Quantum variational solving of nonlinear and multidimensional partial differential equations

Abhijat Sarma,[1,*] Thomas W. Watts,[2] Mudassir Moosa,[1,3] Yilian Liu,[2] and Peter L. McMahon[2,†]
[1]*Department of Physics, Cornell University, Ithaca, New York 14853, USA*
[2]*School of Applied and Engineering Physics, Cornell University, Ithaca, New York 14853, USA*
[3]*Department of Physics and Astronomy, Purdue University, West Lafayette, Indiana 47907, USA*

(Received 25 February 2024; accepted 28 May 2024; published 21 June 2024)

A variational quantum algorithm for numerically solving partial differential equations (PDEs) on a quantum computer was proposed by Lubasch *et al.* [Phys. Rev. A **101**, 010301(R) (2020)]. In this paper we generalize the method introduced by Lubasch *et al.* to cover a broader class of nonlinear PDEs as well as multidimensional PDEs and study the performance of the variational quantum algorithm on several example equations. Specifically, we show via numerical simulations that the algorithm can solve instances of the single-asset Black-Scholes equation with a nontrivial nonlinear volatility model, the double-asset Black-Scholes equation, the Buckmaster equation, and the deterministic Kardar-Parisi-Zhang equation. Our simulations use up to $n = 12$ *Ansatz* qubits, computing PDE solutions with 2^n grid points. We also perform proof-of-concept demonstrations with a trapped-ion quantum processor from IonQ [Nat. Commun. **10**, 5464 (2019)], showing accurate computation of two representative expectation values needed for the calculation of a single time step of the nonlinear Black-Scholes equation. Through our classical simulations and demonstrations on quantum hardware, we identify and discuss several open challenges for using quantum variational methods to solve PDEs in a regime with a large number (much greater than 2^{20}) of grid points, but also a practical number of gates per circuit and circuit shots.

DOI: 10.1103/PhysRevA.109.062616

I. INTRODUCTION

Quantum computing is an alternative paradigm for computation that, for some applications, may enable an exponential advantage in space (memory) and/or time versus classical approaches. One recently proposed application of quantum computing is to solving partial differential equations (PDEs) via a variational approach [1]. The general approach in variational quantum algorithms [2] is as follows. A proposed solution to a problem is represented as a quantum state that is prepared using a low-depth parametrized quantum circuit, often called the *Ansatz* circuit. A cost function, which ideally should be easily computable given a proposed solution state, quantifies the quality of the state as a solution to the problem being solved. An optimizer running on a classical computer is used to update the parameters of the *Ansatz* circuit with the goal of minimizing the cost function. Variational quantum algorithms tend to have lower qubit-count and circuit-depth requirements than more traditional quantum algorithms [3] and partially for this reason have attracted attention as candidates to run on noisy intermediate-scale quantum processors [4].

Nonlinear and multidimensional PDEs are typically intractable analytically and classical numerical methods can be very costly, especially for multidimensional PDEs in which the curse of dimensionality, where costs increase exponentially with the PDE dimension, can arise. Following the publication of the proposal by Lubasch *et al.* [1] to solve PDEs variationally on quantum computers, several studies have explored variational approaches to solving a variety of PDEs [5–14].[1] However, not all of these methods are applicable to nonlinear or multidimensional problems and thus far none have been demonstrated in numerical simulation with more than six *Ansatz* qubits (i.e., $2^6 = 64$ grid points) on nonlinear problems. In Ref. [1] Lubasch *et al.* introduced a method to solve linear homogeneous PDEs and a subset of nonlinear PDEs that are first order in time; they demonstrated their method's applicability to the one-dimensional (1D) heat equation and the Burgers equation. In Ref. [7] Mocz and Szasz show how to use the method of Lubasch *et al.*, together with a subroutine to train a state to represent a nonlinear potential, to solve the Schrödinger-Poisson equation. In Ref. [10] Yew Leong *et al.* solve reaction-diffusion equations in a similar manner. In this paper we present a general version of this quantum method for solving PDEs and demonstrate explicitly how to train intermediate quantum states to represent a large class of nonlinearities and inhomogeneous terms, making the method applicable to most nonlinear and multidimensional PDEs of interest which are first order in time, including the Navier-Stokes equations, Maxwell's equations, the Calabi flow equation, and the Gross-Pitaevskii equation, with applications to fluid mechanics,

*Contact author: as3232@cornell.edu
†Contact author: pmcmahon@cornell.edu

[1]There is also a rich and interesting literature (e.g., Refs. [14–20]) studying how quantum computers could solve PDEs using nonvariational approaches, such as via Hamiltonian simulation or using quantum linear system solving [21] as a subroutine, which we will not review.

electromagnetism, differential geometry, and Bose-Einstein condensates, respectively. We demonstrate how the method can be used to adapt many explicit and semi-implicit numerical schemes, such as the fourth-order Runge-Kutta (RK4) algorithm. We further introduce *Ansätze* well suited for solving differential equations and benchmark them with several popular out-of-the-box optimizers. We then apply the method to the solution of a nonlinear Black-Scholes equation (BSE) with eight *Ansatz* qubits, the 2D linear BSE with six *Ansatz* qubits per degree of freedom (giving $6 \times 2 = 12$ total *Ansatz* qubits), the Buckmaster equation with five *Ansatz* qubits, and the deterministic Kardar-Parisi-Zhang equation with five *Ansatz* qubits, demonstrating applications to options pricing, viscous fluid, and surface growth problems. Finally, we show two-qubit proof-of-concept results on an 11-qubit trapped-ion device offered by IonQ. The hardware details of the device can be found in Ref. [22].

II. GENERAL ALGORITHMIC FRAMEWORK

Consider a general PDE which is first order in time, $\frac{\partial u}{\partial t} = \hat{O}[u]$, where $\hat{O}$ is some potentially nonlinear differential operator depending on u, with an initial condition $u(x, t = 0) = u_0(x)$. Our methodology, based on that of Lubasch *et al.* [1], is to represent the solution $u(x, t)$ as a quantum state $|u(t)\rangle = \sum_{k=0}^{2^n-1} u(x_k, t)|k\rangle$, where n is the number of qubits. In other words, $|u(t)\rangle$ has amplitudes in the computational basis consisting of the function $u(x, t)$ evaluated at each grid point x_k in the discretized domain, of which there is a total of $N = 2^n$. The problem of solving the PDE is analogous to finding the solution state after some small period of time τ, $|u(t + \tau)\rangle$, using the solution state at time t, $|u(t)\rangle$. This state $|u(t)\rangle$ is represented by a low-depth parametrized gate $\hat{U}(\boldsymbol{\lambda})$, known as the *Ansatz*, acting on the $|0\rangle$ state, and a cost function is constructed which, when classically optimized, returns the optimal values of the parameters $\boldsymbol{\lambda} \in \mathbb{R}^{n_p}$ corresponding to the next time step based on the values of the parameters at the current time step. Since the function $u(t)$ is not necessarily normalized, we need an additional parameter λ_0 such that $|u(t)\rangle = \lambda_0 \hat{U}(\boldsymbol{\lambda})|0\rangle = \lambda_0|\psi\rangle$, where we have defined $|\psi\rangle$ as the normalized state generated by the variational *Ansatz*. To construct a cost function, we first replace the time derivative with a finite-difference approximation, e.g., the backward Euler approximation, wherein our discretized PDE becomes $(\mathbb{1} - \tau\hat{O})u(t + \tau) = u(t)$. Then we define the cost function as the square distance between both sides of this equation $C_u = \|(\mathbb{1} - \tau\hat{O})|u(\lambda_0, \boldsymbol{\lambda})\rangle - |\tilde{u}(\tilde{\lambda}_0, \tilde{\boldsymbol{\lambda}})\rangle\|^2$, where we use a tilde to denote the previous time step. The power of this method comes from the fact that this cost function can be efficiently calculated on a quantum computer using the inner product, assuming that we can decompose $\hat{O}$ into quantum gates, as

$$C(\lambda_0, \boldsymbol{\lambda}) = \langle \lambda_0(\mathbb{1} - \tau\hat{O})\psi(\boldsymbol{\lambda}) - \tilde{\lambda}_0\tilde{\psi}(\tilde{\boldsymbol{\lambda}})|\lambda_0(\mathbb{1} - \tau\hat{O})\psi(\boldsymbol{\lambda}) - \tilde{\lambda}_0\tilde{\psi}(\tilde{\boldsymbol{\lambda}})\rangle. \quad (1)$$

The problem remains of how to convert the differential operator $\hat{O}$ into a linear combination of quantum operators that can be implemented by standard quantum gates (see Fig. 1). Any spatial derivative term in $\hat{O}$ can be easily constructed after introducing the adder operator $\hat{A}$. As described in the Supplemental Material of Ref. [1], the adder acts on a state $|\psi\rangle = \sum_{k=0}^{2^n-1} \psi_k|k\rangle$ by cyclically permuting it as $\hat{A}|\psi\rangle = \sum_{k=1}^{2^n} \psi_k|k-1\rangle$, where $\psi_{2^n} = \psi_0$. The conjugate $\hat{A}^\dagger$ clearly has the opposite effect of cyclically permuting it in the other direction. Equipped with this operator, we can approximate spatial derivatives using central finite differences $\frac{\hat{\partial}}{\partial x} = \frac{2^{n-1}}{L}(\hat{A} - \hat{A}^\dagger)$, $\frac{\hat{\partial}^2}{\partial x^2} = \frac{4^n}{L^2}(\hat{A} + \hat{A}^\dagger - 2\mathbb{1})$, and so on for higher-order derivatives, where L is the length of the domain. Note that our use of the adder operator imposes periodic boundary conditions on our solution, which may not always be desirable. However, various coordinate substitutions can be made to convert various other boundary conditions, such as Dirichlet conditions, into periodic boundary conditions. Further, Dirichlet boundary conditions may be approximated by periodic boundary conditions by doubling the spatial domain and reflecting the function about one of the original domain's end points, to create and time evolve a new function which is now initially periodic, as in Ref. [23], a method which we use to solve the Black-Scholes equation in Secs. V and VI. Recent papers have also introduced a method to recover Dirichlet conditions by calculating a small number of additional expectation values, as in Ref. [8].

