Quantum Algorithms for Solving Partial Differential Equations

Arthur Pesah[1]

[1] *Department of Physics and Astronomy, University College London, London WC1E 6BT, UK*

(Dated: March 2020)

Partial Differential Equations (PDEs) are ubiquitous in all scientific fields, and being able to solve them efficiently could unlock significant progress in fields such as aerodynamics, plasma physics and finance. In this case study, I will introduce the different methods that have been proposed in the literature to solve PDEs on a quantum computer, and classify them into a coherent framework. I will then show how to analyze the complexity of those algorithms, and discuss the main challenges and open problems in the field.

I. INTRODUCTION

Most scientific phenomena can be modelled by the evolution of a given quantity through space and time, or in other words, a partial differential equation (PDE). While PDEs can be found everywhere in nature—from plasma physics and fluid mechanics to finance and biology—they are notoriously hard to solve. Therefore, being able to solve them more efficiently than all known classical algorithms, using a quantum computer, could significantly accelerate scientific progress. However, while many quantum algorithms for solving PDEs have been proposed in the literature, it is not yet clear what type of speed-up could be achieved for practical problems. For instance, is an exponential speed-up possible?

To answer this question, we first need to understand the problem being solved by quantum algorithms. In Figure 1, we show the different steps of most quantum PDE solvers: discretization of the PDE, mapping to Schrödinger's equation or to a linear system, execution of either Hamiltonian simulation or the HHL algorithm, and finally measurement. A common feature of those algorithms is that the solution is always stored as the amplitudes of a quantum state. Since extracting the complete solution with quantum state tomography would be impractical, the problem of "solving the PDE" is transformed into the more restricted problem of "obtaining a functional of the solution with an error less than ϵ".

In this report, we will review the main quantum algorithms that can be used to solve PDEs as well as their complexity, with the goal of determining what type of speed-up can be achieved when compared to classical algorithms. After having analyzed the main primitives of quantum PDE solvers, we will discuss each of the steps presented in Figure 1 in more details, by showing how each step can be performed in practice. We will then analyze the overall complexity of those algorithm and deduce some insights on the power of quantum algorithms for solving PDEs. Finally, we will discuss challenges and open problems in the field, suggesting some steps for future research.

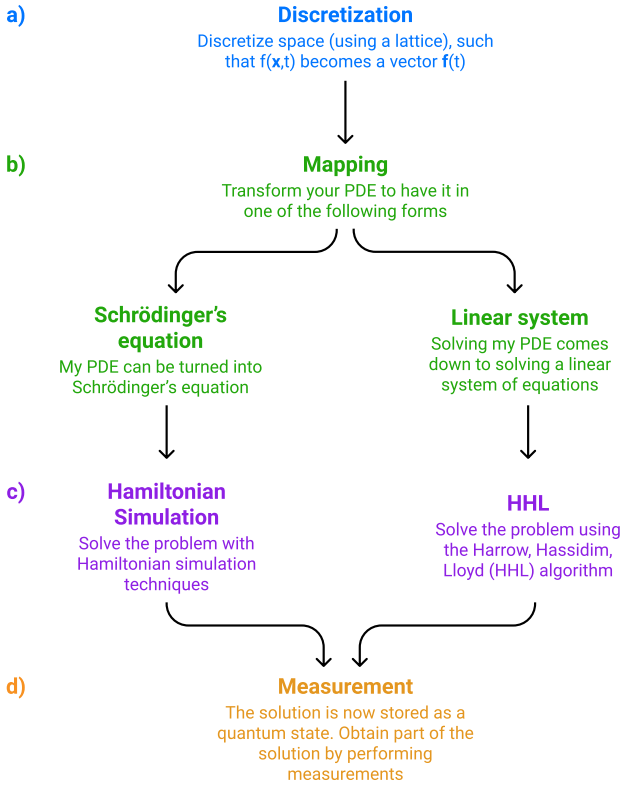

FIG. 1. Common steps to solve a partial differential equation with a quantum computer.

II. BACKGROUND

Most quantum algorithms for PDEs rely on one of those two quantum primitives: Hamiltonian simulation and the quantum linear system solver. We will review those two important quantum algorithms in this section.

