Quantum algorithms and the finite element method

Ashley Montanaro and Sam Pallister[*]
School of Mathematics, University of Bristol, Bristol BS8 1TW, United Kingdom

(Received 5 January 2016; published 17 March 2016)

The finite element method is used to approximately solve boundary value problems for differential equations. The method discretizes the parameter space and finds an approximate solution by solving a large system of linear equations. Here we investigate the extent to which the finite element method can be accelerated using an efficient quantum algorithm for solving linear equations. We consider the representative general question of approximately computing a linear functional of the solution to a boundary value problem and compare the quantum algorithm's theoretical performance with that of a standard classical algorithm—the conjugate gradient method. Prior work claimed that the quantum algorithm could be exponentially faster but did not determine the overall classical and quantum run times required to achieve a predetermined solution accuracy. Taking this into account, we find that the quantum algorithm can achieve a polynomial speedup, the extent of which grows with the dimension of the partial differential equation. In addition, we give evidence that no improvement of the quantum algorithm can lead to a superpolynomial speedup when the dimension is fixed and the solution satisfies certain smoothness properties.

DOI: 10.1103/PhysRevA.93.032324

I. INTRODUCTION

The development of a quantum algorithm for large systems of linear equations is an exciting recent advance in the field of quantum algorithmics. First introduced by Harrow, Hassidim, and Lloyd (HHL) [1], and later improved by other authors [2,3], the algorithm gives an exponential quantum speedup over classical algorithms for solving linear systems. However, the quantum linear equation (QLE) algorithm "solves" a system of equations $A\mathbf{x} = \mathbf{b}$ in an unusually quantum sense. The input $\mathbf{b}$ is provided as a quantum state $|b\rangle$, and the algorithm produces another state $|x\rangle$ corresponding to the desired output $\mathbf{x}$. Whether this is considered to be a reasonable definition of "solution" depends on the intended application [4]. Still, linear equations are so ubiquitous in science and engineering that many applications of the QLE algorithm have been proposed, ranging from machine learning [5] to computing properties of electrical networks [6].

One area in which large systems of linear equations occur is the finite element method (FEM) [7–10]. The FEM is a technique for efficiently finding numerical approximations to the solutions of boundary value problems (BVPs) for partial differential equations, based on discretizing the parameter space via a finite mesh. The FEM is a tempting target for acceleration by the QLE algorithm for several reasons. First, the large systems of linear equations that occur in the FEM are produced algorithmically, rather than being given directly as input. This avoids efficiency issues associated with needing to access data via a quantum RAM [4,5]. Second, the FEM naturally leads to sparse systems of linear equations, which is usually a requirement for quantum speedup via the QLE algorithm. Third, the FEM has many important practical applications. These include structural mechanics, thermal physics, and fluid dynamics [10]. Any quantum speedup for the FEM would thus represent a compelling application of quantum computers.

Clader, Jacobs, and Sprouse [11] have studied the application of the QLE algorithm to the FEM. In particular, they consider an electromagnetic scattering cross-section problem solved via the FEM and argue that the quantum algorithm achieves an exponential speedup for this problem over the best classical algorithm known. In order to achieve this result, the authors of [11] propose ways to avoid issues with the QLE algorithm that can reduce or eliminate a quantum speedup. For example, they show that the important classical technique known as preconditioning, which reduces the condition number of the input matrix A, can be applied within the quantum algorithm.

However, the analysis of [11] does not fully calculate and combine all contributions to the complexity of approximately solving the scattering cross-section problem. The classical and quantum algorithmic complexity is calculated in [11] in terms of two parameters: N (the size of the system of linear equations resulting from applying the FEM) and ϵ (the solution accuracy). The size of the system of equations is a parameter which can be chosen by the user in order to achieve a desired accuracy (i.e., N and ϵ are formally related). In [11] they are treated as independent parameters and hence the complexity analysis is left incomplete. If the scaling of N with ϵ is benign, the classical algorithm might not need to solve a large system of equations to achieve a given accuracy, so the quantum speedup could be reduced or even eliminated.

A. New results

In this paper we work through the details of applying the QLE algorithm to the general FEM and compare the worst-case performance of the quantum algorithm with that of a simple standard classical algorithm. We choose a representative general problem—approximating a linear functional of the solution to a BVP corresponding to an elliptic PDE—which allows the two types of algorithm to be fairly compared.

Our results can be summarized as follows: We find that the QLE algorithm is indeed applicable to the general FEM and can achieve substantial speedups over the classical algorithm.

[*]sam.pallister@bristol.ac.uk

TABLE I. Complexity comparison of the algorithms studied in this work. Quantities listed are the worst-case time complexities of approximating a linear functional of the solution to a d-dimensional BVP up to accuracy ϵ, using the FEM with linear basis functions (see Sec. III for bounds when using higher-degree polynomials). $\|u\|$, $|u|_\ell$, and $\|u\|_\ell$ are the L^2 norm, Sobolev ℓ-seminorm, and Sobolev ℓ-norm of the solution, respectively, defined in Sec. II. The $\widetilde{O}$ notation hides polylogarithmic factors.

Algorithm	No preconditioning	Optimal preconditioning
Classical	$\widetilde{O}((\lvert u\rvert_2/\epsilon)^{(d+1)/2})$	$\widetilde{O}((\lvert u\rvert_2/\epsilon)^{d/2})$
Quantum	$\widetilde{O}(\|u\|\lvert u\rvert_2^2/\epsilon^3 + \|u\|_1\lvert u\rvert_2/\epsilon^2)$	$\widetilde{O}(\|u\|_1/\epsilon)$

However, the quantum speedup obtained is only at most polynomial, if the spatial dimension is fixed and the solution satisfies certain smoothness properties. For example, the maximal advantage of the quantum algorithm for the typical physically relevant PDE defined over $3 + 1$ dimensions (three spatial and one temporal, such that $d = 4$) is approximately quadratic. In a small enough dimension, and if the solution is sufficiently smooth, the run time of the quantum algorithm can actually be worse than the classical algorithm.

Examples of the bounds we derive are listed in Table I, which includes the effect of preconditioning on the run time of the algorithms. Note that in general it is difficult to rigorously analyze the performance of preconditioners. We therefore choose to highlight two extreme possibilities: no preconditioning at all is applied or maximally successful preconditioning is used. The true performance of an algorithm using preconditioning will fall somewhere between these two cases.

The run time of both the classical and the quantum algorithms depends on the Sobolev ℓ-seminorm and Sobolev ℓ-norm of the solution to the BVP, for some ℓ; roughly speaking, these measure the size of the ℓth derivatives of the solution. Assuming that preconditioning has been optimally used within the QLE algorithm, the quantum algorithm's run time is dependent only on the Sobolev ℓ-norm (up to polylogarithmic terms). However, the classical algorithm's run time depends on the Sobolev ℓ-seminorm, for some $\ell \geqslant 2$. Therefore, for problems with solutions whose higher-order derivatives are large, the quantum advantage could be substantial.

Perhaps more importantly, to achieve accuracy ϵ in spatial dimension d, the run time of the classical algorithm scales as $\epsilon^{-O(d)}$, while the scaling with ϵ of the quantum algorithm's run time does not depend on d. For higher-dimensional problems, the quantum speedup can thus be very significant. Interestingly, this holds even if preconditioning is not used. (Note that we cannot quite say that the quantum algorithm achieves an *exponential* speedup, as the run time also contains a dimension-dependent constant factor which may be very large.)

One example application is any dynamical problem involving n bodies, which implies solving a PDE defined over a configuration space of dimension $2n$. Also, there may be a significant advantage for problems in mathematical finance; for example, pricing multiasset options requires solving the Black-Scholes equation over a domain with dimension given by the number of assets [12]. This is discussed further in Sec. V.

The reason for the apparent contradiction between our results and previous work [11], which claimed an exponential speedup in fixed spatial dimension, is the inclusion of an accuracy parameter in the run time, which was not fully incorporated in [11]. Imagine that we would like to produce a solution to some BVP that is accurate up to ϵ. This accuracy parameter will affect the run time of algorithms for the FEM. There are two potential sources of error in producing the solution: the discretization process, which converts the problem to a system of linear equations, and any inaccuracies in solving the system of equations itself and computing the desired function of the solution. The larger the system of equations produced, the smaller the first type of error is.

The QLE algorithm can work with an exponentially larger set of equations in a comparable time to the classical algorithm, so this source of error can be reduced exponentially. However, the scaling with accuracy of the QLE algorithm's extraction of a solution from the system of linear equations is substantially *worse* than that of the classical algorithm. These two effects can come close to canceling each other out.

We remark that there is a subtle point here: the scaling with accuracy of the quantum algorithm is substantially better if we only wish to produce the quantum state corresponding to the solution to the FEM [3], rather than computing some property of the state by measuring it. However, in applications one will always eventually want to perform a measurement to extract information from the final output of the quantum algorithm. We therefore consider it reasonable to compare the quantum and classical complexities of producing a (classical) answer to some given problem.

Finally, we argue that the inability of the quantum algorithm to deliver exponential speedups (in some cases) is not a limitation of the algorithm itself but, rather, any quantum algorithm for the FEM will face similar constraints. We elucidate several barriers with which any quantum algorithm will have to contend. First, we show that, informally, any algorithm which needs to distinguish between two states which are distance ϵ apart must have run time $\Omega(1/\sqrt{\epsilon})$. Second, we argue that the "FEM solving subroutine" of any quantum algorithm can likely be replaced with an equivalent classical subroutine with at most a polynomial slowdown (in fixed spatial dimension and when the solution is smooth). Third, we show that there can be no more than a quadratic speedup if the input to the problem is arbitrary and accessed via queries to a black box or "oracle."

Our results pinpoint the regimes in which one can hope to achieve exponential quantum speedups for the FEM and show that apparent speedups can disappear when one takes the effect of solution accuracy into account. Nevertheless, we believe that the fact that exponential speedups might still be obtained in some cases is encouraging and an incentive to focus on problems with the possibility of a genuine exponential quantum speedup.

B. Other related work

An alternative approach to the approximate numerical solution of PDEs is the finite-difference method (FDM). This method is also based on discretization of the problem domain but differs from the FEM in that it approximates the partial derivatives in the original problem with finite differences.

The QLE algorithm can also be applied to the FDM. One example where this has been done, and described in detail, is in the work of Cao *et al.* [13], who give a quantum algorithm for the Poisson equation in d dimensions. Their algorithm produces a quantum state corresponding to the solution to the equation in time $O(\max\{d, \log 1/\epsilon\} \log^3 1/\epsilon)$. Note that this scaling with ϵ is exponentially better than the best general results on Hamiltonian simulation known at the time; their algorithm used special properties of the Poisson equation to achieve an improved run time. The best classical algorithms require time $\epsilon^{-\Omega(d)}$, as they solve a discretized version of the problem on a d-dimensional grid with cells of size $\epsilon \times \epsilon \times \ldots \times \epsilon$.

However, the quantum algorithm of Cao *et al.* [13] shares the property of the FEM algorithms discussed here that, in order to extract some information from the quantum state produced, one finishes with a scaling with ϵ which is poly$(1/\epsilon)$. In the physically realistic setting of the dimension d being fixed and the accuracy ϵ being the parameter of interest, this is only a polynomial improvement.

Other related work has given quantum algorithms for solving large systems of sparse linear [14] or nonlinear [15] differential equations via Euler's method. In these cases the quantum algorithms can, in principle, achieve an exponential improvement over classical computation for approximately computing properties of the solution to the system, if the system of equations is provided implicitly. Fleshing out this approach requires also specifying how the equations are produced and how the property of interest is computed. If the equations are generated by a discretization procedure such as the FDM, qualitative conclusions similar to those we derive for the FEM seem likely to hold.