In order to deal with nonlinearities in $\hat{O}$, we must define the diagonal operator $\hat{D}$ introduced in Ref. [1]. Given a state $|\beta\rangle = \sum_{k=0}^{2^n-1} \beta_k|k\rangle$, the diagonal operator $\hat{D}_\beta$ acts on a state $|\alpha\rangle = \sum_{k=0}^{2^n-1} \alpha_k|k\rangle$ as $\hat{D}_\beta|\alpha\rangle = \sum_{k=0}^{2^n-1} \beta_k\alpha_k|k\rangle$. In other words, it acts as a point-wise multiplication operator between two quantum states; note that this is not a unitary transformation on $|\alpha\rangle$ and as such requires n dirty ancilla qubits to implement, where n is the number of qubits in $|\beta\rangle$ and $|\alpha\rangle$. Further, this means that $\hat{D}^\dagger\hat{D}$ is not equal to identity. This operator can be constructed from any gate which generates $|\beta\rangle$, i.e., $\hat{U}_\beta$ such that $\hat{U}_\beta|0\rangle = |\beta\rangle$. This will allow us to generate nonlinear operators. Consider a general, potentially nonlinear operator of the form $\hat{O}[u] = f(x, u, \frac{\partial u}{\partial x}, \ldots)$, where f is analytic in u and its derivatives. We need to construct a quantum state $|f\rangle = \hat{O}|u\rangle$ or, for an explicit method, $|f\rangle = \hat{O}|\tilde{u}\rangle$, which we can then inject into our cost function to evaluate it from simple expectation values. As Lubasch *et al.* [1] point out, this is easy to do when $|f\rangle$ can be generated by repeated applications of adders and diagonal gates to $|u\rangle$, as is the case for operators which are polynomial in u and linear in its derivatives. For example, a nonlinear differential operator $\hat{O}[u] = u^2\frac{\partial u}{\partial x}$ has a quantum analog $|f\rangle = \hat{O}|u\rangle = \hat{D}_u^2\frac{\hat{\partial}}{\partial x}|u\rangle = \frac{2^{n-1}}{L}\hat{D}_u^2(\hat{A} - \hat{A}^\dagger)|u\rangle$.

However, for more complicated nonlinearities or operators with an explicit x dependence, it is not clear how one could construct $|f\rangle$ by directly applying operators to $|u\rangle$ or $|\tilde{u}\rangle$. The key is to circumvent this problem by representing $|f\rangle$ variationally in the same way as $|u\rangle$, by optimizing for parameters θ_0 and $\boldsymbol{\theta}$ to represent $|f(\theta_0, \boldsymbol{\theta})\rangle$ from the previous time step's solution $|\tilde{u}\rangle$. This can be done by breaking down $f(x, u, \ldots)$ into simple functions which can be represented by direct applications of adder and diagonal operators, and then synthesizing these simple functions back together into the full nonlinearity. Depending on how complicated the nonlinearity

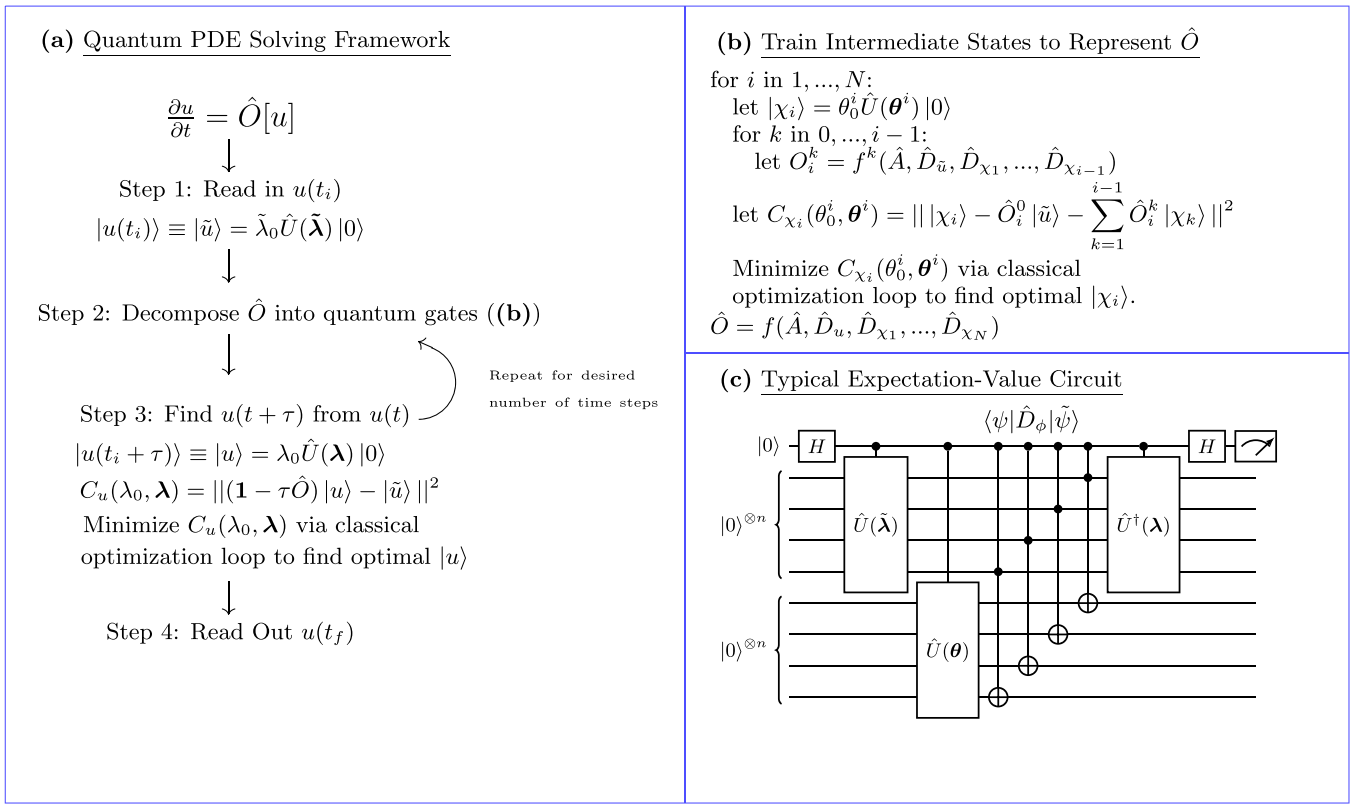

FIG. 1. Algorithmic framework. (a) Quantum variational-PDE-solving workflow. Here $\hat{U}$ is an *Ansatz* gate, and λ_0 and $\boldsymbol{\lambda}$ are the variational parameters to be trained to represent our function of interest, with a tilde denoting the previous time step's parameters; C_u is the main cost function to be minimized, returning the optimal state after one time step. First, read in initial *Ansatz* parameters to represent $u(t_i)$. Decompose the discretized (nonlinear) differential operator $\hat{O}$ into adder and diagonal gates. By training intermediate quantum states, a much broader class of nonlinearities can be represented than possible through naive applications of the adder and diagonal gates as considered in Ref. [1]. Then use $\hat{O}$ to construct a cost function which is then optimized to find $|u\rangle$ from $|\tilde{u}\rangle$, as in Ref. [1]. Here the cost function shown represents the backward Euler discretization scheme, though it can easily be adapted to other schemes. Repeat for the desired number of time steps and then extract the desired classical data from the solution at the final time step $|u(t_f)\rangle$. (b) Decomposition of the operator $\hat{O}$ into adder and diagonal gates via training intermediate quantum states. Here θ_0^i and $\boldsymbol{\theta}^i$ are the variational parameters corresponding to intermediate states $|\chi^i\rangle$, which are utilized in diagonal gates $\hat{D}_{\chi_i}$to synthesize the full nonlinear operator $\hat{O}$. Cost functions C_{χ_i} are iteratively constructed utilizing the adder gate and the diagonal gates of states already trained, which are then minimized to return the optimal intermediate parameters to represent any necessary nonlinearities or inhomogeneous terms. The number of intermediate states N and the functions f^k and f will depend on the exact form of $\hat{O}$. (c) Example of the estimation of a typical expectation value appearing in the cost functions via the Hadamard test. Here $|\psi\rangle$ and $|\phi\rangle$ represent the normalized versions of $|u\rangle$ and $|\chi\rangle$, respectively.

is, this may require several intermediate optimization steps. This makes the numerical scheme semi-implicit, whereby complicated nonlinearities are calculated explicitly from the previous time step's solution before being injected into the implicit time evolution.

For example, consider a nonlinearity of the form $f(\frac{\partial u}{\partial x}) = \sin(\frac{\partial u}{\partial x})$. We expand sin into a truncated Taylor series, to find $f(\frac{\partial u}{\partial x}) \approx \frac{\partial u}{\partial x} - \frac{1}{6}(\frac{\partial u}{\partial x})^3 + \frac{1}{120}(\frac{\partial u}{\partial x})^5$. Let $|\chi\rangle = \theta_0 \hat{U}(\boldsymbol{\theta})|0\rangle = \frac{\hat{\partial}}{\partial x}|\tilde{u}\rangle$. We can solve for θ_0 and $\boldsymbol{\theta}$ by optimizing the cost function $C_\chi(\theta_0, \boldsymbol{\theta}) = \||\chi\rangle - \frac{\hat{\partial}}{\partial x}|\tilde{u}\rangle\|^2$. After doing so, we can fully construct $|f(\theta_0, \boldsymbol{\theta})\rangle = |\chi\rangle - \frac{1}{6}\hat{D}_\chi^2|\chi\rangle + \frac{1}{120}\hat{D}_\chi^4|\chi\rangle$, which can then be used to time evolve our PDE of interest. It is straightforward to construct operators with an explicit x dependence as well, by using quantum state preparation techniques to prepare the state $\sum_{k=0}^{2^n-1} f(x_k)|k\rangle$ corresponding to a generic function $f(x)$. A form of this technique for generating nonlinearities was used in Ref. [7] to solve a nonlinear Poisson equation and in Ref. [10] for reaction-diffusion equations. Here we have presented the most general version, which is applicable to nearly all first-order-in-time PDEs of interest in the literature. The method is straightforwardly adapted to other Runge-Kutta methods, such as the celebrated RK4 algorithm. In the following sections we demonstrate its accuracy for solving several differential equations with vastly different forms of nonlinearities.