A. Hamiltonian simulation

The original motivation for building a quantum computer was to solve Schrödinger's equation. Since then, a lot of progress has been made in determining what quan-

tum systems can be simulated, how and which speed-up can be obtained. The goal of Hamiltonian simulation is to solve

$$\frac{d\psi}{dt} = iH\psi \tag{1}$$

where H is a Hermitian $N \times N$ matrix, by efficiently implementing the unitary evolution

$$|\psi\rangle = e^{-iHt}\,|\psi_0\rangle \tag{2}$$

using a universal gate set. A first solution to this problem was given by Lloyd in 1996 [1], by considering Hamiltonians that can be written as a sum of local Hamiltonians:

$$H = \sum_k H_k \tag{3}$$

In this case, the Trotter formula gives the following decomposition for short-time simulations:

$$e^{-iHt} \approx e^{-iH_n t} \dots e^{-iH_1 t} \tag{4}$$

where each unitary $e^{-iH_k t}$ can in principle be efficiently implemented.

Hamiltonian simulation has later been generalized to sparse matrices [2] and Hamiltonians that can be written as a linear combination of unitaries [3]. A more general framework—called qubitization—led to an optimal complexity of $\mathcal{O}(t + \log(1/\epsilon) + \log(N))$ when H can be *block-encoded* (which includes the d-sparse and the linear combination of unitary models) [4].

The logarithmic scaling of those quantum algorithms in the error and the size of the matrix is an exponential improvement over all known classical methods. For this reason, generalizing Hamiltonian simulation to non-Hermitian matrices has been seen as an attractive approach to solve PDEs on a quantum computer, as we will see in Section III B.

B. Quantum linear equation solver

Introduced in 2008, the Harrow, Hassidim, Lloyd (HHL) algorithm [5] can solve systems of linear equations of the form

$$A\boldsymbol{x} = \boldsymbol{b} \tag{5}$$

in a time logarithmic in the size N of the matrix. However, the solution is stored in the amplitudes of a quantum state,

$$|\boldsymbol{x}\rangle = \sum_i x_i\,|i\rangle \tag{6}$$

and is therefore not classically accessible without performing a (costly) quantum state tomography. HHL is therefore only useful when used as a primitive for other algorithms, or when only a functional of the solution is needed. Other major caveats of this algorithm include:

- It requires the state $|\boldsymbol{b}\rangle$ to be prepared on the quantum computer, which can only be done efficiently for some specific classes of states (e.g. log-concave classical distributions [6])
- While the matrix A is not required to be Hermitian, it must be sparse and well-conditonned
- If A is low-ranked, it can be dequantized, meaning that a classical algorithm has been discovered that can solve the same problem with a polynomial overhead [7–9]. However, it can still be useful if only a polynomial speed-up is sought.

While those caveats will limit the generality of quantum PDE solvers, HHL can still be successfully applied to a large class of problems (e.g. when the initial state is log-concave and the solution is only needed in a small region).

The complexity of the original HHL algorithm was

$$\mathcal{O}(\kappa^2 s^2\,\mathrm{poly}(\log(N), 1/\epsilon)) \tag{7}$$

with κ the condition number and s the sparsity. It has been significantly reduced in the subsequent years, with an improvement in the condition number [10], the sparsity, and the precision [11]. A quantum algorithm based quantum signal processing—a novel technique that has improved and unified several algorithms for matrix arithmetic [12]—has led to an almost-optimal complexity of [13]:

$$\mathcal{O}(\kappa s \log(N) \log(1/\epsilon)) \tag{8}$$

We can now study how those primitives can be applied to solve partial differential equations.

III. SOLVING A PDE WITH A QUANTUM COMPUTER

A. Discretization

Let us consider a partial differential equation that solves for a function $f(\boldsymbol{x}, t)$, where $\boldsymbol{x}$ is a d-dimensional vector. In order to store and manipulate the solution of the PDE on a quantum device, the first step is often to discretize space: we create a d-dimensional lattice and write $f_i(t) := f(\boldsymbol{x}_i, t)$ for the node located at the position $\boldsymbol{x}_i$ in the lattice. Therefore, the problem reduces to solving an ordinary differential equation (ODE) in $\boldsymbol{f}(t)$ and most quantum algorithms to solve ODEs can be applied to our new problem. However, when solving PDEs, the error introduced during the discretization process needs to be taken into account in the complexity analysis. It changes the nature of speed-up that can be obtained, by introducing a dependence between the precision of the solution and the dimension of $\boldsymbol{f}$, as we will see in Section IV.

While discretization is a necessary step for all the quantum algorithms based on the discrete model of quantum computing, continuous-variable quantum computers have the ability to alleviate that step, by directly storing functions of the form $f(\boldsymbol{x}, t)$ as infinite dimensional quantum states, as shown in [14].