C. Organization and notation

We begin, in Sec. II, by introducing the FEM and describing its classical complexity. Section III goes through the details of applying the QLE algorithm to the FEM and determines its complexity. In Sec. IV we describe various limitations on the quantum algorithm. We conclude in Sec. V with some discussion and open problems.

We need to deal with continuous functions, their discretized approximations as vectors, and their corresponding quantum states. Italics denote functions, boldface denotes vectors, and quantum states (usually normalized) are represented as kets. We often let $\Omega \subseteq \mathbb{R}^d$ denote an arbitrary convex set. For a function $f \in L^2(\Omega)$, $\|f\| := (\int_\Omega f(x)^2 dx)^{1/2}$ denotes the L^2 norm of f. For a vector $\mathbf{f}$, $\|\mathbf{f}\| := (\sum_i \mathbf{f}_i^2)^{1/2}$ denotes the ℓ_2 norm of $\mathbf{f}$.

We often use the term "spatial dimension" as shorthand for "number of degrees of freedom in the given PDE," distinct from the dimension of the vector space used for a discretized approximation of the solution of a PDE or the dimension of the Hilbert space acted on by a quantum algorithm; this is merely for convenience and should not be taken to mean that the only PDEs of interest are those in which the degrees of freedom are physical spatial dimensions.

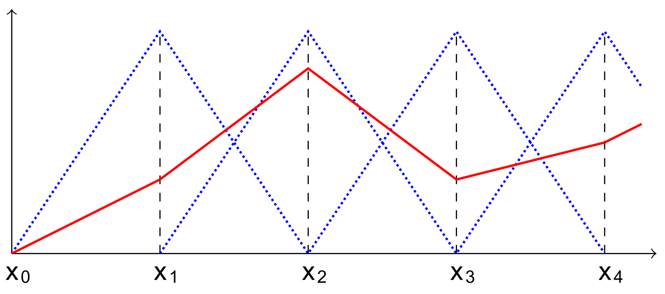

FIG. 1. A basis set of "tent" functions (dotted blue lines) for piecewise linear functions defined on the line (an example of which is given by the solid red line). Any piecewise linear function can be uniquely specified as a sum of scaled tents.

II. THE FINITE ELEMENT METHOD

Rather than provide a formal introduction to the FEM, it is easiest to motivate the procedure via an example (we refer the reader to [7], [8] and [9] for a thorough treatment). Imagine we would like to solve Poisson's equation in the interval [0,1], in one dimension:

$$u'' = f; \quad u(0) = u'(1) = 0.$$

Here f is the input to the problem and we fix the boundary conditions $u(0)$, $u'(1)$. Given a sufficiently smooth "test function" $v \in L^2[0,1]$ such that $v(0) = 0$, one can multiply both sides by v and then integrate by parts:

$$\int_0^1 f(x)v(x)dx = \int_0^1 u''(x)v(x)dx = -\int_0^1 u'(x)v'(x)dx.$$

Assuming certain regularity properties of f, a function u which satisfies this equality for all test functions v will satisfy Poisson's equation. This is known as the *weak formulation* of Poisson's equation. The goal is to reduce this formulation to a problem that is tractable computationally. The approximation is to consider solutions and test functions that instead exist in some finite-dimensional subspace S of $L^2[0,1]$. Denote the approximate solution as $\tilde{u}$, such that $\tilde{u} \in S \subset L^2[0,1]$. Commonly S is taken to be the space of piecewise polynomial functions of some degree k; the choice of "pieces" for these functions is the origin of the finite element mesh.

A particularly simple choice of basis for this example is the space of piecewise linear functions on [0,1], divided into N intervals of size h. A basis for this space is the set of "tent" functions (see Fig. 1), defined as

$$\phi_i(x) = \begin{cases} \frac{1}{h}(x - x_{i-1}) & if \quad x \in [x_{i-1}, x_i], \\ \frac{1}{h}(x_{i+1} - x) & if \quad x \in [x_i, x_{i+1}], \\ 0 & otherwise. \end{cases}$$

More generally, consider some choice of basis for the space S, denoted $B = \{\phi_i\}$, such that $|B| = N$. We choose a basis such that $\phi_i(0) = \phi_i'(1) = 0$, so that every function in S satisfies the boundary conditions. Then $\tilde{u}$ can be expanded in this basis: $\tilde{u} = \sum_j U_j \phi_j$. The corresponding weak formulation of Poisson's equation is

$$-\sum_j U_j \int_0^1 \phi_j'(x)v'(x)dx = \int_0^1 f(x)v(x)\,dx.$$

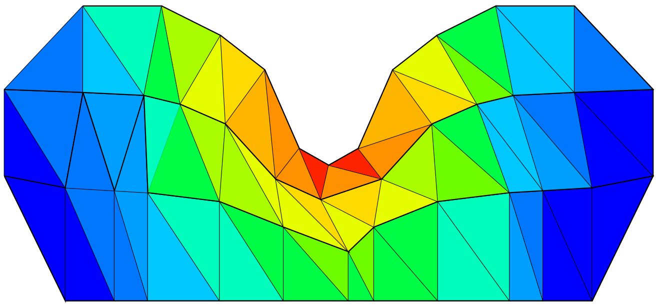

FIG. 2. An example of a "mesh"—a discretization of the domain over which the PDE is defined. Each polygon is a "finite element," with basis functions defined on them. The shading of each polygon represents the amplitude associated with the function supported on each finite element. As a physical example, the diagram could represent the material stress on a plate induced by a deformation by a rod.

For this condition to hold for all $v \in S$, it is sufficient for it to hold on all basis functions ϕ_i:

$$-\sum_j U_j \int_0^1 \phi_j'(x)\phi_i'(x)dx = \int_0^1 f(x)\phi_i(x)\,dx.$$

If we define N-dimensional vectors $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{f}}$ such that

$$\tilde{\mathbf{u}}_i = U_i, \quad \tilde{\mathbf{f}}_i = \int_0^1 f(x)\phi_i(x)\,dx$$

and an $N \times N$ matrix M such that

$$M_{ij} = \int_0^1 \phi_i'(x)\phi_j'(x)dx, \tag{1}$$

then the approximate solution to Poisson's equation can be determined by solving the linear system

$$M\tilde{\mathbf{u}} = \tilde{\mathbf{f}}. \tag{2}$$

This general procedure (expressing the PDE in the weak formulation, choosing a finite element mesh and basis functions, and solving the resultant linear system of equations) can be extended to far more complicated PDEs, domains, and boundary conditions. In higher spatial dimensions, the above framework can be naturally generalized as follows. The uniform division of $[0,1]$ into intervals is replaced with a suitably regular division of the domain into a mesh, whose elements are usually polygons (for example, triangles) or polyhedra. An example of a mesh is shown in Fig. 2. The space S is replaced with the space of piecewise polynomials of degree k on the elements of the mesh, with a basis $\{\phi_i\}$ of polynomials supported only on adjacent mesh elements. Finally, the matrix M defined in (1) is modified such that $M_{ij} = a(\phi_i,\phi_j)$, where $a(u,v)$ is an inner product depending on the PDE in question.

Here we choose not to specify which PDE we wish to solve, as the details of this procedure for particular PDEs will not be very significant when making a general comparison of quantum and classical algorithms for the FEM. However, we restrict ourselves to elliptic second-order PDEs throughout, to avoid some technical complications. Even with this restriction, the following analysis captures many examples of physical interest, for example, electrostatics, subsonic fluid dynamics, and linear elasticity.

A. Comparing quantum and classical algorithms for the FEM

The goal of this paper is to compare the performance of quantum and classical algorithms for solving BVPs via the FEM. However, the quantum algorithm does not allow the full solution u to a given BVP to be obtained but does allow certain properties of u to be approximately computed. In order to fairly compare classical and quantum algorithms for solving general BVPs, we consider the representative problem of computing a linear functional of u. That is, for some known function $r : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^d$, we seek to compute

$$\langle r,u\rangle := \int_\Omega r(\mathbf{x})u(\mathbf{x})d\mathbf{x}.$$

This is one of the simplest properties of u one could hope to access. In general, we do not have complete knowledge of u but have some approximation $\tilde{u}$. Although there are many sensible norms with which one could measure the quality of this approximation, one natural choice is the L^2 norm, $\|f\| := (\int_\Omega f(x)^2dx)^{1/2}$. Then

$$|\langle r,\tilde{u}\rangle - \langle r,u\rangle| = |\langle r,(\tilde{u}-u)\rangle| \leqslant \|r\|\|\tilde{u}-u\|$$

by Cauchy-Schwarz. Hence an accuracy of ϵ in the L^2 norm in an approximation of u translates into an additive error of at most $\epsilon\|r\|$ in an approximation of $\langle r,u\rangle$. Therefore, approximating $\langle r,u\rangle$ up to accuracy $\epsilon\|r\|$ is the prototypical problem considered throughout.

B. Approximation errors

If u is the exact solution to a BVP, henceforth let $\tilde{u}$ be the continuous, exact solution corresponding to the discretized problem, (2); $\tilde{u}$ is the solution that a perfect linear-system solver would find. In general, however, the linear-system solver is iterative and so will not truly reach $\tilde{u}$; so also let $\tilde{\tilde{u}}$ be the continuous, approximate solution generated by the linear-system solver.

Crucially, one can show that $\tilde{u}$ can be made quite close to u by taking a sufficiently fine mesh. Indeed, consider a second-order differential equation defined over a polygonal, d-dimensional domain (or, equivalently, define d as the number of degrees of freedom in the PDE). Then take an infinite, ordered family of progressively finer meshes $\{\mathcal{M}_r\}_{r=1}^\infty$, constructed from a triangulation of the domain with simplices of dimension d. Let k be the total degree of the polynomials used as basis functions. We assume throughout that both d and k are fixed. Given a parameter $m \in \{0,1\}$, and provided that $d > 2(k-m)$, that all angles in the mesh are bounded below by some fixed value, and that the greatest edge length h in the mesh goes to 0, then the following bound is known ([9], Thm. 3.2.1):

$$|u-\tilde{u}|_m \leqslant Ch^{k+1-m}|u|_{k+1}, \quad h \to 0, \tag{3}$$

assuming that weak derivatives of u of order m exist. Here C is a constant, independent of h (but not necessarily independent of d or the definition of the mesh). $|\cdot|_m$ is the Sobolev seminorm

$$|v|_m := \left(\sum_{\alpha,|\alpha|=m} \|\partial^\alpha v\|^2 \right)^{1/2}.$$

Here $\alpha = (\alpha_1,\dots,\alpha_d)$ is a multi-index, $|\alpha| := \sum_i \alpha_i$, and $\partial^\alpha := (\frac{\partial}{\partial x_1})^{\alpha_1}\dots(\frac{\partial}{\partial x_d})^{\alpha_d}$. That is, the sum is over all partial derivatives of order m. We later also need to use the Sobolev m-norm, defined by $\|v\|_m := \sum_{i=0}^m |v|_i$. For $m = 0$, $|v|_m = \|v\|_m = \|v\|$, so we have $\|u - \tilde{u}\| \leqslant C\, h^{k+1}|u|_{k+1}$.