It is obvious that this algorithm requires the *Ansatz* parameters representing the initial condition $u(x, t=0) = u_0(x)$ *a priori*. We defer the discussion of how one can find the appropriate parameters to represent the initial condition to the next section. For optimization purposes, we calculate all of the cost functions appearing in this work using matrix multiplications to directly apply the effects of the adder and diagonal gates to avoid the computational overhead of having to explicitly simulate ancilla qubits, though we did simulate

entire circuits with the adder and diagonal gates using Cirq for a few time steps of each equation to ensure the results were the same as when using the computational shortcut. All of our expectation values in our simulations except for Sec. IX were exactly calculated using this shortcut.

III. *ANSATZ* SELECTION

One important element of this algorithm which has been ignored thus far is the form of the *Ansatz* gate $U(\boldsymbol{\lambda})$. One needs an *Ansatz* which is expressive enough to represent the solution states, the shape of which may not be known *a priori*, yet which is not too general as to impede its trainability. In particular, the *Ansatz* needs to be able to represent functional states, but not general distributions. By functional states we mean states whose amplitudes form a piecewise analytic function, as any solution to a PDE with piecewise analytic terms must be piecewise analytic. This greatly restricts the type of states we wish to be able to represent, and as such it allows us to create a more specialized *Ansatz* which is well equipped to represent solutions to any PDE with piecewise analytic terms with a low number of parameters. In particular, we aim to represent solutions using a finite Fourier series, utilizing the powerful quantum Fourier transform (QFT). We do so using the so-called real-valued Zalka-Grover-Rudolph (ZGR) QFT *Ansatz*, shown in Appendix A, an extension of the Fourier series loader introduced in Ref. [24] which parametrizes real-valued partial Fourier series. This *Ansatz* allocates m qubits to serve as representing Fourier coefficients, of which there is a total of 2^m. These coefficients are fully parametrized using the ZGR parametrization, introduced in Ref. [9]. For a complex Fourier series, we could then apply the QFT to transform from momentum space to real space on all n *Ansatz* qubits, as done in [24]. In order to get a real-valued Fourier series, we must instead apply various controlled versions of the ZGR gate, as well as an adder and controlled Hadamard gates (see Appendix A). Due to the ZGR, this *Ansatz* has complexity $O(2^m)$; however, in most cases, we only need to choose $m \ll n$ to properly represent a functional state, as it rarely takes a very large number of terms in a finite Fourier series to represent sufficiently smooth functions. Accounting for the other gates such as the QFT, the total complexity of the *Ansatz* is $O(n\ln(n) + 2^m)$, with a number of rotation parameters equal to 2^{m+1}. This *Ansatz* has severable desirable properties. First, it is nonchaotic, in the sense that smooth changes in parameters lead to smooth changes in the output distribution. This makes optimization much easier, as optimal parameters change relatively little between time steps as compared to chaotic *Ansätze* (of which most popular *Ansätze* are, including the alternating layered *Ansatz*, hardware efficient *Ansatz*, etc.) and so they are easier to find for most classical optimizers. Second, there exist exact formulas for the parameters based on the target state one wishes to represent, as described in Ref. [25]. These formulas can easily be extended to the full ZGR QFT *Ansatz*, as the QFT is equivalent to a classical discrete Fourier transform. This effectively makes the task of readin, as in finding the *Ansatz* parameters to represent the initial condition $f(x, t = 0)$ [or, in the case of the Black-Scholes equation, the terminal condition $f(x, t = T)$], trivial. The inverse of these formulas also makes readout, as in the problem of extracting useful information from the solution state using the optimized *Ansatz* parameters, trivial in that one can classically reconstruct the solution state directly from the *Ansatz* parameters. In general, when using a different *Ansatz*, for example, one can always make use of approximate polynomial-time function loading methods and the SWAP test to train any variational *Ansatz* to represent a target initial condition distribution for readin. The problem of readout is more complicated in the general case, as the quantities of interest one may want to read out of the solution state depend on the problem at hand. However, recent work has been done on efficiently extracting moments of solutions [7] as well as using quantum machine learning for readout [26]. Finally, for any choice of m, the *Ansatz* has a constant number of parameters, regardless of n. This effectively means that for any choice of m, increasing n does not make the *Ansatz* more difficult to optimize, which is an exceedingly rare property for *Ansätze*. Note that for the real-valued ZGR QFT, m must be less than or equal to $n - 2$. This makes the *Ansatz* perform poorly for low numbers of qubits ($n < 5$), as one does not have access to enough Fourier modes to adequately represent solution states.

On the other hand, in some cases we do need to represent more jagged distributions which are not amenable to Fourier series. For the BSE in particular, the terminal condition is nondifferentiable at a point and as such its derivatives are discontinuous. Because of this, it is difficult to optimize the real-valued ZGR QFT *Ansatz* to represent $|\chi\rangle$ (defined in Sec. V). Instead, we employ the universal layered *Ansatz* (ULA), shown in Appendix B, which is a real-valued *Ansatz* that is similar in form to the popular alternating layered *Ansatz* (ALA) [27], but which is optimally efficient in the sense that it can fully parametrize a real n-qubit state with $\frac{2^n(2^n-1)}{2}$ parameters, which is the minimum number of parameters needed to parametrize $\mathrm{SO}(2^n)$. Further, it is hardware efficient, in that it only contains nearest-neighbor gates. This *Ansatz* has a number of parameters equal to $6(n-1)d$, where d is the number of layers of the *Ansatz*. When the states of interest have low entanglement, we expect that we can take d to be small. In practice, we find that the *Ansatz* has enough expressibility to represent the states of interest ($|\chi\rangle$) when d is of $O(n)$, giving a total number of parameters and gate depth of $O(n^2)$. In addition to representing very jagged distributions, the ULA tends to outperform the real-valued ZGR QFT *Ansatz* for low ($n < 5$) numbers of qubits. Figure 2 shows how the expressibility of the ZGR QFT *Ansatz* and the ULA scales with the number of parameters. It is suspected that ALA-like *Ansätze* are difficult to simulate classically. However, the many desirable properties of the ZGR QFT *Ansatz* also make it classically tractable, in the sense that circuits involving only the ZGR QFT *Ansatz* can be classically simulated in polynomial time. Thus, the ZGR QFT *Ansatz* alone is not enough to build a nontrivial quantum PDE-solving algorithm. It remains a crucial open problem to find an *Ansatz* with similar desirable properties while also remaining classically intractable.

IV. OPTIMIZATION

To optimize the cost functions appearing in this work, we use the previous time step's parameters as the initial guess

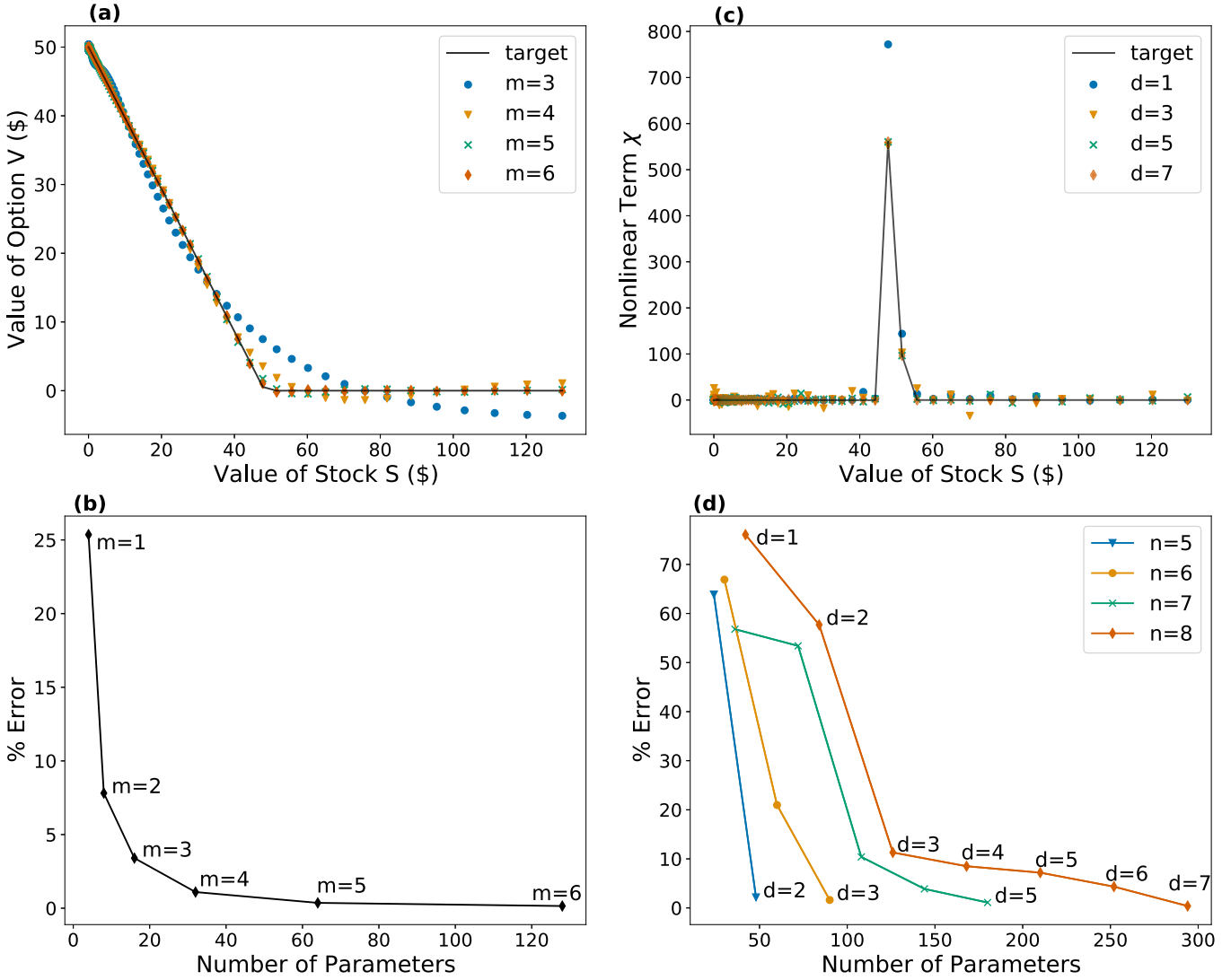

FIG. 2. Expressibility of the *Ansatz* vs the number of parameters. Here n is the number of qubits, m is the number of ZGR qubits for the ZGR QFT, and d is the depth for the ULA. The V represents the function of interest for the Black-Scholes problem (2), while χ is an intermediate state used to construct the nonlinearity in the problem. (a) and (b) The real-valued ZGR QFT *Ansatz* is able to accurately approximate the continuous terminal condition for $|V\rangle$ (the put option) and (c) and (d) the universal layered *Ansatz* is able to accurately approximate the sparse, discontinuous terminal condition for $|\chi\rangle$. We choose these terminal conditions as a representative example for the expressibility [28] of the *Ansätze* because they are the least smooth functions considered in the paper. All simulations were conducted with $n = 8$ *Ansatz* qubits, except for the plot in (d), where n varies. The ZGR QFT has the useful property that its expressibility as a function of the number of parameters is invariant with respect to n, assuming that the function being approximated is continuous. This follows from the fact that, for a constant m, the ZGR QFT will approximate the target distribution with the same number of Fourier coefficients regardless of n, meaning that the *Ansatz* does not become more difficult to train as the number of qubits increases. For the ULA, the necessary depth d to get a good approximation of the target distribution scales roughly linearly with the number of qubits, making the necessary number of parameters $O(n^2)$.