B. Mapping

1. *Mapping to Schrodinger's equation*

One of the first approaches proposed to solve ODEs—and by extension PDEs—was to map the equation to a Hamiltonian simulation problem [15, 16]. However, while solving a problem of the form

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} \tag{9}$$

when A is anti-hermitian (i.e. when iA is hermitian) can be handled directly with Hamiltonian simulation methods, more general ODEs and PDEs require more work to be put into this form. I will highlight here three generic techniques that have been proposed in the literature to solve this problem. More specialized mapping have been proposed, tailored to specific equations (e.g. in plasma physics [17, 18]), but we will not discuss those here.

The first technique, proposed in [15], uses the encoding

$$H = \begin{pmatrix} 0 & iA^\dagger \\ -iA & 0 \end{pmatrix} = iA^\dagger \otimes |0\rangle\langle 1| - iA \otimes |1\rangle\langle 0| \tag{10}$$

of the general matrix A into a Hermitian matrix H. Then, using a first-order approximation of the unitary evolution, applied on an initial state $|\boldsymbol{x}_0\rangle |0\rangle$, gives:

$$e^{-iH\epsilon} |\boldsymbol{x}_0\rangle |0\rangle = |\boldsymbol{x}_0\rangle |0\rangle + \epsilon A |\boldsymbol{x}_0\rangle |1\rangle + \dots \tag{11}$$

Therefore, post-selecting on $|1\rangle$ gives us the state $\epsilon A |\boldsymbol{x}_0\rangle$, which can be used when solving Eq. (9) by Euler's method:

$$|\boldsymbol{x}(t+\epsilon)\rangle = |\boldsymbol{x}(t)\rangle + \epsilon A |\boldsymbol{x}(t)\rangle \tag{12}$$

Given an efficient method to prepare the initial state $|\boldsymbol{x_0}\rangle$, the algorithm producing $|\boldsymbol{x}(t = m\epsilon)\rangle$ has a scaling of $\mathcal{O}\left(\left(\frac{1}{\epsilon^2}\right)^m \log(N)\right)$ (where N is the size of the matrix and ϵ the error). As noted in [15], the exponential scaling with time is a major caveat of this method. However, it has the advantage to easily generalize to non-linear ODEs of the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \tag{13}$$

where f is a polynomial of degree p, by applying the algorithm to p copies of $|\boldsymbol{x}\rangle$.

A second technique was proposed in Section II of [16], as a short detour before considering HHL-based methods. The idea is to decompose A as a sum of a Hermitian and a non-Hermitian matrix:

$$A = A_H + A_{aH} \tag{14}$$

Using the Trotter formula, we get for short-time ϵ:

$$\boldsymbol{x}(\epsilon) = e^{A\epsilon}\boldsymbol{x}_0 \approx e^{A_H\epsilon}e^{A_{aH}\epsilon}\boldsymbol{x}_0 \tag{15}$$

The author then notes that A_{aH} can be efficiently simulated using Hamiltonian simulation, and A_H can be simulated using a combination of Hamiltonian simulation and phase estimation (à la HHL). However, the author quickly discard this method, observing that it would have an exponential scaling with time if $[A_H, A_{aH}] \neq 0$. This approach was developed further in [19] in the context of the Black-Scholes PDE

$$\frac{\partial f}{\partial t} = af + b\frac{\partial f}{\partial x} - c\frac{\partial^2 f}{\partial x^2} \tag{16}$$

This equation can be rewritten as

$$\frac{\partial f}{\partial t} = Af \tag{17}$$

where $A = ib\hat{p} + \left(a\mathrm{I} + c\hat{p}^2\right)$, with $\hat{p} = -i\partial_x$. We can then decompose A as before as $A = A_H + A_{aH}$, where

$$A_{aH} = ib\hat{p},\ A_H = a\mathrm{I} + c\hat{p}^2 \tag{18}$$

Since $[A_H, A_{aH}] = 0$, the long-time evolution can be simulated, circumventing the exponential scaling discussed previously. Moreover, the authors propose a new algorithm to simulate the Hermitian part of A, which does not involve any costly phase estimation primitive. Observing that $\hat{O}(t) = e^{A_H t}$ is Hermitian itself, it can be embedded as a unitary operator using a technique called *unitary dilation*:

$$U(t) = \begin{pmatrix} \hat{O} & \sqrt{1-\hat{O}^2} \\ \sqrt{1-\hat{O}^2} & -\hat{O} \end{pmatrix} \tag{19}$$

Applying this unitary to the initial state gives:

$$U(t) |f_0\rangle |0\rangle = \hat{O} |f_0\rangle |0\rangle + \sqrt{1-\hat{O}^2} |f_0\rangle |1\rangle \tag{20}$$

The desired state $\hat{O} |f_0\rangle$ can then be obtained by post-selecting the $|0\rangle$ state on the auxiliary qubit.