The overall level of inaccuracy in approximating u with $\tilde{\tilde{u}}$ (and hence computing $\langle r,u\rangle$ from $\tilde{\tilde{u}}$) can be bounded using the triangle inequality:

$$\|u - \tilde{\tilde{u}}\| \leqslant \|u - \tilde{u}\| + \|\tilde{u} - \tilde{\tilde{u}}\|.$$

To achieve a final error of $\epsilon\|r\|$ in computing $\langle r,u\rangle$ it is sufficient to achieve $\|u - \tilde{u}\| \leqslant \epsilon/2$, $\|\tilde{u} - \tilde{\tilde{u}}\| \leqslant \epsilon/2$. Thus, by (3), we can take a mesh such that

$$h = O\left(\left(\frac{\epsilon}{|u|_{k+1}}\right)^{1/(k+1)}\right). \tag{4}$$

Observe that $|u|_{k+1}$ might be initially unknown. In the case of the simple instance of the FEM discussed in the previous section, we had $|u|_{k+1} = \|f\|$, so this bound could be explicitly calculated. However, it can be nontrivial to estimate this quantity for more complicated BVPs.

C. Classical complexity of the FEM

The overall complexity of solving a BVP via the FEM is governed by the dimensionality of the problem being solved, the choice of finite element basis, and the desired accuracy criteria. These feed into the complexity of solving the required system of linear equations.

As the matrix M is a Gramian matrix it is necessarily positive semidefinite. Also, the basis ϕ_i is almost universally chosen such that each basis vector only has support on a small number of finite elements, with the implication that M is sparse, i.e., has $s = O(1)$ nonzero entries in each row. The most common choice of algorithm for inversion of matrices of this type (large, sparse, symmetric, and positive semidefinite) is the *conjugate gradient method* [16] (for discussion in the context of the FEM, see [7], Sec. 1.3). This method uses time $O(Ns\sqrt{\kappa}\log 1/\epsilon_{\rm CG})$ to solve a system $M\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$ of N linear equations, each containing at most s terms, with condition number $\kappa = \|M\|\|M^{-1}\|$, up to accuracy $\epsilon_{\rm CG}$ in the "energy norm" $\|\mathbf{x}\|_M := \sqrt{\mathbf{x}^T M\mathbf{x}}$.

We now estimate the values of each of the parameters in this complexity, first calculating the required size of the linear system, N. Let $\mathcal{P}$ be a basis for the space of polynomials of total degree k in d variables. To construct a basis for the space of piecewise degree-k polynomials on the mesh, it is sufficient, for each finite element in the mesh, to include functions defined to be equal to a corresponding function in $\mathcal{P}$ on that finite element and 0 elsewhere. Then the total size of the basis is $N = O(h^{-d})$. Using (4), to achieve a final discretization error of $\epsilon/2$ we can take

$$N = O\left(\left(\frac{|u|_{k+1}}{\epsilon}\right)^{\frac{d}{k+1}}\right).$$

We next determine the required accuracy $\epsilon_{\rm CG}$. Let a be the inner product defining M, such that $M_{ij} = a(\phi_i,\phi_j)$. This inner product induces the energy norm (on functions) $\|u\|_E := \sqrt{a(u,u)}$. Use of this norm makes it easy to interpret the error from the conjugate gradient method, as one can readily calculate that

$$\|\tilde{u} - \tilde{\tilde{u}}\|_E = \|\tilde{\mathbf{u}} - \tilde{\tilde{\mathbf{u}}}\|_M. \tag{5}$$

In many important cases, such as elliptic PDEs, one can show that a is *coercive*: there exists a universal constant c such that $a(u,u) \geqslant c\|u\|^2$ for all u (for further discussion on coercivity in PDEs, see [17]). It follows from coercivity of a that $\|\tilde{u} - \tilde{\tilde{u}}\| \leqslant \sqrt{c}\|\tilde{u} - \tilde{\tilde{u}}\|_E$. To achieve $\|\tilde{u} - \tilde{\tilde{u}}\| \leqslant \epsilon/2$ it is therefore sufficient to take $\epsilon_{\rm CG} = O(\epsilon)$.

The scaling of the condition number of M (denoted κ) with the size and shape of the mesh is discussed extensively in [29] and in Chap. 9 of [8]. Assume that $d \geqslant 2$ and that there exists a universal constant C such that the basis functions ϕ_i satisfy

$$C^{-1}h^{d-2}\|v\|_{L^\infty(T)} \leqslant \sum_{\mathrm{supp}(\phi_i)\cap T\neq\emptyset} v_i^2 \leqslant C h^{d-2}\|v\|_{L^\infty(T)} \tag{6}$$

for any function v such that $v = \sum_i v_i\phi_i$ and any finite element T; this fixes the normalization of the basis functions. Then, for a wide range of relatively regular meshes, the largest eigenvalue $\lambda_{\max}(M) = O(1)$ and the smallest eigenvalue $\lambda_{\min}(M) = \Omega(N^{-2/d})$, so $\kappa = O(N^{2/d})$. Finally, we have $s = O(1)$ by our assumption about the supports of the basis elements ϕ_i. The overall complexity of the algorithm is thus

$$O\left(\left(\frac{|u|_{k+1}}{\epsilon}\right)^{\frac{d+1}{k+1}}\log 1/\epsilon\right).$$

In many practical cases, however, *preconditioning* is applied in order to reduce this scaling by improving the condition number. This can be seen as replacing the matrix M with a matrix $M' = PM$ for some "preconditioner" P and solving the new system of linear equations $M'\tilde{\mathbf{u}} = P\tilde{\mathbf{f}}$. A number of preconditioners are known; one frequently used example in the case of the FEM is the sparse approximate inverse (SPAI) preconditioner. Although there is no guarantee that this preconditioner can improve the condition number in the worst case, experimental results suggest that it can be very effective in practice [18–21]. If the condition number were reduced to the best possible scaling $O(1)$, we would obtain a "best-case" run time of the classical algorithm which is

$$O\left(\left(\frac{|u|_{k+1}}{\epsilon}\right)^{\frac{d}{k+1}}\log 1/\epsilon\right).$$

We remark that the preconditioned matrix M' may no longer be symmetric; the dependence of the conjugate gradient method on the condition number κ is quadratically worse for nonsymmetric matrices, but as we have assumed that $\kappa = O(1)$ following preconditioning, this does not affect the complexity.

The best classical run time following this approach is then found by optimizing over allowed values of k. Observe that in either case, if $|u|_{k+1}$ and d are fixed, this complexity is bounded by a polynomial in $1/\epsilon$.

III. SOLVING THE FEM WITH A QUANTUM ALGORITHM

The key step towards solving the FEM more quickly using a quantum computer is to replace the classical algorithm for solving the corresponding system of linear equations with a quantum algorithm. The fastest such algorithm known was recently presented by Childs, Kothari, and Somma [3], improving previous algorithms of HHL [1] and Ambainis [2].

Theorem 1: Childs, Kothari, and Somma [3]. Let A be an $N \times N$ Hermitian matrix such that $\|A\|\|A^{-1}\| \leqslant \kappa$ and A has at most s nonzero entries in each row. Assume there is an algorithm $\mathcal{P}_A$ which, on input (r,i), outputs the location and value of the ith nonzero entry in row r. Let $\mathbf{b}$ be an N-dimensional unit vector, and assume that there is an algorithm $\mathcal{P}_b$ which produces the corresponding state $|b\rangle$. Let

$$\mathbf{x}' = A^{-1}\mathbf{b}, \quad |x\rangle = \frac{\mathbf{x}'}{\|\mathbf{x}'\|}.$$

Then there is a quantum algorithm which produces state $|x\rangle$ up to accuracy ϵ in the ℓ_2 norm, with a bounded probability of failure, and makes

$$O(s\kappa \operatorname{poly}(\log(s\kappa/\epsilon)))$$

uses of $\mathcal{P}_A$ and $\mathcal{P}_b$. The run time is the same up to a poly(log N) factor.

We also need to approximate the Euclidean norm of the solution, $\|\mathbf{x}'\|$. The most efficient approach known to achieving this appears to be based on the original HHL algorithm. The number of uses of $\mathcal{P}_A$ required to estimate $\|\mathbf{x}'\|$ up to accuracy $\epsilon\|\mathbf{x}'\|$ can be shown to be

$$O((s\kappa^2/\epsilon) \operatorname{poly}\log(s\kappa/\epsilon));$$

the number of uses of $\mathcal{P}_b$ required is $O(\kappa/\epsilon)$. Assuming that $\mathcal{P}_A$ and $\mathcal{P}_b$ can each be implemented in time poly(log N), the run time of the algorithm is

$$O((s\kappa^2/\epsilon) \operatorname{poly}\log(Ns\kappa/\epsilon)).$$

As we were unable to find statements of these bounds in the literature, we sketch the argument behind them in Appendix A.

Here we apply these results to the linear system $M\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$. We see from the above bounds that the complexity of the overall quantum algorithm for solving the FEM is determined by the following parameters:

(1) The complexities of the algorithms $\mathcal{P}_M$ and $\mathcal{P}_{\tilde{f}}$, which, respectively, determine elements of M and (approximately) produce $|\tilde{f}\rangle$.

(2) The condition number κ and sparsity s of the matrix M.

(3) The complexity of determining some quantity of interest given a state which approximates $|\tilde{u}\rangle$.

These quantities will depend in turn on the desired accuracy of the output. We now investigate each of them.

Note that most of the algorithms we use will have some arbitrarily small, but nonzero, probability of failure. We assume throughout that failure probabilities have been made sufficiently low that they can be disregarded.

A. Preparing the input

The purpose of this section is to discuss the time to prepare the input state $|\tilde{f}\rangle$ (i.e., the complexity of the subroutine $\mathcal{P}_{\tilde{f}}$ required for the QLE algorithm). To achieve an efficient algorithm overall, we would like to be able to prepare $|\tilde{f}\rangle$ in time poly(log N). Rather than rely on a quantum RAM to provide $|\tilde{f}\rangle$, we instead refer to a scheme introduced by Zalka [22] and independently rediscovered both by Grover and Rudolph [23] and by Kaye and Mosca [24].

The scheme can be used to produce a real quantum state $|\psi\rangle$ of n qubits in time polynomial in n, given the ability to compute the weights

$$W_x := \sum_{y\in\{0,1\}^{n-k}} |\langle xy|\psi\rangle|^2$$

for arbitrary $k = 1,\dots,n$ and arbitrary $x \in \{0,1\}^k$ in time poly(n), as well as the ability to determine the sign of $\langle x|\psi\rangle$ for arbitrary x in time poly(n).

To approximately produce $|\psi\rangle$ up to a high level of accuracy [e.g., $O(2^{-n})$] in time polynomial in n, it is actually sufficient to be able to approximately compute each weight W_x up to accuracy ϵ in time $O(\log 1/\epsilon)$, for arbitrary ϵ. We sketch the argument as follows. The algorithm of [23], [24] and [22] is designed to produce a state $|\psi'\rangle$ with non-negative amplitudes in the computational basis, such that $\langle x|\psi'\rangle = |\langle x|\psi\rangle| = \sqrt{W_x}$ for all $x \in \{0,1\}^n$, and then flips the signs of amplitudes as required. To produce $|\psi'\rangle$ the algorithm expresses W_x, for each $x \in \{0,1\}^n$, as a telescoping product,

$$W_x = W_{x_1} \times \frac{W_{x_1x_2}}{W_{x_1}} \times \frac{W_{x_1x_2x_3}}{W_{x_1x_2}} \times \cdots \times \frac{W_x}{W_{x_1\dots x_{n-1}}},$$

computes each fraction in turn (in superposition), and uses this to set $\langle x|\psi'\rangle$. If the goal is to produce $|\psi\rangle$ up to accuracy ϵ in the ℓ_2 norm, from the inequality $(|\langle x|\psi\rangle| - |\langle x|\psi'\rangle|)^2 \leqslant |\langle x|\psi\rangle^2 - \langle x|\psi'\rangle^2|$, it is sufficient to approximate each weight W_x, $x \in \{0,1\}^n$, up to additive error $O(\epsilon^2/2^n)$. So the product can be truncated at the point i where the weight $W_{x_1\dots x_i} = O(\epsilon^2/2^n)$, because any subsequent multiplications can only decrease W_x, and weights below this size can be ignored.