for the new time step's parameters and we conduct a line search on the norm parameter λ_0 with a very low budget before running a standard optimization algorithm on all of the parameters. We tested several different popular gradient-free optimizers (Differential Evolution, Implicit Filtering, Particle Swarm, Snob Fit, and an adaptive optimizer offered by the NEVERGRAD PYTHON package called NGOpt) on the first time step of the nonlinear Black-Scholes equation (2) for both cost functions and *Ansätze*, shown in Fig. 3. The previous time step's parameters were used as the initial guess of the new parameters for each optimizer. We choose the first time step as a representative case to test the optimization for the whole algorithm because the nonlinearity is largest at the beginning of the time evolution. We also tested some popular gradient-based optimizers (BFGS, Truncated Newton's Method, and Adam), though all of them required a budget (maximum number of cost-function evaluations per time step) at least on the order of $10^4 p$ to perform well, where p is the number of *Ansatz* parameters, which we found to be too high to extract useful results. This is not particularly surprising as we expect that our cost functions may suffer from the so-called barren plateau problem [2]. Further work may be done to investigate and mitigate the existence of barren plateaus in our cost functions. For our full simulation of the algorithm we use Differential

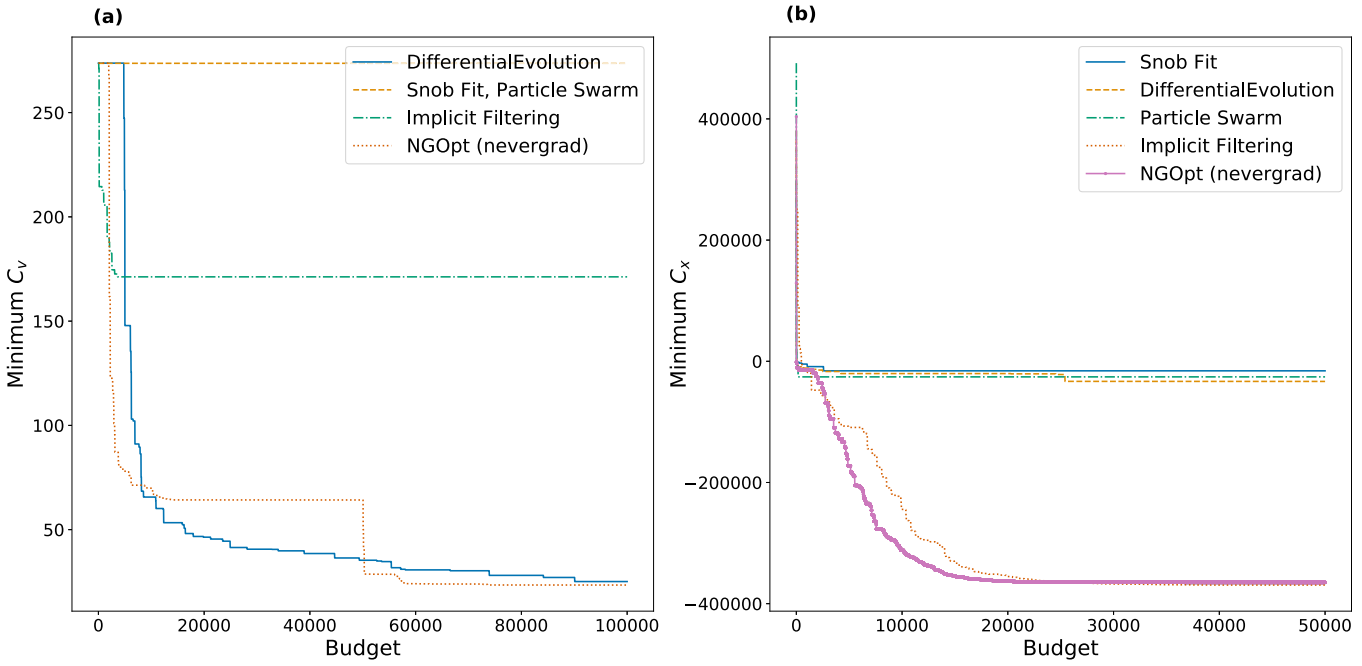

FIG. 3. Best cost-function value found by the optimizer vs the budget. Here budget refers to the total number of cost-function evaluations that the optimizer software is allowed to make per PDE time step in the PDE solving (step 3 in Fig. 1). (In our circuit simulations, to evaluate the cost function, we directly computed expectation values, so this plot has nothing to do with the number of times a circuit would need to be executed on quantum hardware to estimate each expectation value.) This figure shows how difficult it is for different optimizers to optimize the (a) ZGR QFT *Ansatz* and (b) ULA. Both *Anszätze* were set to use $n = 8$ qubits, with $m = 6$ for the ZGR QFT and $d = 6$ for the ULA. For the ZGR QFT only Differential Evolution and NGOpt converged to the global minimum, and for the ULA only NGOpt and Implicit Filtering reached the global minimum. The optimizations were performed for the first time step in solving the nonlinear Black-Scholes equation

Evolution to optimize the ZGR QFT *Ansatz* and NGOpt to optimize the universal layered *Ansatz*. Because the ZGR QFT *Ansatz* is smooth, we can place relatively tight bounds on the rotation parameters while optimizing them, as most of them only change on the order of 10^{-1} or 10^{-2}.

V. SOLVING THE BARLES-SONER NONLINEAR BLACK-SCHOLES EQUATION

A. Cost functions, circuits, and complexity

The Black-Scholes equation is a practically important PDE to solve in finance; it models the price of an option as a function of time and the underlying asset price [29]. Denoting by V the option price and by S the underlying asset price, the Black-Scholes model with the put-option terminal condition can be simplified by making the substitution $x = \ln(S)$ [23], reducing it to the form

$$\frac{\partial V}{\partial t} = rV + \left(\frac{\sigma^2}{2} - r\right)\frac{\partial V}{\partial x} - \frac{\sigma^2}{2}\frac{\partial^2 V}{\partial x^2},$$
$$V(x, t = T) = \max(K - e^x, 0),$$
$$V(x \to \infty, t) = 0. \tag{2}$$

Here r is the interest rate. The factor σ is a model-dependent value known as the volatility. In the linear BSE, the volatility is taken to be a constant, leading to a heatlike equation which can be easily solved analytically. However, more complicated and robust volatility models can have the volatility as a function of the option price, its derivatives, the stock price, and time. These models necessarily lead to a nonlinear PDE which is often impossible to solve analytically. Of these models, one of particular interest is the Barles-Soner model as described in Ref. [30], in which the volatility (squared) takes the form

$$\sigma^2 = \sigma_0^2\left[1 + e^{r(T-t)}a^2\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right)\right]. \tag{3}$$

From Eq. (2) it is clear that $\hat{O} = r + (\frac{\sigma^2}{2} - r)\frac{\partial}{\partial x} + \frac{\sigma^2}{2}\frac{\partial^2}{\partial x^2}$ in this case. To convert this into a quantum operator, we must first generate an intermediate quantum state to represent $\chi = \frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}$, $|\chi\rangle$. After we have the state $|\chi\rangle$, the quantum operator takes the form

$$\hat{O} = r\mathbb{1} + \left(\frac{\sigma_0^2}{2}(\mathbb{1} + e^{r(T-t)}a^2\hat{D}_\chi) - r\right)\frac{\hat{\partial}}{\partial x} + \frac{\sigma_0^2}{2}(\mathbb{1} + e^{r(T-t)}a^2\hat{D}_\chi)\frac{\hat{\partial}^2}{\partial x^2}. \tag{4}$$

This necessitates a two-cost-function scheme, in which at each time step we first find the optimal variational parameters to represent $|\chi\rangle$ from the previous time step's parameters for $|\tilde{V}\rangle$ and then we use those parameters to generate the diagonal gate $\hat{D}_\chi$, which in turn we use to calculate the cost function to find the optimal parameters to represent $|V\rangle$. We therefore also need a cost function for $|\chi\rangle$, which can straightforwardly be calculated using the inner product distance method as before. Using the backward Euler finite difference, our scheme

becomes semi-implicit, with the cost functions given by

$$C_\chi = \left\| |\chi\rangle - \left(\frac{\hat{\partial}^2}{\partial x^2} - \frac{\hat{\partial}}{\partial x}\right)|\tilde{V}\rangle \right\|^2,$$
$$C_V = \|(\mathbb{1} - \tau\hat{O})|V\rangle - |\tilde{V}\rangle\|^2, \tag{5}$$

where we have dropped the explicit parameter dependence of the cost functions. One could equivalently define and minimize a joint cost function $C_{\text{total}} \equiv C_V + \mu C_\chi$ for $\mu > 0$, though this makes optimization more difficult in practice. The full cost functions can be found in Appendix C. The equivalent cost function for the 1D linear model in which the volatility is a constant is included in Appendix C as well. The nonlinear cost function for V contains 19 expectation values which can be calculated via the Hadamard test. The diagonal gate requires n ancilla to generate, and the adder can be diagonalized by the QFT to be generated without any ancilla. The largest circuits are therefore the ones which contain two diagonal gates. These circuits require n ancilla for each diagonal gate. In addition to the n *Ansatz* qubits and the control qubit, these circuits therefore have a total of $3n + 1$ qubits. The diagonal gate contains $O(n)$ gates and the adder contains $O(n\ln(n))$ gates, due to the QFT. Therefore, the largest circuits contain $O(n)$ qubits and $O(n\ln(n))$ gates, not accounting for the complexity of the *Ansatz* or *Ansätze*. In general, this will be true when applying the method to any equation. Our *Ansätze* as described in the preceding section have at most $O(n^2)$ gates and $O(n^2)$ parameters, giving a total circuit complexity of $O(n^2)$ gates. As in Eq. (2), the BSE demands a Dirichlet condition on the right boundary, while the adder imposes periodic boundary conditions. However, as previously mentioned, the Dirichlet boundary condition can be approximated by a periodic boundary condition by doubling the length of the domain, reflecting the terminal condition across the right boundary, and time evolving the whole reflected function. This means that for n *Ansatz* qubits one gets 2^{n-1} grid points representing the domain of interest, as one qubit must be used for the reflection. Note that this method does not perfectly capture the Dirichlet boundary conditions and is a source of error accruing after several time steps. Recently, methods have been proposed in the literature to impose Dirichlet boundary conditions exactly by calculating a small number of additional expectation values, as in Ref. [8], though we were not aware of this method when running our simulations.