The third technique I would like to discuss was proposed in [20] to solve the wave equation

$$\frac{\partial^2 f}{\partial t^2} = -\Delta f \tag{21}$$

where Δ is the d-dimensional Laplacian. It can be written after discretization as:

$$\frac{d^2 f}{dt^2} = -Lf \tag{22}$$

where L is the discretized Laplacian. The authors of [20] then observe that the derivative of the Schrödinger's equation can be written as

$$\frac{d^2\phi}{dt^2} = -H^2\phi \tag{23}$$

and therefore any solution of Schrödinger's equation will also be solution of Eq. (23). Since L cannot always be decomposed as H^2 with H hermitian, we use the encoding

$$H = \begin{pmatrix} 0 & B \\ B^\dagger & 0 \end{pmatrix} \tag{24}$$

such that

$$H^2 = \begin{pmatrix} BB^\dagger & 0 \\ 0 & B^\dagger B \end{pmatrix} \tag{25}$$

Therefore, it is sufficient to find a decomposition $L = BB^\dagger$ to be able to simulate the wave equation using Hamiltonian simulation. For the Laplacian matrix, the matrix B can be obtained analytically as it corresponds to the signed incidence matrix, a matrix which played an important role in the theory of graph Laplacians.

While Hamiltonian simulation is a natural and efficient way to solve many classes of differential equations—first-order PDEs with commuting Hermitian and anti-Hermitian parts, second-order PDEs such as the wave equation, etc.—, not all PDEs can be written as a Schrödinger's equation. Mapping our PDE to a linear system before solving it with variants of HHL is a more general technique that can be applied in a variety of contexts, as we will now see.

2. *Mapping to linear system*

The idea to use HHL for solving PDEs comes from the observation that most classical algorithms for PDEs involve matrix inversion, from the finite-difference and finite-element methods to spectral methods and direct inversion of non-homogeneous problems. As for Hamiltonian simulation, I will discuss here three classes of techniques that are representative of this mapping method.

Our first technique, introduced in [16], is called the multi-step method as it solves the PDE for all times in a single state. Let us consider a discretized PDE of the form:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + \boldsymbol{b} \tag{26}$$

The simplest way to obtain a linear system is to consider Euler's method, i.e. to discretize time to the first order:

$$\frac{\boldsymbol{x}(t_{j+1}) - \boldsymbol{x}(t_j)}{h} \approx A\boldsymbol{x}(t_j) + \boldsymbol{b} \tag{27}$$

Writing $\boldsymbol{x}_j = \boldsymbol{x}(t_j)$, we get the following system for a simple example where $j \leq 2$:

$$\begin{pmatrix} \mathrm{I} & 0 & 0 \\ -(\mathrm{I} + Ah) & \mathrm{I} & 0 \\ 0 & -(\mathrm{I} + Ah) & \mathrm{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_{\text{in}} \\ \boldsymbol{b}h \\ \boldsymbol{b}h \end{pmatrix} \tag{28}$$

We can then apply HHL and obtain the desired solution for all times within a state

$$|\boldsymbol{x}\rangle = \sum_{j=0}^{N_t} |t_j\rangle\,|\boldsymbol{x}_j\rangle \tag{29}$$

This method has for instance been applied and rigorously analyzed for the heat equation [21]. While this technique is the most straight-forward way to encode a PDE into a linear system, it has been shown that it leads to a poor scaling with the precision—$\mathcal{O}(\text{poly}(1/\epsilon))$—even when using improved versions of the HHL algorithm [22]. To circumvent this problem, the authors of [22] introduce a variation of this approach, where Eq. (27) is replaced by the Taylor development in Ah of

$$\boldsymbol{x}_{j+1} = e^{Ah}\boldsymbol{x}_j + \left(e^{Ah} - \mathrm{I}\right) A^{-1}\boldsymbol{b} \tag{30}$$

leading to a high-precision algorithm with a complexity of $\mathcal{O}(\log(1/\epsilon))$ with respect to the error.