If the algorithm does not compute the weights W_x and W_y in some fraction W_x/W_y exactly but, instead, computes the approximations $\widetilde{W_x}$ and $\widetilde{W_y}$ such that $|\widetilde{W_x} - W_x| \leqslant \gamma W_x$ and $|\widetilde{W_y} - W_y| \leqslant \gamma W_y$ for some γ, then $|\widetilde{W_x}/\widetilde{W_y} - W_x/W_y| = O(\gamma W_x/W_y)$. As we have assumed that $W_x = \Omega(\epsilon^2/2^n)$ for all k-bit strings x for which we compute W_x ($1 \leqslant k \leqslant n$), it is sufficient to approximate each weight W_x up to additive accuracy $O(\epsilon^2/(n2^n))$ for each fraction to be accurate up to a multiplicative error of $O(\epsilon^2/(n2^n))$ and hence the overall product of weights to be accurate up to an additive error of $O(\epsilon^2/2^n)$. From the assumption about the complexity of the algorithm for approximately computing W_x, we can achieve this level of accuracy in poly(n, log $1/\epsilon$) time.

In the case of the FEM, the weights W_x correspond to quantities of the form

$$S(a,b) := \sum_{i=a}^{b} \left(\int_\Omega \phi_i(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \right)^2,$$

where $\mathbf{x} \in \mathbb{R}^d$ and a and b are integers. Expressions of this form can be computed (either exactly or approximately) for many functions f of interest. For example, consider the one-dimensional setting discussed in Sec. II. If f is a polynomial, then the integral can be easily calculated and corresponds to a polynomial in x_{i-1}, x_i, and x_{i+1}. If the finite elements are regularly spaced, so $x_i = ih$ for some h, the entire sum $S(a,b)$ is a polynomial in a and b which can be explicitly calculated for any a and b.

For a choice of polynomial basis of degree k [i.e., where the $(k+1)^{th}$ derivative $\phi_i^{(k+1)} = 0$], then from Darboux's formula one has that

$$\int_{\Omega} \phi_i(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = \sum_{j=1}^{k} (-1)^j \phi_i^{(j)}(\mathbf{x}) \underbrace{\int \cdots \int}_{j+1 \text{ times}} f(\mathbf{x}) d\mathbf{x}.$$

So, once the basis is specified, computing individual amplitudes is only as difficult as integrating the function $f(\mathbf{x})$. However, the state-production algorithm requires the computation of weights which depend on up to $N = 2^n$ squared amplitudes. To obtain an efficient algorithm, it is therefore necessary to find a more concise expression for these sums.

As discussed above, this can be achieved when f is a polynomial and the finite element mesh is suitably regular. This includes some physically interesting cases; even a constant function f can be of interest. An efficiently computable expression for $S(a,b)$ can also be obtained when f is only supported on a few basis elements. However, it appears challenging to compute this quantity efficiently for more general functions f. Indeed, see Sec. IV C for an argument that this should not be achievable in general.

For simplicity in the subsequent bounds, we henceforth assume that the state $|\tilde{f}\rangle$ can be produced perfectly in time poly($\log N$).

B. Solving the system of linear equations

Let M be the matrix defined by $M_{ij} = a(\phi_i, \phi_j)$. Recall from Theorem III that the quantum algorithm assumes that it has access to an algorithm $\mathcal{P}_M$ which, on input (r,i), outputs the location and value of the ith nonzero entry in row r (or "not found" if there are fewer than i nonzero entries). If the finite element mesh is suitably regular, $\mathcal{P}_M$ is easy to implement. For instance, consider the set of n piecewise linear functions on $[0,1]$ defined in Sec. II. Then M is a tridiagonal matrix whose diagonal elements are equal to $2/h$ and whose off-diagonal elements are equal to $-1/h$. Hence for $r > 1$,

$$\mathcal{P}_M(r,i) = \begin{cases} (r-1, -1/h) & \text{if} \quad i = 1, \\ (r, 2/h) & \text{if} \quad i = 2, \\ (r+1, -1/h) & \text{if} \quad i = 3, \\ \text{not found} & \text{otherwise.} \end{cases}$$

More generally, it will be possible to implement $\mathcal{P}_M$ efficiently if there is an efficient procedure for mapping an index to a finite element and for listing the neighboring elements for a given element. This will be the case, for example, when the finite element mesh is a regular triangulation of a polygon. For discussion of automated mesh generation and indexing schemes, see both [25] and Sec. 5.1 in [7].

When solving the system of linear equations, inaccuracies in the prepared state $|\tilde{f}\rangle$ will translate into inaccuracies in the output state $|\tilde{u}\rangle$. Let $|\tilde{\tilde{f}}\rangle$ be the approximate state that was actually prepared. Then the state produced after applying the QLE algorithm is (approximately)

$$\frac{M^{-1}|\tilde{\tilde{f}}\rangle}{\| M^{-1}|\tilde{\tilde{f}}\rangle \|}.$$

If $|\tilde{f}\rangle$ is prepared up to accuracy ϵ in the ℓ_2 norm, then the inaccuracy of the output state in the ℓ_2 norm is

$$\left\| \frac{M^{-1}|\tilde{f}\rangle}{\| M^{-1}|\tilde{f}\rangle \|} - \frac{M^{-1}|\tilde{\tilde{f}}\rangle}{\| M^{-1}|\tilde{\tilde{f}}\rangle \|} \right\|.$$

Writing $|\tilde{\tilde{f}}\rangle = |\tilde{f}\rangle + |\epsilon\rangle$ for some vector $|\epsilon\rangle$ such that $\| |\epsilon\rangle \| = \epsilon$, this quantity is equal to

$$\begin{aligned} &\left\| \frac{M^{-1}|\tilde{f}\rangle(\| M^{-1}|\tilde{\tilde{f}}\rangle \| - \| M^{-1}|\tilde{f}\rangle \|)}{\| M^{-1}|\tilde{f}\rangle \| \| M^{-1}|\tilde{\tilde{f}}\rangle \|} - \frac{M^{-1}|\epsilon\rangle}{\| M^{-1}|\tilde{\tilde{f}}\rangle \|} \right\| \\ &\leqslant \frac{|\| M^{-1}|\tilde{\tilde{f}}\rangle \| - \| M^{-1}|\tilde{f}\rangle \||}{\| M^{-1}|\tilde{\tilde{f}}\rangle \|} + \frac{\| M^{-1}|\epsilon\rangle \|}{\| M^{-1}|\tilde{\tilde{f}}\rangle \|} \\ &\leqslant 2 \frac{\| M^{-1}|\epsilon\rangle \|}{\| M^{-1}|\tilde{\tilde{f}}\rangle \|} \leqslant 2\epsilon\kappa, \end{aligned}$$

by the triangle inequality, the reverse triangle inequality, and the definition of the condition number κ. We therefore see that, if preconditioning is not applied to the matrix M to reduce κ, it is necessary for $|\tilde{f}\rangle$ to be prepared up to accuracy of $O(N^{-2/d}\epsilon)$. (Note that this is not an issue if we can produce $|\tilde{f}\rangle$ exactly, as for some examples discussed in the previous section.)

Clader, Jacobs, and Sprouse [11] showed that the SPAI preconditioner can be used within the overall framework of the QLE algorithm. Preconditioning replaces M with $M' = PM$ for some matrix P, to obtain the corresponding linear system $PM\tilde{\mathbf{u}} = P\tilde{\mathbf{f}}$. In the SPAI preconditioner, P is chosen such that $P \approx M^{-1}$ and also that P is sparse. The sparsity desired is a parameter of the algorithm; although one has no guarantees that either P or PM will be sparse while PM achieves a low condition number, in practice this is often the case. The structure of the SPAI is designed such that queries to entries of PM can be computed from queries to M with a modest overhead [11].

If preconditioning is used, we no longer need to prepare the initial state $|\tilde{f}\rangle$, but a state proportional to $P|\tilde{f}\rangle$. Note that preparing the input $P\tilde{\mathbf{f}}$ in the classical case requires only multiplication of a vector by a sparse matrix, which is computationally cheap compared to matrix inversion. As such, it is typically neglected when considering the classical computational complexity. However, the situation is more complicated in the quantum setting.

The most straightforward way to prepare $P|\tilde{f}\rangle$ is to construct $|\tilde{f}\rangle$ and then attempt to apply the (nonunitary) operation P. There are several known approaches which can be used to achieve this probabilistically. One elegant example is a simple special case of the "Chebyshev" approach in Sec. 4 of [3]. That work uses a quantum walk to apply nth-order Chebyshev polynomials $T_n(P)$ in an arbitrary s-sparse Hermitian matrix P. (If P is not Hermitian, a standard trick [1] can be used to express it as a submatrix of a Hermitian matrix.) As the first Chebyshev polynomial T_1 is simply $T_1(x) = x$, this allows P itself to be implemented. If the subroutine in [3] succeeds when applied to a state $|\psi\rangle$, then $|\psi\rangle$ is (exactly) mapped to $P|\psi\rangle/\|P|\psi\rangle\|$. The success probability is at least

$$\frac{\|P|\psi\rangle\|^2}{s^2\|P\|_{\max}^2} \geqslant \frac{1}{\kappa(P)^2 s^2},$$

where $\|P\|_{\max} = \max_{i,j}|P_{ij}|$ and we use $\|P\|_{\max} \leqslant \|P\|$. Using amplitude amplification, the failure probability can be made at most δ, for arbitrarily small $\delta > 0$, with $O(1/(\kappa(P)s))$ repetitions. Each repetition requires time polylogarithmic in N, $\kappa(P)$, and s.

Combining all these considerations, we see that if preconditioning is used, the complexity of the quantum algorithm will depend on a number of parameters, each of which may be hard to estimate in advance. These are the condition number of PM, the complexity of computing entries of P, the sparsity of P, and the condition number of P. Here, in order to give a best-case comparison of the preconditioned quantum algorithm with the classical algorithm, we make the optimistic assumption that preconditioning is optimal [i.e., $\kappa(PM) = O(1)$] and that taking all of these additional sources of complexity into account multiplies the run time by only a poly($\log(N)$) factor.

C. Measuring the output

By running the QLE algorithm, we obtain an output state $|\widetilde{\tilde{u}}\rangle$ which approximates the normalized state

$$|\tilde{u}\rangle = \frac{\sum_i \tilde{\mathbf{u}}_i|i\rangle}{\sqrt{\sum_i \tilde{\mathbf{u}}_i^2}},$$

where we associate each basis state $|i\rangle$ with the basis function ϕ_i. Given copies of $|\tilde{u}\rangle$, we can carry out measurements to extract information about u. One example is the prototypical problem we consider here, approximating the L^2 inner product $\langle u,r\rangle$ between u and a fixed function r. This can be achieved by approximately computing the inner product $\langle\tilde{u}|r\rangle$ between $|\tilde{u}\rangle$ and the state $|r\rangle$ defined by

$$|r\rangle = \frac{1}{\left(\sum_i\langle\phi_i,r\rangle^2\right)^{1/2}}\sum_i\langle\phi_i,r\rangle|i\rangle \qquad (7)$$

for some function r; then $\langle\tilde{u}|r\rangle$ is the L^2 inner product between $\tilde{u}$ and r, up to an overall scaling factor. $|r\rangle$ can be produced using techniques described in the previous section. Some interesting cases are particularly simple: for example, taking r to be uniform in a region, $\langle\tilde{u}|r\rangle$ gives the average of $\tilde{u}$ over that region.