B. Results

For our simulations, we consider a European put-type option boundary value, where the option price at the maturity time T, which we choose to be $T = 3$ years, is given as $V(S, T) = \max(K - S, 0)$. As in Ref. [23], we choose a domain of $S \in [\frac{1}{135}, 135]$, giving a transformed domain of $x \in [-\ln(135), \ln(135)]$, with parameter values such as $K = 50$, $r = 0.3$, and $\sigma_0^2 = 0.04$. We also set the nonlinear parameter $a = 0.1$. We begin with the transformed European call option $V(x, T) = \max(K - e^x, 0)$ and time evolve backward from $t = T$ to $t = 0$, with ten time steps of $\tau = -0.3$. We simulate eight noiseless *Ansatz* qubits, though one of them is used to reflect our function to deal with the Dirichlet boundary conditions (see Ref. [23]), giving us a reflected transformed domain of length $L = 4\ln 135$. Our target function is represented by a grid mesh of $2^{(8-1)} = 128$ points. We use the ZGR QFT *Ansatz* to represent $|V\rangle$ with $m = 6$, giving 129 total parameters, and the universal layered *Ansatz* to represent $|\chi\rangle$ with $d = 6$, giving 253 total parameters. For comparison, we also solved the linear and nonlinear BSEs with the same parameters using a classical numerical solver on 128 grid points, which we constructed using the same semi-implicit numerical scheme used for our quantum solver and a classical matrix inversion algorithm. In Fig. 4 we employ the classical solver using the exact terminal condition for V and the exact Dirichlet boundary conditions to get the ideal results for "true" time evolution. In Fig. 5 we employ the classical solver using the imperfect Fourier approximation of the terminal condition and the imperfect (periodic) boundary condition to isolate the errors which accrue as a result of time evolution. As can be seen from the figures, the behavior of the time evolution is the same in the classical and the quantum simulations, showing that our method effectively captures the nonlinear dynamics of the equation in the noiseless setting. We find a total error of 2.36% at the end of the time evolution as compared with the classical solution, which mostly stems from the approximate boundary condition; there is around 0.13% error which arises before the time evolution due to the imperfect *Ansatz* representation of the terminal condition.

VI. SOLVING THE 2D LINEAR BLACK-SCHOLES EQUATION

A. The 2D extension

Consider the 2D linear Black-Scholes equation

$$\frac{\partial V}{\partial t} = rV + \left(\frac{1}{2}\sigma_x^2 - r\right)\frac{\partial V}{\partial x} + \left(\frac{1}{2}\sigma_y^2 - r\right)\frac{\partial V}{\partial y} - \frac{1}{2}\sigma_x^2\frac{\partial^2 V}{\partial x^2} - \frac{1}{2}\sigma_y^2\frac{\partial^2 V}{\partial y^2} - \frac{1}{2}\rho\sigma_x\sigma_y\frac{\partial^2 V}{\partial x\partial y},$$
$$V(x, y, t = T) = \max(K - w_x e^x - w_y e^y),$$
$$V(x \to \infty, y, t) = 0,$$
$$V(x, y \to \infty, t) = 0, \tag{6}$$

where σ_x and σ_y are the (constant) volatilities of the asset prices S_1 and S_2, respectively, ρ is the cross correlation of S_1 and S_2, and w_x and w_y are constant positive weights. We have made the substitutions $x = \ln(S_1)$ and $y = \ln(S_2)$. Since this equation is linear, we choose the (fully implicit) backward Euler method for its numerical stability.

In order to generalize the method of Sec. II to two dimensions, we must first decide how to encode our solution state as a quantum state. We wish to partition our 2D domain into a grid of $2^{n_x} \times 2^{n_y}$ grid points, where we designate the first n_x qubits to the x dimension and the last n_y qubits to the y dimension, with a total of $n = n_x + n_y$ *Ansatz* qubits. In this form, the solution state takes the form $|u(t)\rangle = \sum_{k=0}^{2^n-1} u(x_{k_x}, y_{k_y}, t)|k\rangle$, where $k_y = k \bmod 2^{n_y}$ spans $[0, 2^{n_y} - 1]$ and $k_x = \lfloor \frac{k}{2^{n_y}} \rfloor$ spans $[0, 2^{n_x} - 1]$, as expected. In other words, the solution state has amplitudes corresponding to the flattened 2D array of $u(x, y, t)$ evaluated at each point in the discretized domain, where the x index is treated as the

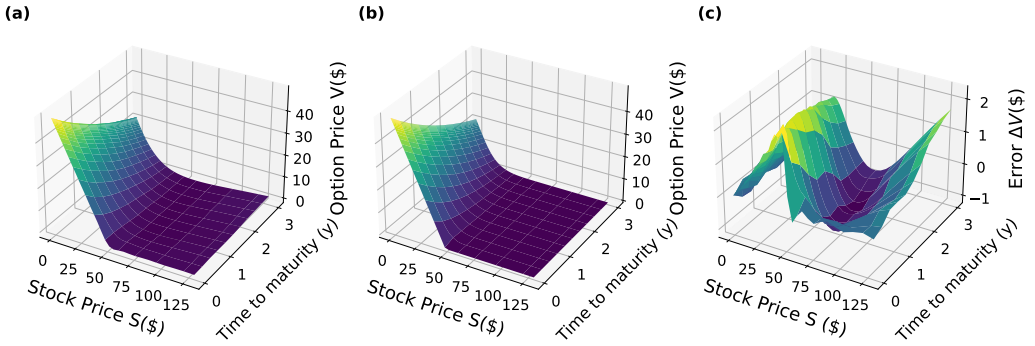

FIG. 4. 1D nonlinear Black-Scholes equation time evolution. The (a) quantum ($n = 8$) and (b) classical solutions closely align, with a final relative error of 2.36% after the last time step, calculated as $(V_{\text{quantum}} - V_{\text{classical}})/||V_{\text{classical}}||$, giving an error per time step of 0.236%. At the boundaries, most of the error accumulates due to the Gibbs phenomenon and the imperfect approximation of the Dirichlet boundary condition. There is also relatively large error due to the Gibbs phenomenon near $S = 50$, where the terminal condition has a discontinuous derivative. (c) Absolute error.

more significant index (one could just as well choose the y index to be the more significant index, though we choose the former convention here). As before, we wish to represent $|u\rangle$ variationally so that $|u(t)\rangle = \lambda_0 \hat{U}(\boldsymbol{\lambda})|0\rangle = \lambda_0|\psi\rangle$. The 2D diagonal operator (which we do not need for the 2D linear Black-Scholes equation) is the same as the 1D diagonal operator. To convert the partial derivatives $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ and any higher-order spatial derivatives into quantum operators, we need only introduce the so-called x adder and y adders

$$\hat{A}_x = \hat{A}_{n_x} \otimes \mathbb{1}^{\otimes n_y}, \quad \hat{A}_y = \mathbb{1}^{\otimes n_x} \otimes \hat{A}_{n_y},$$

where $\hat{A}_{n_x}$ denotes the previously discussed adder operator acting on n_x qubits. In other words, the x and y adders are just the adder operator acting on the x qubits and the y qubits, respectively. From these, the partial derivatives can be decomposed in terms of these adders completely analogously to the 1D case, and the cost function can be derived in the same way as well, which is shown in Appendix D. The circuit complexities are the same as in the 1D case. This flattened array method of encoding 2D functions and constructing derivatives from tensor products of adders with the identity generalizes straightforwardly to arbitrary dimensions, and even vector or tensor fields, allowing this quantum PDE-solving method to be applied to even the Navier-Stokes equations, demonstrating the immense generality of this method. To solve the 2D BSE, we choose our *Ansatz* to be the straightforward generalization of the (complex) ZGR QFT *Ansatz* to two dimensions (see Appendix C of [24]), which utilizes a finite 2D Fourier series in the same way that the 1D version utilizes a finite Fourier series. Here we do not need to restrict ourselves to a real-valued Fourier series because our PDE of interest is linear, and thus we can just take the real part of our solution at the end. Using $n_x = n_y = 6$ qubits for the x and y dimensions and $m_x = m_y = 3$ qubits for the Fourier coefficients in each dimension, the *Ansatz* has a total of $2^{m_x+m_y+3} = 512$ parameters.