A second set of techniques that have been developed in the recent years relies on spectral methods. Instead of encoding the state in the time-domain as we saw with the multi-step method, [23] proposes to encode it in the frequency domain (Fourier basis), leading to a different linear system. As emphasized in their paper, this method has the advantage to easily generalize to time-dependent ODEs of the form

$$\dot{\boldsymbol{x}} = A(t)\boldsymbol{x}(t) + \boldsymbol{b}(t) \tag{31}$$

An improved version of this algorithm adapted to elliptic equations (such as the Poisson equation) gives a complexity of $\mathcal{O}(\log(1/\epsilon))$ with respect to the error [24], similarly to the improved multi-step method.

Finally, our last technique concerns non-homogeneous PDEs, such as the non-homogeneous Poisson equation in d dimensions:

$$-\Delta f = u \tag{32}$$

where Δ is the d-dimensional Laplacian. It is possible to directly solve the infinite-dimensional system of Eq. (32) using a continuous-variable quantum computer with an infinite-dimensional version of HHL [14]. However, for discrete quantum computers, we can consider the discretized version of Eq. (32):

$$-L\boldsymbol{f} = \boldsymbol{u} \tag{33}$$

which is a sparse linear system that can efficiently be solved by HHL (with a runtime polynomial in $1/\epsilon$) [25]. The finite-element method can also be used to solve the

Poisson equation: the discretized Laplacian in Eq. (33) is simply replaced by the matrix obtained when considering the variational (or integral) formulation of the Poisson equation [26] . However, only the second technique discussed here has been proven to achieve a logarithmic complexity with the precision for the Poisson equation.

3. *Other types of mapping*

While Hamiltonian simulation and HHL are the main techniques to solve PDEs on a quantum computers, other types of algorithms can be found in the literature. For instance, one of the first quantum solver proposed in the literature was based on Grover's search, achieving a quadratic speed-up over some classical methods [27]. Building on this method, and particularly amplitude estimation—a generalization of Grover's search—the authors of [28] proposed an algorithm to solve Navier-Stokes equations. Amplitude estimation was also considered as one of the methods to solve the heat equation in [21], using it to improve the random walk algorithm and leading to the best polynomial speed-up for this PDE. Finally, a variational algorithm has been proposed to solve nonlinear PDEs by mapping the solution to the ground-state of well-designed Hamiltonian [29].

C. Initial state preparation

All the algorithms described in the previous sections require the preparation of a state $|\boldsymbol{b}\rangle$, whether it be the initial state of the PDE, an inhomogeneous term, or a combination of both as in Eq. (28).

General states cannot be prepared efficiently on a quantum computer, even approximately. However, in some special cases, efficient state preparation algorithm can be constructed. An example is when the state corresponds to a classical distribution

$$|\psi\rangle = \sum_i \sqrt{p_i}\,|i\rangle \tag{34}$$

such that there exists an efficient classical algorithm that can integrate/sample from p [6]. For instance, log-concave distributions—for which $\frac{\partial^2 \log(p(x))}{\partial x^2} < 0$, such as normal and exponential distributions—can be shown to be efficiently integrable. Examples of algorithms that can be used to prepare such states are the uniformly-controlled rotations [30] and the quantum Generative Adversarial Network [31] .

While this establishes an important constraint on quantum PDE solvers, it is often possible to find physically relevant initial states that are efficiently preparable. For instance, [17] shows that a Maxwellian distribution at rest can be efficiently implemented and used to solve the Vlasov equation. In finance, one can show that the initial condition for the pricing problem of a European put option is log-concave, and can therefore efficiently be used when solving the Black-Scholes PDE [19].

D. Measurement

Finally, the last step of any quantum PDE algorithm is to perform measurements to obtain a function of the solution, since the global solution cannot be easily extracted in general. There are many functions of a state that can be extracted efficiently, such as the inner product with another state (using the so-called SWAP test) or a small set of amplitudes (using amplitude estimation). For instance, [18] enumerates a few quantities of interest that can be extracted from the solution of several plasma equations, e.g. the total energy of a wave or the power dissipated in a finite volume.

For elliptic equations (such as the Poisson equation), long-range effects can play an important role. Therefore, knowing the value of the solution in a small region of interest often requires solving it in a larger region that includes the boundary conditions. Extracting the solution in the small region could therefore be both interesting and not costly.