This inner product can be estimated using a procedure known as the Hadamard test [26], a subroutine whose output is a ±1-valued random variable with expectation $\langle\tilde{u}|r\rangle$. By applying amplitude estimation [27] to approximately compute this expectation, $\langle\tilde{u}|r\rangle$ can be estimated up to accuracy ϵ with $O(1/\epsilon)$ uses of algorithms to produce the states $|\tilde{u}\rangle$ and $|r\rangle$. A related approach was used by Clader, Jacobs, and Sprouse [11] to compute an electromagnetic scattering cross section, which corresponds to a quantity of the form $|\langle\tilde{u}|r\rangle|^2$. This can be approximately computed using the swap test [28], a subroutine which, given two states $|\psi\rangle$, $|\psi'\rangle$, outputs "same" with probability $\frac{1}{2}+\frac{1}{2}|\langle\psi|\psi'\rangle|^2$ and "different" otherwise.

We remark that more complicated properties of u seem to be more problematic to compute directly from the state $|\tilde{u}\rangle$, due to the nonorthogonality of the basis $\{\phi_i\}$. For example, one common use of the QLE algorithm is to determine the similarity of solutions to sets of linear equations by using the swap or Hadamard test to compare them [1,2]. Consider two states $|a\rangle$ and $|b\rangle$ corresponding to functions $a = \sum_i \mathbf{a}_i\phi_i$ and $b = \sum_i \mathbf{b}_i\phi_i$. Then

$$\langle a|b\rangle \propto \sum_i \mathbf{a}_i\mathbf{b}_i,$$

while a sensible measure of similarity of the functions a and b is the inner product

$$\int_\Omega a(\mathbf{x})b(\mathbf{x})d\mathbf{x} = \sum_{i,j}\mathbf{a}_i\mathbf{b}_j\int_\Omega \phi_i(\mathbf{x})\phi_j(\mathbf{x})d\mathbf{x}.$$

One would hope for this to be approximately proportional to $\sum_i \mathbf{a}_i\mathbf{b}_i$. However, although ϕ_i and ϕ_j do not have overlapping support for most pairs $i \neq j$, there are still enough such pairs where this overlap is nonzero that the integral can sometimes be a poor approximation.

D. Overall complexity

The total complexity of the quantum algorithm for solving an FEM problem is found by combining the complexities of all of the above pieces. Assume that we would like to compute $R := \int_\Omega r(\mathbf{x})u(\mathbf{x})d\mathbf{x}$ for some $r : \Omega \to \mathbb{R}$ up to additive error $\epsilon\|r\|$. Write $\alpha = (\sum_i\langle\phi_i,r\rangle^2)^{1/2}$. The quantum algorithm will perform the following steps by applying the QLE algorithm to the system of linear equations $M\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$:

Step 1. Estimate $\|\tilde{\mathbf{u}}\|$ up to an additive term ϵ_N. Let $\widetilde{N}$ be the estimate.

Step 2. Use the QLE algorithm to produce copies of $|\widetilde{\tilde{u}}\rangle$, an approximation to $|\tilde{u}\rangle$. Use these to estimate $\langle r|\widetilde{\tilde{u}}\rangle$ up to an additive term ϵ_{out}. Let $\widetilde{R}$ be the estimate.

Step 3. Output $\alpha\widetilde{N}\widetilde{R}$ as an estimate of R.

We can bound the overall error as follows. Let ϵ_L be the inaccuracy, in the ℓ_2 norm, in solving the system of linear equations in step 2, i.e., $\epsilon_L = \||\widetilde{\tilde{u}}\rangle - |\tilde{u}\rangle\|$. This encompasses any errors in producing the initial state $|\tilde{f}\rangle$, as well as inaccuracies arising from the QLE algorithm itself (although recall that we have in fact assumed that we can produce $|\tilde{f}\rangle$ perfectly). Then

$$\begin{aligned}\widetilde{R} &= \langle r|\widetilde{\tilde{u}}\rangle + \epsilon_{\text{out}} = \langle r|\tilde{u}\rangle + \langle r|(|\widetilde{\tilde{u}}\rangle - |\tilde{u}\rangle) + \epsilon_{\text{out}} \\ &= \langle r|\tilde{u}\rangle + \epsilon_L' + \epsilon_{\text{out}}\end{aligned}$$

for some ϵ_L', where $|\epsilon_L'| \leqslant \epsilon_L$ by Cauchy-Schwarz. So

$$\widetilde{R} = \frac{\sum_i \tilde{\mathbf{u}}_i\langle\phi_i,r\rangle}{\|\tilde{\mathbf{u}}\|\left(\sum_i\langle\phi_i,r\rangle^2\right)^{1/2}} + \epsilon_L' + \epsilon_{\text{out}} = \frac{\langle\tilde{u},r\rangle}{\alpha\|\tilde{\mathbf{u}}\|} + \epsilon_L' + \epsilon_{\text{out}}$$

using the definition of $|r\rangle$ from (7) and of $|\tilde{u}\rangle$ as a normalized version of $\tilde{\mathbf{u}}$. Writing $\widetilde{N} = \|\tilde{\mathbf{u}}\| + \epsilon_N$, we have

$$\alpha \widetilde{N}\widetilde{R} = \langle \tilde{u},r\rangle\left(1 + \frac{\epsilon_N}{\|\tilde{\mathbf{u}}\|}\right) + \alpha(\|\tilde{\mathbf{u}}\| + \epsilon_N)(\epsilon'_L + \epsilon_{\text{out}}).$$

The analysis of the remaining term $\langle \tilde{u},r\rangle$ is now similar to the classical setting,

$$\langle \tilde{u},r\rangle = \langle u,r\rangle + \langle \tilde{u} - u,r\rangle = \langle u,r\rangle + \epsilon'_D,$$

where

$$|\epsilon'_D| \leqslant \|r\|\|\tilde{u} - u\| =: \|r\|\epsilon_D$$

by Cauchy-Schwarz. Combining all the terms together, we have

$$\alpha\widetilde{N}\widetilde{R} - R = \epsilon'_D\left(1 + \frac{\epsilon_N}{\|\tilde{\mathbf{u}}\|}\right) + \frac{\langle u,r\rangle\epsilon_N}{\|\tilde{\mathbf{u}}\|} + \alpha(\|\tilde{\mathbf{u}}\| + \epsilon_N)(\epsilon'_L + \epsilon_{\text{out}}).$$

We, finally, use $\langle u,r\rangle \leqslant \|u\|\|r\|$ to obtain

$$\alpha\widetilde{N}\widetilde{R} - R = \epsilon'_D\left(1 + \frac{\epsilon_N}{\|\tilde{\mathbf{u}}\|}\right) + \frac{\|u\|\|r\|\epsilon'_N}{\|\tilde{\mathbf{u}}\|} + \alpha(\|\tilde{\mathbf{u}}\| + \epsilon_N)(\epsilon'_L + \epsilon_{\text{out}})$$

for some ϵ'_N such that $|\epsilon'_N| \leqslant |\epsilon_N|$. To achieve overall accuracy $\epsilon\|r\|$ it is sufficient for each term in this expression to be upper-bounded by $\epsilon\|r\|/3$, which follows from

$$\begin{aligned}\epsilon_D &= O(\epsilon),\\ \epsilon_N &= O(\min\{\|\tilde{\mathbf{u}}\|,\epsilon\|\tilde{\mathbf{u}}\|/\|u\|\}),\\ \epsilon_L,\epsilon_{\text{out}} &= O(\epsilon\|r\|/(\alpha\|\tilde{\mathbf{u}}\|)).\end{aligned}$$

Assume, for simplicity in the final bound, that $\epsilon \leqslant \|u\|$; then the second condition becomes $\epsilon_N = O(\epsilon\|\tilde{\mathbf{u}}\|/\|u\|)$. We now calculate the complexity of achieving these accuracies.

Using the discretization error bound, (3), we have $\epsilon_D \leqslant C\,h^{k+1}|u|_{k+1}$ for some universal constant C. We can therefore take

$$h = O\left(\left(\frac{\epsilon}{|u|_{k+1}}\right)^{\frac{1}{k+1}}\right).$$

As in the classical case, this choice of h corresponds to solving a system of

$$N = O\left(\left(\frac{|u|_{k+1}}{\epsilon}\right)^{\frac{d}{k+1}}\right)$$

linear equations. For any $\delta > 0$, as discussed in Sec. III and Appendix A, $\|\tilde{\mathbf{u}}\|$ can be approximated up to accuracy $\delta\|\tilde{\mathbf{u}}\|$ in time $O((s\kappa^2/\delta)\,\text{poly}\log(Ns\kappa/\delta))$ using the HHL algorithm, recalling that s and κ are the sparsity and condition number of M, respectively. Inserting $\delta = \epsilon/\|u\|$ and the bound on N, this part requires time

$$O\left(\frac{s\kappa^2\|u\|}{\epsilon}\text{poly}(\log(s\kappa\|u\||u|_{k+1}/\epsilon))\right).$$

We can also put an upper bound on ϵ_L and ϵ_{out} by upper-bounding $\alpha/\|r\|$ and $\|\tilde{\mathbf{u}}\|$. In the former case, it holds that

$$\frac{\alpha}{\|r\|} = O(h\sqrt{s});$$

we prove this technical claim in Appendix B. In the latter case,

$$\|\tilde{\mathbf{u}}\| = O(\|\tilde{\mathbf{u}}\|_M/h) = O(\|\tilde{u}\|_E/h) = O(\|u\|_1/h).$$

The first two equalities follow from $\lambda_{\min}(M) = \Omega(N^{-2/d}) = \Omega(h^2)$ [8,29] and the equivalence between $\|\tilde{\mathbf{u}}\|_M$ and $\|\tilde{u}\|_E$ [see (5)]. The third follows from $\|\tilde{u}\|_E = O(\|\tilde{u}\|_1)$, where $\|\cdot\|_1$ is the Sobolev ℓ-norm. This is a consequence of the inner product $a(\cdot,\cdot)$, which defines the energy norm corresponding to an underlying elliptic PDE [8], and the bound (3). Combining these bounds, the requirement on ϵ_L and ϵ_{out} can be rewritten as

$$\epsilon_L,\epsilon_{\text{out}} = O\left(\frac{\epsilon}{\sqrt{s}\|u\|_1}\right).$$

To achieve accuracy ϵ_{out} using the Hadamard test and amplitude estimation requires $O(1/\epsilon_{\text{out}})$ uses of the QLE algorithm, each of which runs in time $O(s\kappa\,\text{poly}(\log(Ns\kappa/\epsilon_L)))$ by Theorem III. Inserting the bounds on ϵ_L, ϵ_{out} gives a complexity for Step 2 which is

$$O\left(\frac{\sqrt{s}\kappa\|u\|_1}{\epsilon}\text{poly}(\log(s\kappa\|u\|_1|u|_{k+1}/\epsilon))\right).$$

Combining these bounds, we obtain an overall run time of

$$O\left(\frac{s\kappa^2\|u\| + \sqrt{s}\kappa\|u\|_1}{\epsilon}\text{poly}(\log(s\kappa\|u\|_1|u|_{k+1}/\epsilon))\right).$$

In fixed spatial dimension, $s = O(1)$, and if preconditioning is not used, $\kappa = O(N^{2/d}) = O((|u|_{k+1}/\epsilon)^{2/(k+1)})$. Inserting these values, we obtain a bound of

$$\widetilde{O}\left(\frac{\|u\||u|_{k+1}^{\frac{4}{k+1}}}{\epsilon^{\frac{k+5}{k+1}}} + \frac{\|u\|_1|u|_{k+1}^{\frac{2}{k+1}}}{\epsilon^{\frac{k+3}{k+1}}}\right),$$

where the $\widetilde{O}$ notation hides polylogarithmic factors. On the other hand, if we assume that optimal preconditioning has been applied, so κ reduces to $O(1)$, we would obtain an overall bound of just

$$\widetilde{O}\left(\frac{\|u\|_1}{\epsilon}\right).$$

These run times should be compared with the corresponding run times of the classical algorithm:

$$\widetilde{O}\left(\left(\frac{|u|_{k+1}}{\epsilon}\right)^{\frac{d+1}{k+1}}\right) \quad \text{and} \quad \widetilde{O}\left(\left(\frac{|u|_{k+1}}{\epsilon}\right)^{\frac{d}{k+1}}\right),$$

respectively. First, note that if preconditioning is used, the dependence on $|u|_{k+1}$ in the quantum algorithm's run time is significantly milder than for the classical algorithm, being only polylogarithmic. Even if preconditioning is not used, for large enough d the dependence is polynomially better.