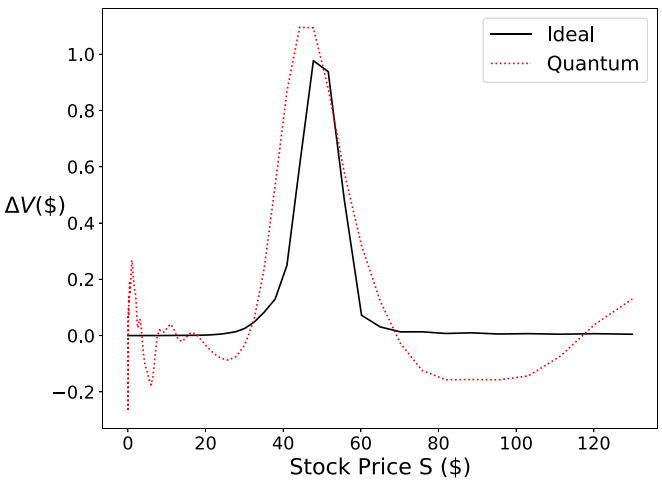

FIG. 5. Effect of nonlinear volatility σ^2 in the Black-Scholes equation (first time step, $n = 8$ qubits). The difference between the solutions to the nonlinear and linear Black-Scholes equations $V_{\text{nonlinear}} - V_{\text{linear}}$ is plotted. The key feature of the nonlinear volatility (as described in Ref. [30]) is that it causes the price of the option to sharply increase compared to the linear model, peaked asymmetrically near the discontinuity in the terminal condition (around $S = 50$). As evident from the plot, this qualitative behavior is reflected in the quantum solution, which differs from the ideal solution mostly due to edge effects, an artifact of the Fourier-based nature of the *Ansatz* and the Gibbs phenomenon. The rest of the error stems from imperfect optimization. The ideal plot was calculated by classically solving numerically for the first time step of the nonlinear Black-Scholes time evolution where the terminal condition is given by the (imperfect) ZGR QFT representation of the put option, and we use the reflected periodic boundary instead of Dirichlet boundary conditions so that errors associated with the optimization are not confounded with errors associated with the readin of the terminal condition or the imperfect boundary condition.

B. Results

For our numerical simulations, we let $r = 0.3$, $\sigma_x = \sigma_y = 0.2$, $w_1 = w_2 = 1$, and $\rho = 0$ and once again time evolve backward from $t = 3$ to $t = 0$ in ten time steps of -0.3. We set $n_x = n_y = 6$, but one of each is used to reflect the function to deal with boundary conditions as in the 1D case, giving us a discretized domain of $32 \times 32 = 1024$ grid points. We found a final error of 2.60%, and the results are depicted in Fig. 6. Like the 1D case, there is a small amount of error

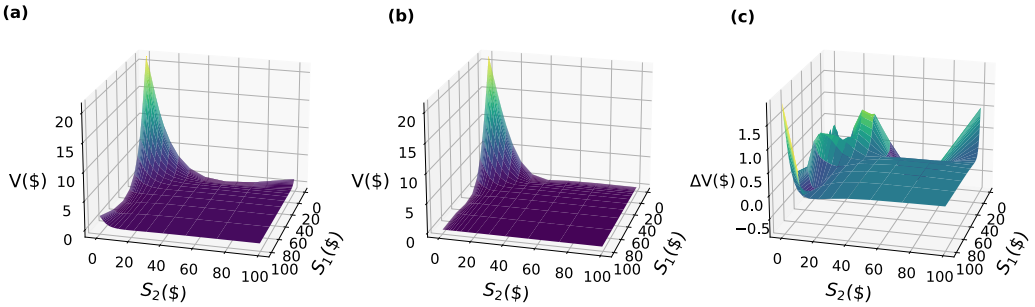

FIG. 6. 2D linear Black-Scholes equation solution (final time step). (a) The quantum solution has an error of 2.60% compared with the classical, giving an error per time step of 0.260% ($n_x = 6$ and $n_y = 6$). As in the 1D case, most of the error is near the boundaries due to the approximate boundary conditions. (b) The classical solution is calculated using the backward Euler scheme with the exact initial condition. (c) Absolute error.

associated with readin, and most of the error comes from the approximation of the boundary conditions.

VII. SOLVING THE BUCKMASTER EQUATION

Consider the Buckmaster equation

$$\frac{\partial f}{\partial t} = \frac{\partial^2}{\partial x^2}(f^4) + \alpha \frac{\partial}{\partial x}(f^3), \tag{7}$$

where α is a known parameter. This equation governs the time evolution of the surface of a viscous liquid, a problem of relevance to fluid dynamics and for which it is difficult to find analytical solutions. Unlike the nonlinear Black-Scholes equation, the time evolution of the Buckmaster equation is entirely nonlinear, in that it cannot be approximated as a perturbation of a linear equation. For this reason we choose to verify the quantum PDE method on this problem. For simplicity, we consider a fully explicit forward Euler scheme, which requires fewer expectation values, allowing us to more quickly time evolve our solution state and verify the robustness of the algorithm in the many-time-step regime. The procedure for deriving the cost function for the Buckmaster equation is the same as described in Sec. II and the example given in Sec. V A, so we omit discussion of the cost function, though it can be found in Appendix E. For our simulations we let $\alpha = 1$ and consider a domain of $x \in [-\pi, \pi]$ ($L = 2\pi$) with a sinusoidal initial condition $f(x, t = 0) = \frac{1}{3}[2 - \sin(x)]$ and periodic boundary conditions so that we need not use the reflection trick to deal with the boundary conditions. We time evolve the equation for $n = 5$ qubits and therefore a mesh of $2^5 = 32$ grid points from $t = 0$ to $t = 2$ in 250 time steps of size 0.008, using the real ZGR QFT *Ansatz* with $m = 3$. The results are shown in Fig. 7, in which the quantum solution completely captures the nonlinear time dynamics of the equation.

VIII. SOLVING THE DETERMINISTIC KARDAR-PARISI-ZHANG EQUATION

Finally, consider the deterministic Kardar-Parisi-Zhang (KPZ) equation, used to model deterministic surface growth,

$$\frac{\partial f}{\partial t} = \alpha \frac{\partial^2 f}{\partial x^2} + \beta \left(\frac{\partial f}{\partial x} \right)^2, \tag{8}$$

where α and β are given. The deterministic KPZ equation describes the asymptotic behavior of a large class of deterministic surface growth models [31]. We again use a fully explicit forward Euler scheme. The nonlinear term in the KPZ equation requires one auxiliary quantum state to be trained, as in the case of the nonlinear Black-Scholes equation, and a full description of the cost functions can be found in Appendix F. For our simulations we let $\alpha = 0.5$ and $\beta = 0.5$ and consider a domain of $x \in [-2.5, 2.5]$ ($L = 5$) with a Gaussian initial condition $f(x, t = 0) = e^{-x^2}$ and periodic boundary conditions so that we need not use the reflection trick to deal with the boundary conditions. We time evolve the equation for $n = 5$ qubits and therefore a mesh of $2^5 = 32$ grid points from $t = 0$ to $t = 4$ in 200 time steps of size 0.02, using the ZGR QFT *Ansatz* with $m = 3$. The results are shown in Fig. 8, and once again the quantum algorithm performs well.

IX. EXECUTION ON HARDWARE

An important consideration for the variational-PDE-solving method discussed in this paper is the precision with which the expectation values appearing in the cost functions must be estimated, which in turn impacts the computational cost of computing the cost function.[2] The cost functions contain expectation values [whose magnitudes are $O(1)$] being multiplied by prefactors that are exponential in n; these prefactors arise from the finite-difference methods used to derive the cost functions. This suggests that one may have to estimate the expectation values to a precision that increases exponentially with n in order to get a sufficiently accurate estimate of the cost function. Suppose we want to estimate the cost function with a maximum error of ϵ. To do this, we need to estimate certain expectation values with an error of less than $O(\epsilon/e^n)$. The error in the estimation of $\langle \mathcal{O} \rangle$ using M shots of the Hadamard test is given by $\mathcal{E}_M(\mathcal{O}) \sim \frac{\sqrt{\langle \mathcal{O}^2 \rangle - \langle \mathcal{O} \rangle^2}}{\sqrt{M}}$. This means that to get $\mathcal{E}_M = O(\epsilon/e^n)$, M must scale as $O(e^{2n}/\epsilon^2)$. This is characterized numerically in Fig. 9, in which we plot how the estimated value of the cost function for the Buckmaster equation converges to its true value as a function of

[2]Reference [32] presents a related study of the cost for solving linear PDEs using the variational-quantum-linear-solver approach.

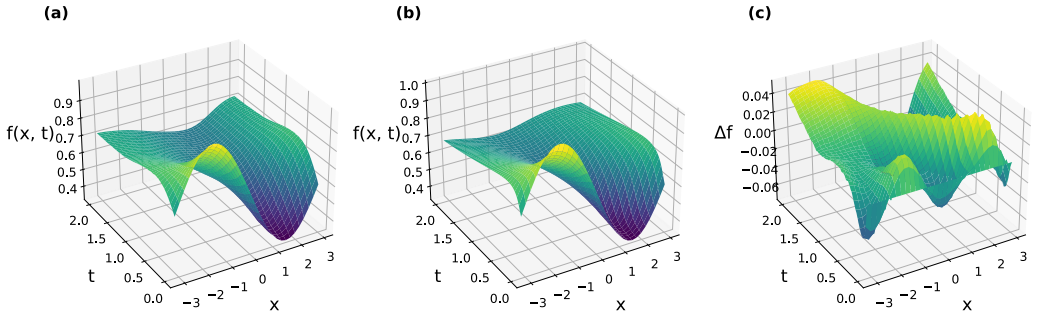

FIG. 7. Buckmaster equation time evolution. (a) The quantum solution has an error of 6.54% compared with the classical, giving an error per time step of 0.0262% ($n = 5$). The error is large near the boundaries despite the periodic boundary condition because any errors that accrue naturally near the boundaries as a result of the imperfect initial condition or optimization will grow larger over time due to the Gibbs phenomenon. (b) The classical solution is calculated using the forward Euler scheme with the exact initial condition. (c) Absolute error.

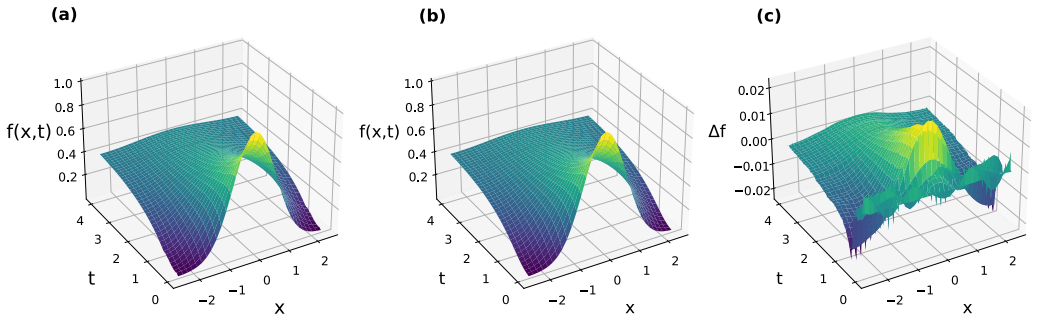

FIG. 8. KPZ equation time evolution. (a) The quantum solution has an error of 0.415% compared with the classical, giving an error per time step of 0.002 08% ($n = 5$). (b) The classical solution is calculated using the forward Euler scheme with the exact initial condition. (c) Absolute error.