However, measuring any functional requires $\mathcal{O}(\text{poly}(1/\epsilon))$ repetitions to obtain a precision ϵ, as noted in the original HHL paper [5]. This cost is unavoidable (under reasonable complexity assumptions) and is therefore an important bottleneck in analyzing the complexity of a quantum algorithm for PDEs, as we will see in the next section.

IV. COMPLEXITY ANALYSIS

In this section, we will analyze the overall complexity of the quantum algorithms discussed in the previous section, with the goal of determining if an exponential speed-up is possible. We consider the problem of solving a general linear PDE with either Hamiltonian simulation or HHL, and measuring a functional of the result at the end.

A. Variables of interest

We analyze here a quantum algorithm that takes as input a d-dimensional PDE (i.e. with d spatial variables) and output an approximation of the functional $B(f)$ of the solution f, with an error ϵ. This solution is often discretized in a lattice made of N nodes, leading to inversion or simulation problems involving $N \times N$-matrices. Therefore, we can enumerate three variables that can be used in the complexity analysis:

- ϵ: the error in the approximation
- d: the dimension of the PDE

- N: the number of nodes in the discretization

However, as observed in [25, 26], N and ϵ are related in PDE problems: to reduce the error of a solution, increasing the number of nodes in the discretization is necessary. Therefore, it is possible to show for many equations that achieving a discretization error less than ϵ requires to take

$$N = \mathcal{O}\left(\text{poly}\left(\frac{1}{\epsilon^d}\right)\right) \tag{35}$$

where the degree of the polynomial depends on the order of the discretization but is independent on d [21, 25, 26]. Therefore, our only remaining variables are ϵ and d, and claims of exponential speed-up for PDE problems due to $\log(N)$ factors (such as [19, 32]) need to be considered carefully.

B. Analysis of the different components

As we have seen, a quantum algorithm for PDEs is made of three important steps: initial state preparation, quantum algorithm, and measurement. For log-concave distributions, the initial state can be efficiently prepared, i.e. in less than $\mathcal{O}(\text{polylog}(N))$. While we have considered a large variety of quantum algorithms with different complexities, the most promising approaches required a number of step scaling as $\mathcal{O}(\text{polylog}(N, 1/\epsilon))$ (assuming that the sparsity and condition number scale at most logarithmically with N and $1/\epsilon$). Finally, we have seen that the measurement process cannot take less than $\mathcal{O}(\text{poly}(1/\epsilon))$.

Adding all those costs, we end up with a total complexity of

$$\begin{aligned} &\mathcal{O}(\text{polylog}(N, 1/\epsilon)\,\text{poly}(1/\epsilon)) \\ &= \mathcal{O}(\text{polylog}(1/\epsilon^d)\,\text{poly}(1/\epsilon)) \\ &= \mathcal{O}(\text{poly}(d, 1/\epsilon)) \end{aligned} \tag{36}$$

Therefore, all the algorithms we have considered in this case study will have complexity scaling at best polynomially in d and $1/\epsilon$.

C. Is an exponential speed-up in solving PDEs possible?

In order to determine if a quantum algorithm can achieve an exponential speed-up, we need to compare it to the best classical algorithms. In general, it seems that most PDEs have a best classical solution scaling as either $\mathcal{O}(\text{poly}(1/\epsilon^d))$ or $\mathcal{O}(\text{poly}(1/\epsilon))$. For instance, random walks give the best complexity for the heat equation, with a complexity scaling as $\mathcal{O}(1/\epsilon^2)$ [21]. On the other hand, it can be shown that solving an elliptic PDE such as the Poisson equation requires $\mathcal{O}(\text{poly}(1/\epsilon^d)$ steps in the worst-case, a phenomenon that is sometimes referred to as the *curse of dimensionality* [33].

From those considerations, it seems that an exponential speed-up cannot be easily achieved with respect to ϵ (for fixed d), while an exponential speed-up with respect to the dimension has been shown in several cases [14, 25, 26] and suggests a promising direction to solve the curse of dimensionality using quantum computers.

V. DISCUSSION AND OPEN PROBLEMS

As we have seen throughout this case study, a large variety of quantum algorithms have been proposed to solve the three main types of PDEs—elliptic (Poisson), hyperbolic (wave) and parabolic (heat)—through the use of clever mappings to existing quantum primitives. While recent work has led to a large decrease in the overall complexity of those algorithms—building on the improvements of Hamiltonian simulation and HHL over the years—the need to measure a functional at the end has not been alleviated and will probably always make exponential speed-ups in the precision impossible.

