preconditioning techniques to reduce κ, and from parallel implementation on graphics processing units (GPUs). For a sense of scale, [713] used the preconditioned conjugate gradient method within a finite element analysis to compute the thermal conductivity and elasticity of certain 3D cast iron samples imaged with micro-computed tomography (a task chosen mainly to benchmark their method). Among other reported results, their implementation solved the end-to-end problem, which required solving several linear systems, with $N \geq 10^6$ in less than 1 second, and $N \geq 4 \times 10^8$ in less than 30 minutes using a single GPU with 8 gigabytes of RAM.

For linear and nonlinear initial value problems, classical methods could apply conjugate gradient or other linear system solvers to the same linear equation $L|u\rangle = |f\rangle$ that the quantum algorithm constructs to solve the ODE. The complexity of this approach would have a linear-in-N dependence, but since the solution would be a full classical description of the history state, it would

not need to also pay the factor of q arising from postselecting on the $t = T$ branch of the history state, and it would avoid the $O(1/\epsilon)$ readout cost.

However, most classical methods do not follow this route and instead integrate the ODE with a time-marching method, where a description of the length-N vector is propagated forward in time, for which there are many options [1024, 1025]. Similar to the quantum complexity, nearly linear-in-T scaling is achieved by high-order methods, so long as $A(t)$ is smooth up to corresponding order, or by spectral methods [930]. In the time-independent or smooth case, one would achieve $NTC' \cdot \text{polylog}(T, 1/\epsilon)$ complexity, where C' is some constant depending on the spectral properties of A, similar to C. As in the quantum case, care must be taken to choose ϵ appropriately when solutions are exponentially growing or decaying.

We also mention that there has recently been work on using machine learning methods to classically simulate nonlinear PDEs, especially for CFD [1019, 730]. These methods are generally very fast, but they are heuristic, so they are suitable in some instances but not when high-confidence, high-accuracy solutions are required.

Speedup

For linear PDEs in d dimensions, solved via discretization with $N = (1/\epsilon)^{\Omega(d)}$ grid points and inverting the corresponding linear system, the speedup is a reduction from time roughly $\epsilon^{-\Omega(d)}$ classically to $d^{O(1)}\epsilon^{-1}\epsilon^{-O(1)}$ quantumly, here omitting dependencies on $\| |u\rangle \|$ and $\log(\epsilon^{-1})$. The $O(1)$ powers depend on the details of the discretization and the efficacy of preconditioning. Other discretization schemes may give rise to slightly distinct complexity forms, but in any case, the conclusion is that for fixed dimension d, the speedup is at best *polynomial*, a point that has been made in more detail in, for example, [777, 692].

The speedup can be exponential in the parameter d. However, in many engineering applications, the number of dimensions is fixed to be fewer than four (three for space, one for time), limiting the advantage quantum methods can obtain. Furthermore, for PDEs with large d, trajectory-based classical strategies can avoid the exponential-in-d complexity scaling, and in cases where these methods apply, the best possible speedup is typically a quadratic reduction from $O(1/\epsilon^2)$ to $O(1/\epsilon)$; see, for example, [692] and Section 8.2 on option pricing.

For integrating ODEs of N variables forward in time, there can be an exponential speedup in the parameter N. However, since N and ϵ are typically related by $N \leq (1/\epsilon)^{\Omega(d)}$ (e.g., when the ODE arises via discretization of a PDE), the $O(1/\epsilon)$ cost of readout will contribute a much larger factor than the

polylog(N) cost of the QLSS, and the best possible speedup is again polynomial. An exponential speedup could be possible if $N \geq \exp(\Omega(1/\epsilon))$, or if samples from the state $|u\rangle$ were directly useful within the end-to-end application, but this assessment would also require that the solution-decay-factor q appearing in the quantum complexity does not cancel the speedup.

In general, these methods do offer the possibility of an exponential saving in space, since the quantum methods can represent the vector using a logarithmic number of qubits. Nevertheless, the overall take-home message is that quantum algorithms can potentially outperform classical algorithms, but major gains are only to be expected when the number of spatial dimensions is large, or if there is otherwise a reason that the linear systems involved are much larger than the precision demanded in the output. This intuition is corroborated by the analysis of quantum computing algorithms for ab initio chemistry, where the number of dimensions scales with the number of electrons.

NISQ implementations

Various proposals at NISQ implementations of PDE solvers have been made; see [677] and references therein. The idea is to start from some discretization of the PDE $L|\psi(\theta)\rangle = |b\rangle$, where $|\psi(\theta)\rangle$ is an appropriately chosen variational circuit, and to optimize the parameters θ of the circuit. This is an example of a variational quantum algorithm. Another proposal applies a variational approach to nonlinear PDEs [725]. Note that even if these methods find parameters to generate a good approximation of the solution, they would still pay the $O(1/\epsilon)$ cost to read out properties. Thus, they offer at best a polynomial speedup over classical methods. It is difficult to imagine that sufficient size and precision can be reached in the NISQ regime to be competitive with the best classical solvers.

Outlook

While the simulation of differential equations is one of the most important large-scale computational tasks, constituting a sizable fraction of HPC workloads in industry, at present the benefit of quantum solvers for real-world problems appears limited to relatively modest polynomial speedups. Extensive work on quantum algorithms for solving differential equations has developed methods with rigorous analyses and likely close-to-optimal complexities; the challenge that remains is how these methods can integrate into an end-to-end application pipeline, in such a way as to reduce the cost. To find a high-value application related to solving differential equations (beyond potentially ab initio chemistry), one would likely need to find a situation that meets some or all of the following criteria: (i) involves a very large number of spatial dimensions,

(ii) has simple geometry or initial conditions in order to avoid the need for a QRAM input model, (iii) requires high precision, ruling out heuristic classical approximate methods, (iv) requires learning a relatively small number of properties of (or ideally requires only samples from) the solution vector. There remains the possibility for substantial improvements in memory usage, but this is not currently a bottleneck in classical PDE solving.