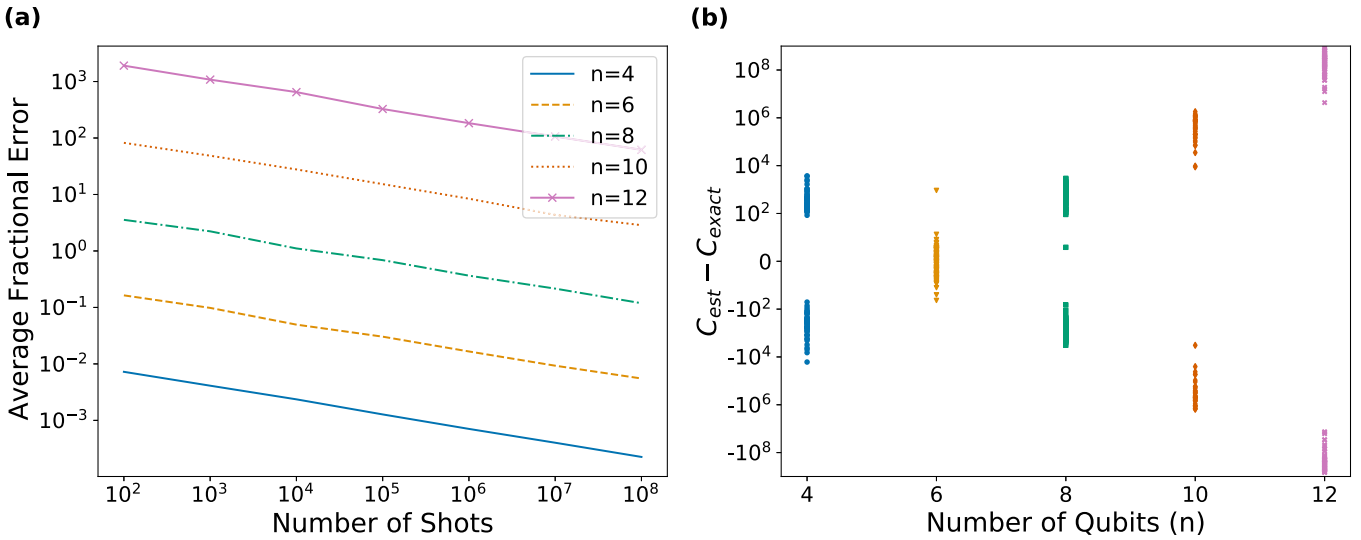

FIG. 9. Analysis of Buckmaster cost-function estimation. We estimated the cost function by estimating each relevant expectation value using a binomial experiment with the given number of shots, effectively replicating what the output from a noiseless quantum computer would be. We repeat this experiment 100 times to get 100 different estimates of the cost function. (a) Average fractional error, defined as the average of $|C_{\text{est}} - C_{\text{exact}}|/|C_{\text{exact}}|$ over all 100 trials as a function of shots. The slope of each line in the log-log plot is very close to -0.5, while the y intercept is exponentially increasing with the number of qubits, exhibiting a dependence of error proportional to $e^n(\text{shots})^{-1/2}$. This suggests that to achieve an average error of ϵ in the estimation of the cost function, one needs $O(\frac{e^n}{\epsilon^2})$ shots (ignoring the effect of noise from hardware imperfections). (b) Scatter plot of all 100 samples of the cost function with 10^8 shots for each number of qubits. The estimations of the cost function tend to exponentially deviate from the true value as the number of qubits increases.

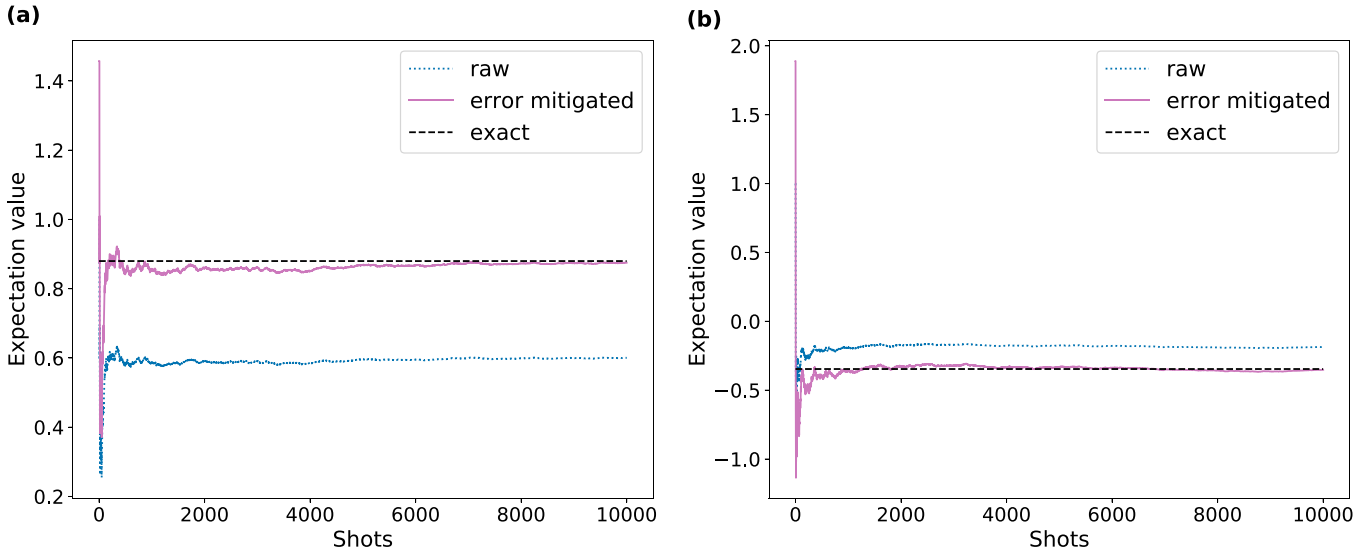

FIG. 10. Estimation of expectation values on a trapped-ion processor: (a) $\langle\psi|\hat{A}|\psi\rangle$ and (b) $\langle\psi|\hat{D}_\phi|\psi\rangle$. The data were generated by runs on an 11-qubit trapped-ion device offered by IonQ [22]. The first expectation value was calculated with two *Ansatz* qubits using the ULA. The second was calculated with one *Ansatz* qubit, using a single R_y gate as the *Ansatz*. The circuits are shown in Appendix G. To mitigate the errors in the estimation of the expectation values, we ran each circuit with one set of parameters for which the expectation value is known, in order to estimate the damping factor [33] of the circuits. We then ran the circuits with the true set of parameters and divided by the damping factor to recover a better estimate of the true expectation value. Using this method, we found errors of 0.552% and 1.56% in the estimation of the first and second expectation values, respectively.

the number of qubits and the number of shots, displaying the scaling behavior derived above. This issue is relevant even for relatively small problem sizes; for example, when solving the Buckmaster equation with $n = 6$ qubits, expectation values must be sampled with 10^8 shots to achieve around 1% error in the estimation of the cost function, which we found to be roughly the minimum required accuracy for the optimization procedure to converge to the solution of the Buckmaster equation. The required number of shots to achieve acceptable accuracy constrains what can be demonstrated on real hardware (due to the costs in both money and time of hardware shots), and with the scaling of the present approach for formulating and estimating cost functions, ultimately presents an obstruction to achieving quantum advantage. This problem appears to persist to higher-dimensional problems. For example, in the 2D linear BSE cost function (Appendix D), there are terms with prefactors that scale as $O(e^{n_x+n_y}) = O(e^n)$, suggesting that even for high-dimensional problems with a low number of qubits per dimension, the exponential scaling in the total number of qubits is still present.

We sought to demonstrate that the cost functions we derived to solve the nonlinear Black-Scholes equation can be computed on prototype quantum hardware. To this end, we estimated two expectation values appearing in the cost function on a trapped-ion quantum processor from IonQ [22], albeit in the admittedly toy case of having just two or fewer *Ansatz* qubits. In particular, using the Hadamard test, we estimated one linear expectation value $\langle\psi|\hat{A}|\psi\rangle$ with two *Ansatz* qubits and one nonlinear expectation value $\langle\psi|\hat{D}_\phi|\psi\rangle$ with one *Ansatz* qubit, as a function of number of measurement shots. Both circuits had a total of three qubits. The results are shown in Fig. 10; we were able to estimate both expectation values to within an accuracy of approximately 1% using the error-mitigation technique described in Ref. [33]. We were unable to estimate expectation values like $\langle\psi|\tilde{\psi}\rangle$ due to the controlled *Ansätze* appearing in the circuits for computing them, which require many more two-qubit gates than we could reliably run on the hardware available to us. This is not a situation specific to the nonlinear Black-Scholes equation: In general, controlled *Ansätze* will appear in the circuits for computing some of the terms of the cost function for a nonlinear PDE, yielding rather deep circuits when more than a few qubits are used (unless the *Ansatz* is exceptionally simple, in which case it will probably also be insufficiently expressive).

X. CONCLUSION AND OUTLOOK

In this paper we presented an extension of the variational-PDE-solving algorithm introduced in Ref. [1] and demonstrated how it could be applied to a wide class of nonlinear and multidimensional PDEs that are first order in time. We demonstrated that the algorithm is amenable to various explicit and semi-implicit Runge-Kutta numerical schemes, applying both the forward Euler and backward Euler discretizations in simulation. We used two *Ansätze*: a Fourier *Ansatz* [the real-valued ZGR QFT *Ansatz* (Appendix A)] and a hardware-efficient *Ansatz* [the universal layered *Ansatz* (Appendix B)], which is universal if sufficiently many layers are used. We benchmarked these *Ansätze* for expressibility and trainability using several popular optimization algorithms for minimizing the cost functions we derived. While both *Ansätze* are fully general, the ZGR QFT is more efficient for representing smooth functions, while we found that the ULA can be more efficient for representing certain "jagged" functions. We then, in simulation, applied the quantum PDE-solving method to solving a nonlinear Black-Scholes equation, the double-asset

linear Black-Scholes equation, the Buckmaster equation, and the deterministic Kardar-Parisi-Zhang equation. We were able to achieve accurate solutions for up to $n = 12$ *Ansatz* qubits in noiseless simulations and demonstrate proof-of-concept results on a trapped-ion quantum computer with up to $n = 2$ *Ansatz* qubits. We did not apply the quantum PDE-solving method to an equation which is both nonlinear and multidimensional due to the difficulty of simulating a large number of qubits (e.g., the 2D nonlinear BSE with $6 + 6$ *Ansatz* qubits would require simulating circuits with over 36 qubits), though in principle the method is amenable to nonlinear multidimensional equations. During our investigations we encountered several challenges that appear necessary to address before one can hope to solve PDEs using quantum variational approaches in a regime that might give a quantum advantage over classical PDE solvers.