I can identify several open problems and directions for future research:

1. What is the extent of the polynomial speed-up that can be achieved with quantum PDE algorithms? In particular, can we achieve a super-quadratic speed-up in a real-world application? It has recently been shown that quadratic speed-ups will probably not be practical for a long period [34], making the discovery of super-quadratic speed-ups an important research direction. Rigorous analyses of polynomial speed-ups for end-to-end quantum PDE solvers have only been performed in a small number of cases (Poisson equation [26], heat equation [21], and Black-Scholes SDE [35] for instance). Extending those analyses to more equations would help to better understand the speed-ups achievable with quantum algorithms.

2. Is an exponential speed-up in the dimension possible, and if so, in what settings? While some papers have shown that such a speed-up is possible for the Poisson equation [14, 26], those quantum algorithm need to be compared to a broader class of classical algorithms designed under the same assumptions. If such a speed-up is indeed possible, what practical problems will it allow to solve?

3. Can we use quantum algorithms to solve non-linear PDEs? Recent papers have presented new methods to solve non-linear ODEs efficiently, based either on concatenating multiple copies of the states [29, 36] or on Carleman linearization [37]. However, as the authors do not take into account the cost of discretization, new analyses need to be performed to see if the quantum speed-up shown for ODEs generalizes to PDEs.

4. Solving first-order ODEs often comes down to implementing imaginary-time evolution on the quantum computer. Can we use the recent results in simulating imaginary-time evolution [38, 39] to create new quantum algorithms for PDEs/ODEs?

While the existing proposals to solve PDEs on a quantum computer come with many caveats, the potentially large polynomial speed-up of those algorithms as well their ability to solve the curse of dimensionality makes them promising candidates to solve real-world applications with a quantum computer, and a lot still remains to be discovered about them.

[1] Seth Lloyd, "Universal quantum simulators," Science , 1073–1078 (1996).

[2] Dorit Aharonov and Amnon Ta-Shma, "Adiabatic quantum state generation and statistical zero knowledge," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing* (2003) pp. 20–29.

[3] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma, "Simulating hamiltonian dynamics with a truncated taylor series," Physical review letters **114**, 090502 (2015).

[4] Guang Hao Low and Isaac L Chuang, "Hamiltonian simulation by qubitization," Quantum **3**, 163 (2019).

[5] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd, "Quantum algorithm for linear systems of equations," Physical review letters **103**, 150502 (2009).

[6] Lov Grover and Terry Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions," arXiv preprint quant-ph/0208112 (2002).

[7] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang, "Quantum-inspired sublinear classical algorithms for solving low-rank linear systems," arXiv preprint arXiv:1811.04852 (2018).

[8] András Gilyén, Seth Lloyd, and Ewin Tang, "Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension," arXiv preprint arXiv:1811.04909 (2018).

[9] Changpeng Shao and Ashley Montanaro, "Faster quantum-inspired algorithms for solving linear systems," arXiv preprint arXiv:2103.10309 (2021).

[10] Andris Ambainis, "Variable time amplitude amplification and quantum algorithms for linear algebra problems," in *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, Vol. 14 (LIPIcs, 2012) pp. 636–647.

[11] Andrew M Childs, Robin Kothari, and Rolando D Somma, "Quantum linear systems algorithm with exponentially improved dependence on precision," arXiv preprint arXiv:1511.02306 **83** (2015).

[12] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang, "A grand unification of quantum algorithms," arXiv preprint arXiv:2105.02859 (2021).

[13] Lin Lin and Yu Tong, "Optimal quantum eigenstate filtering with application to solving quantum linear systems," arXiv preprint arXiv:1910.14596 (2019).

[14] Juan Miguel Arrazola, Timjan Kalajdzievski, Christian Weedbrook, and Seth Lloyd, "Quantum algorithm for nonhomogeneous linear partial differential equations," Physical Review A **100**, 032306 (2019).

[15] Sarah K Leyton and Tobias J Osborne, "A quantum algorithm to solve nonlinear differential equations," arXiv preprint arXiv:0812.4423 (2008).

[16] Dominic W Berry, "High-order quantum algorithm for solving linear differential equations," Journal of Physics A: Mathematical and Theoretical **47**, 105301 (2014).

[17] Alexander Engel, Graeme Smith, and Scott E Parker, "Quantum algorithm for the vlasov equation," Physical Review A **100**, 062315 (2019).