Perhaps more importantly, observe that in the term dependent on ϵ in the algorithms' run times, the quantum algorithm's run time no longer depends on the dimension d. This holds whether or not preconditioning is used. Thus the quantum algorithm will achieve a large speedup when ϵ is small and d is large. [Note that we cannot quite call this an exponential quantum speedup with respect to d: the run time of the quantum algorithm also depends on a term C in (3) which is constant for a fixed dimension and family of meshes but has unbounded dependence on d.]

As a demonstrative example of the speedup (or otherwise) expected for the quantum algorithm, consider the case of solving a BVP in four dimensions (three spatial and one temporal, say), using piecewise linear basis functions. Then the classical run times both without preconditioning and with optimal preconditioning are

$$\widetilde{O}\left(\left(\frac{|u|_2}{\epsilon}\right)^{\frac{5}{2}}\right) \quad \text{and} \quad \widetilde{O}\left(\left(\frac{|u|_2}{\epsilon}\right)^{2}\right),$$

respectively. The analogous quantum run times are

$$\widetilde{O}\left(\frac{\|u\||u|_2^2}{\epsilon^3} + \frac{\|u\|_1|u|_2}{\epsilon^2}\right) \quad \text{and} \quad \widetilde{O}\left(\frac{\|u\|_1}{\epsilon}\right).$$

In this case, lack of preconditioning leads to a quantum algorithm which might or might not outperform the classical algorithm, depending on the relative sizes of ϵ, $\|u\|$, $\|u\|_1$, and $|u|_2$. In the optimally preconditioned case, the quantum algorithm both scales better with accuracy and has a less stringent condition on the solution smoothness.

IV. QUANTUM LOWER BOUNDS

We have seen that the QLE algorithm can be used to obtain polynomial quantum speedups over the best-known classical algorithms for the FEM. We now argue that, in the physically realistic setting of fixed dimension and smooth solutions, a polynomial quantum speedup is the largest speedup one can expect. We first discuss the general question of putting lower bounds on the complexity of algorithms based on a QLE subroutine.

A. A general quantum lower bound

We observe from Theorem III and the discussion in Sec. III C that producing the quantum state $|x\rangle \propto A^{-1}|b\rangle$ for some well-conditioned, sparse matrix A can be achieved in time $\operatorname{poly}\log(1/\epsilon)$, while the apparently simpler task of approximating some natural properties of $|x\rangle$ uses time $O(1/\epsilon)$. It is therefore natural to suspect that the run time of this component could be improved substantially, e.g., to $\operatorname{poly}\log(1/\epsilon)$. However, it was shown by HHL [1] that the existence of a quantum algorithm with this scaling for approximating some very simple properties of $|x\rangle$ would imply the complexity-theoretic consequence BQP = PP, which is considered highly unlikely (implying, for example, that quantum computers could efficiently solve NP-complete problems).

As well as this complexity-theoretic argument, we now give an argument based on ideas from query complexity which lower-bounds the run time of any algorithm which approximates some function of the output of the QLE algorithm, without making use of the internal structure of the algorithm. This encompasses all the uses of QLE for the FEM discussed in Sec. III C.

We adapt a standard technique of Bennett *et al.* [30]. Consider an algorithm which has access to a unitary subroutine $\mathcal{A}_\psi$, parametrized by an unknown state $|\psi\rangle$, such that $\mathcal{A}_\psi$ maps $|0\rangle$ to $|\psi\rangle$. The algorithm may also have access to the inverse subroutine $\mathcal{A}_\psi^{-1}$. The algorithm does not know anything about how $\mathcal{A}_\psi$ is implemented and uses it as a "black box." It aims to estimate some property of $|\psi\rangle$. In the context of the FEM, we think of $\mathcal{A}_\psi$ as the QLE algorithm, where $|\psi\rangle$ is the output state corresponding to the approximate solution of the desired BVP. We assume that the algorithm only makes use of the QLE subroutine for one instance, i.e., it uses $\mathcal{A}_\psi$ throughout, rather than $\mathcal{A}_{\psi'}$ for some $|\psi'\rangle \neq |\psi\rangle$; relaxing this assumption would only make the problem harder.

For notational simplicity, we also assume in the proof that the overall algorithm does not use $\mathcal{A}_\psi^{-1}$ and does not use any ancilla qubits. (These assumptions can easily be relaxed without changing the conclusions.) Further assume that the overall algorithm makes T uses of $\mathcal{A}_\psi$, interspersed with arbitrary unitary operators $U_1,\dots,U_{T+1}$. Let $|\phi\rangle$ be such that $\||\psi\rangle - |\phi\rangle\| \leqslant \epsilon$ and such that the output of the algorithm should be different when using $\mathcal{A}_\phi$ rather than $\mathcal{A}_\psi$. Finally, let $|\eta\rangle_{\psi,t}$ be the state of the overall algorithm after t uses of $\mathcal{A}_\psi$. Then

$$\begin{aligned}
&\||\eta\rangle_{\psi,T} - |\eta\rangle_{\phi,T}\| \\
&= \|U_{T+1}\mathcal{A}_\psi U_T \dots \mathcal{A}_\psi U_1|0\rangle - U_{T+1}\mathcal{A}_\phi U_T \dots \mathcal{A}_\phi U_1|0\rangle\| \\
&= \|\mathcal{A}_\psi U_T \dots \mathcal{A}_\psi U_1|0\rangle - \mathcal{A}_\phi U_T \dots \mathcal{A}_\phi U_1|0\rangle\| \\
&\leqslant \|\mathcal{A}_\psi U_T \mathcal{A}_\psi U_{T-1}\mathcal{A}_\psi \dots \mathcal{A}_\psi U_1|0\rangle \\
&\quad - \mathcal{A}_\psi U_T \mathcal{A}_\phi U_{T-1}\mathcal{A}_\phi \dots \mathcal{A}_\phi U_1|0\rangle\| \\
&\quad + \|\mathcal{A}_\psi U_T \mathcal{A}_\phi U_{T-1}\mathcal{A}_\phi \dots \mathcal{A}_\phi U_1|0\rangle - \mathcal{A}_\phi U_T \dots \mathcal{A}_\phi U_1|0\rangle\| \\
&\leqslant \|\mathcal{A}_\psi U_{T-1}\mathcal{A}_\psi \dots \mathcal{A}_\psi U_1|0\rangle - \mathcal{A}_\phi U_{T-1}\mathcal{A}_\phi \dots \mathcal{A}_\phi U_1|0\rangle\| \\
&\quad + \|\mathcal{A}_\psi - \mathcal{A}_\phi\|.
\end{aligned}$$

The first inequality is the triangle inequality, while the second uses the fact that unitaries do not change the Euclidean distance. As the algorithm does not use any information about the internal structure of $\mathcal{A}_\psi$, $\mathcal{A}_\phi$, we are free to assume that $\mathcal{A}_\psi = |\psi\rangle\langle 0| + |\phi'\rangle\langle 1| + \sum_{i\geqslant 2}|\zeta_i\rangle\langle i|$, $\mathcal{A}_\phi = |\phi\rangle\langle 0| + |\psi'\rangle\langle 1| + \sum_{i\geqslant 2}|\zeta_i\rangle\langle i|$. Here $|\phi'\rangle$ and $|\psi'\rangle$ are states orthonormal to $|\psi\rangle$ and $|\phi\rangle$, respectively, within the subspace spanned by $|\psi\rangle$ and $|\phi\rangle$, and $|\zeta_i\rangle$ are arbitrary states which are orthonormal to both of these states and each other. Explicitly, we can take

$$|\phi'\rangle = \frac{|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle}{\sqrt{1-|\langle\psi|\phi\rangle|^2}}, \quad |\psi'\rangle = \frac{|\psi\rangle - \langle\phi|\psi\rangle|\phi\rangle}{\sqrt{1-|\langle\psi|\phi\rangle|^2}}.$$

Then

$$\|\mathcal{A}_\psi - \mathcal{A}_\phi\| = \|(|\psi\rangle - |\phi\rangle)\langle 0| + (|\phi'\rangle - |\psi'\rangle)\langle 1|\|.$$

Writing $|\delta\rangle := |\psi\rangle - |\phi\rangle$, $|\delta'\rangle := |\phi'\rangle - |\psi'\rangle$ and upper-bounding the operator norm by the Frobenius norm, we have

$$\begin{aligned}
\|\mathcal{A}_\psi - \mathcal{A}_\phi\| &\leqslant \sqrt{\operatorname{tr}\left(\mathcal{A}_\psi^\dagger - \mathcal{A}_\phi^\dagger\right)(\mathcal{A}_\psi - \mathcal{A}_\phi)} \\
&= \sqrt{\langle\delta|\delta\rangle + \langle\delta'|\delta'\rangle} = \sqrt{2}\||\psi\rangle - |\phi\rangle\|,
\end{aligned}$$

where we use the fact (which can easily be seen by direct calculation) that $\||\phi'\rangle - |\psi'\rangle\| = \||\psi\rangle - |\phi\rangle\|$. Hence

$$\|\mathcal{A}_\psi - \mathcal{A}_\phi\| \leqslant \sqrt{2}\epsilon$$

and in turn, by induction,

$$\||\eta\rangle_{\psi,T} - |\eta\rangle_{\phi,T}\| \leqslant T\sqrt{2}\epsilon.$$

As the algorithm is supposed to output something different if it is given $\mathcal{A}_\phi$ rather than $\mathcal{A}_\psi$, assuming that it succeeds, the final measurement made distinguishes between the two states $|\eta\rangle_{\psi,T}$ and $|\eta\rangle_{\phi,T}$. The optimal worst-case probability p of distinguishing these states is given by the trace distance between them [31], so

$$p = \frac{1}{2} + \frac{1}{4}\|\eta_{\psi,T} - \eta_{\phi,T}\|_1 \leqslant \frac{1}{2} + \frac{1}{2}\||\eta\rangle_{\psi,T} - |\eta\rangle_{\phi,T}\| \leqslant \frac{1}{2} + \frac{T\epsilon}{\sqrt{2}}.$$

Therefore, in order for the algorithm to succeed with probability (say) 2/3, it must use $\mathcal{A}_\psi$ at least $\Omega(1/\epsilon)$ times. As a simple example of how this bound can be applied, consider an algorithm which attempts to distinguish between these two cases: (a) the output from the QLE subroutine is a particular state $|\psi_0\rangle$; and (b) the output from the QLE subroutine is some state $|\phi\rangle$ such that the overlap $|\langle\phi|\psi_0\rangle|^2 = 1 - \epsilon$. Then $\||\phi\rangle - |\psi_0\rangle\| = O(\sqrt{\epsilon})$, so any algorithm distinguishing between these two cases by using QLE as a black box must use it $\Omega(1/\sqrt{\epsilon})$ times.