[18] Ilya Y Dodin and Edward A Startsev, "On applications of quantum computing to plasma simulations," arXiv preprint arXiv:2005.14369 (2020).

[19] Javier Gonzalez-Conde, Ángel Rodríguez-Rozas, Enrique Solano, and Mikel Sanz, "Pricing financial derivatives with exponential quantum speedup," arXiv preprint arXiv:2101.04023 (2021).

[20] Pedro CS Costa, Stephen Jordan, and Aaron Ostrander, "Quantum algorithm for simulating the wave equation," Physical Review A **99**, 012323 (2019).

[21] Noah Linden, Ashley Montanaro, and Changpeng Shao, "Quantum vs. classical algorithms for solving the heat equation," arXiv preprint arXiv:2004.06516 (2020).

[22] Dominic W Berry, Andrew M Childs, Aaron Ostrander, and Guoming Wang, "Quantum algorithm for linear differential equations with exponentially improved dependence on precision," Communications in Mathematical Physics **356**, 1057–1081 (2017).

[23] Andrew M Childs and Jin-Peng Liu, "Quantum spectral methods for differential equations," Communications in Mathematical Physics , 1–31 (2020).

[24] Andrew M Childs, Jin-Peng Liu, and Aaron Ostrander, "High-precision quantum algorithms for partial differential equations," arXiv preprint arXiv:2002.07868 (2020).

[25] Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, and Sabre Kais, "Quantum algorithm and circuit design solving the poisson equation," New Journal of Physics **15**, 013021 (2013).

[26] Ashley Montanaro and Sam Pallister, "Quantum algorithms and the finite element method," Physical Review A **93**, 032324 (2016).

[27] Bolesław Kacewicz, "Randomized and quantum algorithms yield a speed-up for initial-value problems," Journal of Complexity **20**, 821–834 (2004).

[28] Frank Gaitan, "Finding flows of a navier–stokes fluid through quantum computing," npj Quantum Information **6**, 1–6 (2020).

[29] Michael Lubasch, Jaewoo Joo, Pierre Moinier, Martin Kiffner, and Dieter Jaksch, "Variational quantum algorithms for nonlinear problems," Physical Review A **101**, 010301 (2020).

[30] Mikko Mottonen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa, "Transformation of quantum states using uniformly controlled rotations," arXiv preprint quant-ph/0407010 (2004).

[31] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner, "Quantum generative adversarial networks for learning and loading random distributions," npj Quantum Infor-

mation **5**, 1–9 (2019).
[32] B David Clader, Bryan C Jacobs, and Chad R Sprouse, "Preconditioned quantum linear system algorithm," Physical review letters **110**, 250504 (2013).
[33] Arthur G Werschulz, *The computational complexity of differential and integral equations: An information-based approach* (Oxford University Press, Inc., 1991).
[34] Ryan Babbush, Jarrod R McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven, "Focus beyond quadratic speedups for error-corrected quantum advantage," PRX Quantum **2**, 010103 (2021).
[35] Dong An, Noah Linden, Jin-Peng Liu, Ashley Montanaro, Changpeng Shao, and Jiasu Wang, "Quantum-accelerated multilevel monte carlo methods for stochastic differential equations in mathematical finance," arXiv preprint arXiv:2012.06283 (2020).
[36] Seth Lloyd, Giacomo De Palma, Can Gokler, Bobak Kiani, Zi-Wen Liu, Milad Marvian, Felix Tennie, and Tim Palmer, "Quantum algorithm for nonlinear differential equations," arXiv preprint arXiv:2011.06571 (2020).
[37] Jin-Peng Liu, Herman Øie Kolden, Hari K Krovi, Nuno F Loureiro, Konstantina Trivisa, and Andrew M Childs, "Efficient quantum algorithm for dissipative nonlinear differential equations," arXiv preprint arXiv:2011.03185 (2020).
[38] Mario Motta, Chong Sun, Adrian TK Tan, Matthew J O'Rourke, Erika Ye, Austin J Minnich, Fernando GSL Brandao, and Garnet Kin-Lic Chan, "Determining eigenstates and thermal states on a quantum computer using quantum imaginary time evolution," Nature Physics **16**, 205–210 (2020).
[39] Sam McArdle, Tyson Jones, Suguru Endo, Ying Li, Simon C Benjamin, and Xiao Yuan, "Variational ansatz-based quantum simulation of imaginary time evolution," npj Quantum Information **5**, 1–6 (2019).