This bound is tight for this particular problem, which can be solved by using the QLE subroutine $O(1/\sqrt{\epsilon})$ times within quantum amplitude estimation [27]. However, for other problems it may be possible to put stronger lower bounds on the complexity.

B. Replacing the QLE subroutine with a classical algorithm

The above lower bound shows, roughly speaking, that any algorithm which uses the QLE subroutine as a black box and attempts to determine up to accuracy ϵ some property of the output state must make $\Omega(1/\sqrt{\epsilon})$ uses of the subroutine. However, in some cases it can be of interest to approximate properties of the output state to quite low levels of accuracy.

For example, consider the problem of distinguishing between the following two cases: (a) the solution to a BVP is periodic; and (b) the solution is far from periodic. As it is known that quantum algorithms can test the periodicity of functions exponentially faster than classical algorithms can [32], one might hope to use QLE, together with the quantum periodicity tester, to solve this problem exponentially faster than any classical algorithm.

Also note that it is likely to be hard to prove that it is impossible to obtain a superpolynomial quantum speedup for solving BVPs, if we define "solving" a BVP as computing an arbitrary function of the solution to a BVP. For example, we could contrive a BVP where the solution is easy to write down, and can be interpreted as an integer, and could then ask the algorithm to output the prime factors of that integer. Proving that quantum computers could not outperform classical computers for this task would imply an efficient classical algorithm for integer factorization.

Nevertheless, we believe that, even given a quantum algorithm for solving problems of this form, any uses of the QLE algorithm as a subroutine could be replaced with a classical algorithm, with at most a polynomial slowdown if the spatial dimension is fixed and the solution is suitably smooth. This would imply that any exponential quantum speedup in the overall algorithm is not due to the part of it that solves the FEM. Making this argument rigorous seems challenging for technical reasons related to regularity of meshes and comparing different norms to measure accuracy, so we do not attempt it here, instead merely sketching the ideas informally.

The argument proceeds as follows. Imagine we have an overall quantum algorithm which uses the QLE algorithm as a subroutine to solve T FEM instances in a spatial dimension bounded by $d = O(1)$, such that the solution to each instance has all relevant Sobolev norms bounded by $O(1)$. Then each such instance can be approximately solved by a classical algorithm using a mesh of size poly$(1/\epsilon)$, for any desired accuracy ϵ. We replace each subroutine which applies the QLE algorithm to solve an instance of the FEM, using a mesh $\mathcal{M}$ to achieve accuracy ϵ, with the following procedure:

(1) Classically solve the same FEM instance, using a mesh $\mathcal{M}'$ which achieves accuracy $\max\{\gamma/T,\epsilon\}$ for some universal constant γ. Note that if $\epsilon < \gamma/T$, this will in general be a coarser mesh than $\mathcal{M}$.

(2) Construct the quantum state corresponding to the output of the solver, as a superposition of basis functions from $\mathcal{M}'$.

(3) Map this quantum state to the equivalent quantum state on the finer mesh $\mathcal{M}$. This is essentially equivalent to the classical task of expressing each element of $\mathcal{M}'$ in terms of elements of $\mathcal{M}$.

Here we are assuming that the meshes $\mathcal{M}$ and $\mathcal{M}'$ are sufficiently regular that the last step makes sense (in particular, that $\mathcal{M}$ is a submesh of $\mathcal{M}'$).

If $\epsilon \geqslant \gamma/T$, the state produced by the original subroutine is left essentially unchanged. If $\epsilon < \gamma/T$, the original state produced was within distance $O(1/T)$ of the actual solution to the corresponding FEM instance, as is the state produced by the new subroutine. By the triangle inequality, the new state must be within distance $O(1/T)$ of the old state. If each such state produced by one of the new subroutines is within Euclidean distance $O(1/T)$ of the corresponding original state produced by one of the QLE subroutines, then using an argument similar to that in Sec. IV A, the whole algorithm does not notice the difference between the original and the modified sequence of subroutines except with a low probability.

We now examine the complexity of the steps in the modified subroutines. Each use of step 1 solves the FEM with precision $O(1/T)$, which requires time poly(T) and a mesh of size poly(T). In step 2 we need to construct a known poly(T)-dimensional quantum state. This can be done in time poly(T) for any such state (see, e.g., claim 2.1.1 in [33]). If $\mathcal{M}$ and $\mathcal{M}'$ are suitably regular, the mapping required for step 3 can be implemented efficiently, i.e., in time polynomial in n, the number of qubits used by the original algorithm.

As the original quantum algorithm solved T instances of the FEM and acts nontrivially on all n qubits, its run time must be lower-bounded by $\max\{T,n\}$. Therefore, the run time of the new algorithm is at most polynomial in the run time of the old algorithm. As the new algorithm no longer contains any quantum subroutines which solve the FEM, we see that any quantum speedup achieved by it does not come from quantum acceleration of the FEM.

C. Solving oracular FEM instances

We finally observe that there cannot be an efficient quantum (or classical) algorithm for solving an instance of the FEM if the input function $f(x)$ is initially unknown and provided via an oracle (black box) and does not satisfy some smoothness properties. Indeed, this even holds for near-trivial FEM instances.

Imagine we are given an FEM instance of the form $u(x) = f(x)$, for $f \in L^2[0,1]$, and are asked to approximate the quantity $\int_0^{\frac{1}{2}} u(x)^2 dx$ to within accuracy ϵ: this is a very simple property of a trivial PDE. Further assume that we are given access to f via an oracle which maps $x \mapsto f(x)$ for $x \in [0,1]$, and that there are N possibilities for what the function f can be. We show that this problem is hard by encoding an unstructured search on N elements as an instance of the FEM.

Let B be the "bump" function defined by $B(x) = \exp(-1/(1-x^2))$ for $-1 < x < 1$, and $B(x) = 0$ elsewhere. Fix N and let f_0 be the shifted and rescaled bump function $f_0(x) = \sqrt{N}B(2Nx-1)$. f_0 is supported only on $[0,1/N]$ and has continuous derivatives of all orders, and $\|f_0\| = \Theta(1)$.

Assume that we have access to an oracle function $O : \{0,\dots,N-1\} \to \{0,1\}$ such that there is a unique $y_0 \in \{0,\dots,N-1\}$ with $O(y_0) = 1$. It is known that determining whether $y_0 < N/2$ or $y_0 \geqslant N/2$ requires $\Omega(\sqrt{N})$ quantum queries to O [34]. We define f in terms of O as follows. Given $x \in [0,1]$, set $y = \lfloor Nx \rfloor$ and evaluate $O(y)$. If the answer is 1, return $f_0(x - y/N)$. Otherwise, return 0.

f (equivalently, u) is a bump function in the range $[y_0/N,(y_0+1)/N]$ and is 0 elsewhere. So, if $y_0 < N/2$, $\int_0^{\frac{1}{2}} u(x)^2 dx \geqslant C$ for some constant $C > 0$, while if $y_0 \geqslant N/2$, $\int_0^{\frac{1}{2}} u(x)^2 dx = 0$. Hence approximating this integral up to additive accuracy ϵ, for sufficiently small constant $\epsilon > 0$, allows us to determine whether or not $y_0 < N/2$. As this task requires $\Omega(\sqrt{N})$ quantum queries, solving this instance of the FEM must require $\Omega(\sqrt{N})$ queries to f. A similar classical lower bound of $\Omega(N)$ queries also holds. Note that this does not contradict the bound, (3), as the norms of derivatives of u are large.

V. CONCLUSIONS

We have shown that, when one compares quantum and classical algorithms for the FEM fairly by considering every aspect of the problem—including the complexity of producing an accurate approximation of the desired classical output—an apparent exponential quantum advantage can sometimes disappear. However, there are still two types of problems where quantum algorithms for the FEM could achieve a significant advantage over classical algorithms: those where the solution has large higher-order derivatives and those where the spatial dimension is large.

For ease of comparison with the quantum algorithm, we have only considered a very simple classical FEM algorithm here; there is a large body of work concerned with improving the complexity of such algorithms. For example, the finite element mesh can be developed adaptively and made more refined near parts of the domain which are more complex or of particular interest. This can substantially improve the convergence speed. It is our suspicion that more advanced classical FEM algorithms might eliminate the quantum algorithm's advantage with respect to BVPs whose solutions have large higher-order derivatives.

For example, adaptive schemes such as "hp-FEM" have, in principle, a discretization error that scales far better than the scaling shown here; it can be shown [35] that a perfect adaptive scheme has scaling

$$\|u - \tilde{u}\| = O(e^{-\frac{1}{h}}),$$

provided that the dimension of the domain is both small and fixed. While this is a large improvement over the "vanilla" classical complexity presented above, it is not always apparent how to generate adaptive schemes that are effective enough to saturate this scaling, in practice. Also, it does not seem impossible that the quantum algorithm could be substantially improved using similar adaptive schemes.

Additionally, the case for the possibility of substantial improvement in the classical algorithm is less clear with respect to problems in high spatial dimension d. Indeed, any reasonable discretization procedure seems likely to lead to systems of linear equations which are of size exponential in d (this is the so-called "curse of dimensionality"). This is precisely the regime in which the quantum algorithm might be expected to have a significant advantage. One setting in which such high-dimensional BVPs occur is mathematical finance; for example, the problem of pricing multiasset basket options using the Black-Scholes equation [12]. Alternatively, producing a solution to any problem in many-body dynamics requires solving a PDE where the dimension grows with the number of bodies. However, Monte Carlo methods and related techniques can sometimes be used to alleviate the curse of dimensionality in practice [36,37]. It is therefore an interesting open question whether quantum algorithms can in fact yield an exponential speedup for problems of practical interest in this area.

ACKNOWLEDGMENTS

A.M. was supported by the UK EPSRC under Early Career Fellowship EP/L021005/1 and would like to thank Robin Kothari for helpful discussions and explanations of the results in [3]. S.P. was supported by the EPSRC Centre for Doctoral Training in Quantum Engineering.

APPENDIX A: USE OF THE HHL ALGORITHM TO APPROXIMATE THE NORM OF THE SOLUTION

Assume that we have an s-sparse system of linear equations $A\mathbf{x} = \mathbf{b}$, for some Hermitian $N \times N$ matrix A such that $\lambda_{\max}(A) \leqslant 1$, $\lambda_{\min}(A) \geqslant 1/\kappa$. We would like to approximate $\|\mathbf{x}\|$ up to accuracy $\epsilon\|\mathbf{x}\|$ using the HHL algorithm [1]. Here we sketch how the complexity of this task can be bounded, using the same notation as in Theorem III (see [1] for further technical details). The HHL algorithm is based on a subroutine $\mathcal{P}_{\text{sim}}$ whose probability of acceptance is approximately $p := \|A^{-1}|b\rangle\|^2/\kappa^2$. For any $\delta > 0$, approximating the probability p that a subroutine accepts, up to additive accuracy δp, can be achieved using amplitude estimation [27] with $O(1/(\delta\sqrt{p}))$ uses of the subroutine. Therefore, approximating $\kappa\|\mathbf{b}\|\sqrt{p} = \|\mathbf{x}\|$

up to additive accuracy $\epsilon\|\mathbf{x}\|$ can be achieved with

$$O\left(\frac{\kappa\|\mathbf{b}\|}{\epsilon\|\mathbf{x}\|}\right) = O\left(\frac{\kappa}{\epsilon}\right)$$

uses of $\mathcal{P}_{\text{sim}}$, where we use $\lambda_{\max}(A) \leqslant 1$. The run time of the $\mathcal{P}_{\text{sim}}$ subroutine, which is described in [1], depends on the accuracy with which its actual probability of acceptance $\widetilde{p}$ approximates p. Using the best-known algorithm for Hamiltonian simulation [38] within $\mathcal{P}_{\text{sim}}$, an accuracy of $|\widetilde{p} - p| = O(\epsilon p)$ can be achieved with $O((s\kappa/\epsilon)\,\text{poly}\log(s\kappa/\epsilon))$ uses of the algorithm $\mathcal{P}_A$ for determining entries in A. The run time is the same up to a polylogarithmic term in N, s, κ, and ϵ. Each use of the subroutine within amplitude estimation requires two uses of $\mathcal{P}_b$ to reflect about state $|b\rangle$. Therefore, the overall number of uses of $\mathcal{P}_A$ required is

$$O((s\kappa^2/\epsilon)\,\text{poly}\log(s\kappa/\epsilon)),$$

and the number of uses of $\mathcal{P}_b$ is $O(\kappa/\epsilon)$. Note that quantum linear equation algorithms subsequent to HHL [2,3] achieved a better dependence on κ, ϵ, or both for the task of producing $|x\rangle$; however, it does not seem obvious how to use these to achieve an improved accuracy for estimating $\|\mathbf{x}\|$.

APPENDIX B: PROOF OF THE TECHNICAL BOUND

In this Appendix we prove the claimed bound in Sec. III D that

$$\frac{\alpha}{\|r\|} = O(h\sqrt{s}),$$

where $\alpha = \left(\sum_i \langle\phi_i, r\rangle^2\right)^{1/2}$. Indeed, we show that

$$\sup_{r\neq 0} \frac{\left(\sum_i \langle\phi_i, r\rangle^2\right)^{1/2}}{\|r\|} = O(h\sqrt{s}).$$

Observe that this expression will be maximized when r is in the subspace spanned by the $\{\phi_i\}$ functions, so we can assume that $r = \sum_i \mathbf{r}_i\phi_i$ for some $\mathbf{r}_i$. Then the numerator satisfies

$$\begin{aligned}\left(\sum_i \langle\phi_i, r\rangle^2\right)^{1/2} &= \left(\sum_i \left(\int_\Omega \phi_i(\mathbf{x}) r(\mathbf{x}) d\mathbf{x}\right)^2\right)^{1/2}\\ &= \left(\sum_i \left(\int_\Omega \phi_i(\mathbf{x}) \sum_j \mathbf{r}_j\phi_j(\mathbf{x})\right)^2\right)^{1/2}\\ &= \left(\sum_i \left(\sum_j \mathbf{r}_j \int_\Omega \phi_i(\mathbf{x})\phi_j(\mathbf{x}) d\mathbf{x}\right)^2\right)^{1/2}\\ &= \|W\mathbf{r}\|,\end{aligned}$$

where we define the matrix $W_{ij} := \int_\Omega \phi_i(\mathbf{x})\phi_j(\mathbf{x}) d\mathbf{x}$. Similarly, for the denominator we have

$$\begin{aligned}\|r\| &= \left(\int_\Omega \left(\sum_i \mathbf{r}_i\phi_i(\mathbf{x})\right)^2 d\mathbf{x}\right)^{1/2}\\ &= \left(\sum_{i,j} \mathbf{r}_i\mathbf{r}_j \int_\Omega \phi_i(\mathbf{x})\phi_j(\mathbf{x}) d\mathbf{x}\right)^{1/2} = (\mathbf{r}^T W\mathbf{r})^{1/2}.\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{\alpha}{\|r\|} &\leqslant \sup_{\mathbf{r}\neq 0} \left(\frac{\mathbf{r}^T W^T W\mathbf{r}}{\mathbf{r}^T W\mathbf{r}}\right)^{1/2}\\ &= \sup_{\mathbf{r}', \|\mathbf{r}'\|=1} ((\mathbf{r}')^T W\mathbf{r}')^{1/2} = \|W\|^{1/2}.\end{aligned}$$

Assume that W is s sparse. To upper-bound $\|W\|$ we use

$$\|W\| \leqslant s \max_{i,j} |W_{ij}| = s \max_{i,j} |\langle\phi_i, \phi_j\rangle| \leqslant s \max_i \|\phi_i\|^2,$$

where the first inequality can be found in [39], for example, and the second is Cauchy-Schwarz. Then

$$\|\phi_i\|^2 = \int_T \phi_i(\mathbf{x})^2 d\mathbf{x} \leqslant h^d \max_{\mathbf{x}\in T} \phi_i(\mathbf{x})^2 = O(h^2),$$

where we assume that ϕ_i is supported in a region T of diameter at most h, and we use (6) to bound $\max_{\mathbf{x}\in T} \phi_i(\mathbf{x})^2 = O(h^{2-d})$. Thus $\alpha/\|r\| = O(h\sqrt{s})$.

[1] A. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, Phys. Rev. Lett. **103**, 150502 (2009).

[2] A. Ambainis, Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations, in *Proceedings of the 29th Annual Symposium on Theoretical Aspects of Computer Science. Leibniz International Proceedings in Informatics, Vol. 14* (Schloss Dagstuhl—Leibniz Center for Informatics, Wadern, Germany, 2012), pp. 636–647.

[3] A. Childs, R. Kothari, and R. Somma, Quantum linear systems algorithm with exponentially improved dependence on precision (2015).

[4] S. Aaronson, Quantum machine learning algorithms: Read the fine print, Nature Phys. **11**, 291 (2015).

[5] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning (2013).

[6] G. Wang, Quantum algorithms for approximating the effective resistances in electrical networks (2013).

[7] O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems: Theory and Computation* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001).

[8] S. C. Brenner and L. R. Scott, The Mathematical Theory of Finite Element Methods. *Texts in Applied Mathematics, Vol. 15* (Springer, New York, 2008).

[9] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems* (Elsevier, Amsterdam, 1978).

[10] S. S. Rao, *The Finite Element Method in Engineering* (Butterworth-Heinemann, London, 2005).

[11] B. Clader, B. Jacobs, and C. Sprouse, Preconditioned Quantum Linear System Algorithm, Phys. Rev. Lett. **110**, 250504 (2013).

[12] L. Jiang and C. Li, *Mathematical Modeling and Methods of Option Pricing* (World Scientific, Singapore, 2005).

[13] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, Quantum algorithm and circuit design solving the Poisson equation, New J. Phys. **15**, 013021 (2013).
[14] D. Berry, High-order quantum algorithm for solving linear differential equations, J. Phys. A: Math. Theor. **47**, 105301 (2014).
[15] S. Leyton and T. Osborne, A quantum algorithm to solve nonlinear differential equations (2008).
[16] J. Shewchuk, An introduction to the conjugate gradient method without the agonizing pain, Technical Report CMU-CS-TR-94-125 (Carnegie Mellon University, Pittsburgh, PA, 1994); http://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.ps.
[17] A. Ern and J.-L. Guermond, *Theory and Practice of Finite Elements* (Springer-Verlag, Berlin, 2013).
[18] M. Benzi, C. D. Meyer, and M. Tuma, A sparse approximate inverse preconditioner for the conjugate gradient method, SIAM J. Sci. Comput. **17**, 1135 (1996).
[19] M. Benzi and M. Tuma, A comparative study of sparse approximate inverse preconditioners, Appl. Numer. Math. **30**, 305 (1999).
[20] S. S. Li, P. L. Rui, and R. S. Chen, An effective sparse approximate inverse preconditioner for vector finite element analysis of 3D EM problems, in *IEEE International Symposium on Antennas and Propagation Society, 2006* (IEEE, New York, 2006), pp. 1765–1768.
[21] X. W. Ping and T.-J. Cui, The factorized sparse approximate inverse preconditioned conjugate gradient algorithm for finite element analysis of scattering problems, Prog. Electromagnet. Res. **98**, 15 (2009).
[22] C. Zalka, Simulating quantum systems on a quantum computer, Proc. Roy. Soc. A: Math., Phys. Eng. Sci. **454**, 313 (1998).
[23] L. Grover and T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions (2002).
[24] P. Kaye and M. Mosca, Quantum networks for generating arbitrary quantum states, in *Optical Fiber Communication Conference and International Conference on Quantum Information, 2001*, OSA Technical Digest Series (Optical Society of America, Washington DC, 2001).
[25] R. Haber, M. S. Shephard, J. F. Abel, R. H. Gallagher, and D. P. Greenberg, A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mappings, Int. J. Numer. Methods Eng. **17**, 1015 (1981).
[26] D. Aharonov, V. Jones, and Z. Landau, A polynomial quantum algorithm for approximating the Jones polynomial, Algorithmica **55**, 395 (2009).
[27] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, Quantum amplitude amplification and estimation, in *Quantum Computation and Quantum Information: A Millennium Volume. AMS Contemporary Mathematics Series, Vol. 305* (American Mathematical Society, Providence, RI, 2002), pp. 53–74.
[28] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, Quantum Fingerprinting, Phys. Rev. Lett. **87**, 167902 (2001).
[29] R. E. Bank and L. R. Scott, On the conditioning of finite element equations with highly refined meshes, SIAM J. Numer. Anal. **26**, 1383 (1989).
[30] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, Strengths and weaknesses of quantum computing, SIAM J. Comput. **26**, 1510 (1997).
[31] C. W. Helstrom, *Quantum Detection and Estimation Theory* (Academic Press, New York, 1976).
[32] S. Chakraborty, E. Fischer, A. Matsliah, and R. de Wolf, New results on quantum property testing, in *Proceedings of FSTTCS, Leibniz International Proceedings in Informatics (LIPIcs)* (Schloss-Dagstuhl - Leibniz Center for Informatics, Dagstuhl, Germany, 2010), pp. 145–156.
[33] A. Prakash, Quantum algorithms for linear algebra and machine learning, Ph.D. thesis, University of California, Berkeley, 2014.
[34] L. Grover and J. Radhakrishnan, Is partial quantum search of a database any easier? in *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '05* (Association for Computing Machinery, New York, 2005), pp. 186–194.
[35] B. Guo and I. Babuška, The H-P version of the finite element method, Comput. Mech. **1**, 203 (1986).
[36] P. Boyle, M. Broadie, and P. Glasserman, Monte Carlo methods for security pricing, J. Econ. Dynam. Control **21**, 1267 (1997).
[37] P. L'Ecuyer, Quasi-Monte Carlo methods with applications in finance, Finance Stochast. **13**, 307 (2009).
[38] D. Berry, A. Childs, and R. Kothari, Hamiltonian simulation with nearly optimal dependence on all parameters, in *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS 2015)* (IEEE, New York, 2015), pp. 792–809.
[39] A. Childs and R. Kothari, Limitations on the simulation of non-sparse Hamiltonians, Quantum Info. Comput. **10**, 669 (2010).