(i) Sufficiently accurate estimation of the value of the cost function. The cost functions used in variational PDE solving involve expectation values [whose magnitudes are $O(1)$] multiplied by prefactors that are exponential in the number of qubits n. This suggests that the expectation values must be estimated with an exponentially increasing precision (in n) in order to calculate the cost functions accurately enough that the optimizer can steer towards good solutions rather than poor ones. If one uses a method (such as the Hadamard test) requiring exponentially many circuit evaluations (shots) to achieve exponential precision, the approach both becomes impractical to test with moderately sized ($n > 10$) *Ansätze* on current hardware and has scaling that likely rules out a quantum advantage.

(ii) Design of Ansätze that are sufficiently expressive, use few parameters, and do not render the algorithm classical simulable. The ideal variational *Ansatz* can probe the region of Hilbert space where the problem solutions lie with as few parameters as possible (e.g., needing only n parameters would be very favorable) while also remaining classically intractable, in the sense that there should not be a polynomial-time classical algorithm to simulate circuits involving the *Ansatz*. The desirable properties which make the ZGR QFT good at representing smooth functions also make it classically tractable. It is an open problem to identify an *Ansatz* that has the desirable properties (nonchaotic, low number of parameters, adequately expressive, and trainable) while remaining classically intractable, or if none exists to elucidate the tradeoffs that the best possible *Ansätze* have.

(iii) Deep circuits. The circuits appearing in the variational-PDE-solving algorithm involve controlled applications of the *Ansatz* unitary. If the *Ansatz* uses many two-qubit gates, then the controlled *Ansatz* will be difficult to execute on near-term hardware. In our demonstrations on IonQ's 11-qubit device, we found that the full algorithm for solving simple nonlinear PDEs is not practical to run on current hardware due to the circuit depths involved, even with as few as two *Ansatz* qubits.

These findings are consistent with the analysis in Ref. [34] of quantum variational solving of a linear PDE (the heat equation). We hope that through alternative constructions of cost functions or alternative methods of evaluating the cost functions, as well as the design of shallow *Ansätze* tailored to specific PDEs, progress can be made toward practical nonlinear PDE solving using variational quantum algorithms.

Finally, we acknowledge the recent work in Ref. [35] in which the authors propose a variational PDE time-evolution scheme in which the entire time evolution is captured in one optimization procedure by utilizing an additional qubit register corresponding to the time dimension. We find the authors' ideas to be interesting and promising. They argue that the temporal discretization can be chosen such that the exponential factors associated with the spatial and temporal discretizations cancel out, circumventing the issues described in Sec. IX. However, such a scheme may prevent the use of clever *Ansätze* such as the ZGR QFT, which take advantage of the spatial structure of candidate solutions, making cost function optimization much more difficult.

The code used to run the numerical simulations and the hardware runs on IonQ and to generate the figures in this paper, as well as the data from our simulations and hardware demonstrations, is open source and available from [36]. This includes implementations of the ZGR QFT *Ansatz*, the universal layered *Ansatz*, and the cost functions for the PDEs considered in the paper.

P.L.M., A.S., and T.W.W. conceived the project. A.S. derived the cost functions, coded the time-evolution circuits, ran the numerical simulations and hardware demonstrations, and analyzed the data. Y.L. and M.M. helped derive the cost functions, Y.L. helped debug the code for the circuits, and M.M. helped run the simulations. M.M. and T.W.W. developed and provided the code for the 1D and 2D ZGR QFT *Ansätze* and wrote Appendix A. A.S. and P.L.M. wrote the main manuscript, which was revised by all authors. P.L.M. supervised the project.

APPENDIX A: REAL-VALUED ZGR QFT *ANSATZ*

To construct the real-valued ZGR QFT, we wish to parametrize a truncated real-valued Fourier series. Consider the partial Fourier series

$$f(x) = \sum_{k=0}^{M-1} a_k \cos(2\pi kx) + \sum_{k=1}^{M-1} b_k \sin(2\pi kx), \tag{A1}$$

where $M = 2^m$. Note that we can equivalently write the above function as

$$f(x) = \sum_{k=0}^{M-1} c_k e^{-i2\pi kx} + \sum_{k=1}^{M-1} c_k^* e^{i2\pi kx}, \tag{A2}$$

where $c_0 = a_0$ and $c_k = \frac{1}{2}(a_k + ib_k)$ for $k \neq 0$.

In accordance with the real-valued discrete Fourier transform, the n-qubit state $|f\rangle$ can be written as

$$|f\rangle = \sqrt{N}\mathrm{QFT}^{-1}|\psi_f\rangle, \tag{A3}$$

where $N = 2^n$ and the (unnormalized) state $|\psi_f\rangle$ is given by

$$|\psi_f\rangle = \sum_{k=0}^{M-1} c_k|k\rangle + \sum_{k=1}^{M-1} c_k^*|N-k\rangle. \tag{A4}$$

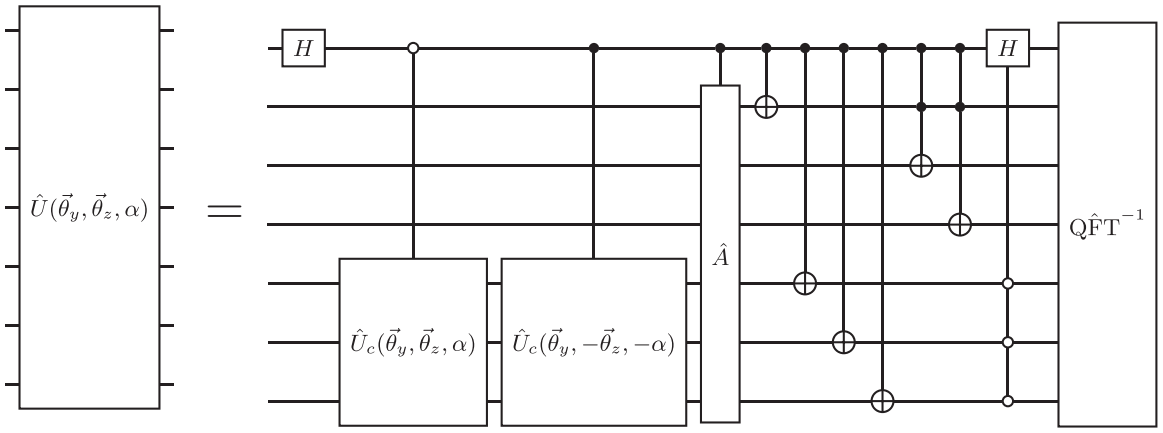

FIG. 11. Real-valued ZGR QFT *Ansatz* circuit diagram, displayed here with $n = 7$ and $m = 3$. The generalization for arbitrary m and $n \geqslant m + 2$ is straightforward. Note that the lines through the adder gate mean that the adder is not acting on the corresponding qubits. One can straightforwardly verify that the gates before the QFT prepare a state proportional to $|\psi_f\rangle$.

1. Quantum circuit to prepare $|\psi_f\rangle$

Consider the following m-qubit quantum states which encode the Fourier coefficients:

$$|c\rangle = \frac{1}{\mathcal{N}_c}\left(\frac{c_0}{\sqrt{2}}|0\rangle + \sum_{k=1}^{M-1} c_k|k\rangle\right), \tag{A5}$$

$$|c^*\rangle = \frac{1}{\mathcal{N}_c}\left(\frac{c_0}{\sqrt{2}}|0\rangle + \sum_{k=1}^{M-1} c_k^*|k\rangle\right). \tag{A6}$$

Here the normalization constant is

$$(\mathcal{N}_c)^2 = \frac{c_0^2}{2} + \sum_{k=1}^{M-1} |c_k|^2. \tag{A7}$$

The states $|c\rangle$ and $|c^*\rangle$ can efficiently be prepared using the ZGR unitary, which we denote by $\hat{U}_c(\vec{\theta}_y, \vec{\theta}_z, \alpha)$ (shown in Fig. 1(b) of [24]). These ZGR unitaries consist of a cascade of uniformly controlled rotations, as described in [25]. Here $\vec{\theta}_{y,z}$ contains all of the rotation parameters for the $\hat{R}_{y,z}$ gates, while α is a global phase. It is straightforward to see from the formulas of ZGR parameters in [25] that the ZGR parameters to prepare the conjugate state $|c^*\rangle$ are the same as the parameters for $|c\rangle$, but with $\vec{\theta}_z$ and α negated:

$$|c\rangle = \hat{U}_c(\vec{\theta}_y, \vec{\theta}_z, \alpha)|0\rangle^{\otimes m}, \tag{A8}$$

$$|c^*\rangle = \hat{U}_c(\vec{\theta}_y, -\vec{\theta}_z, -\alpha)|0\rangle^{\otimes m}. \tag{A9}$$

Given the unitary $\hat{U}_c$, the circuit to produce a state proportional to $|f\rangle$ is given in Fig. 11. To convert this into a variational *Ansatz*, we simply promote the variables $(\vec{\theta}_y, \vec{\theta}_z, \alpha)$ to variational parameters.

2. Adder operator

The quantum circuit that we have proposed contains the $(m + 1)$-qubit adder operator. One way to implement this adder operator is using $O(m)$ CNOT gates and Toffoli gates, but this implementation requires $m - 1$ ancilla qubits. Another way to implement this is using the phase shift gates and $(m + 1)$-qubit QFT and its inverse. This method does not require ancilla qubits but requires $O(m^2)$ CNOT gates.

In our code we make use of both of these implementations. If $n > 2m$, then we use the implementation of the adder with ancilla qubits and use the remaining $(n - 2 - m)$ qubits as ancilla qubits. On the other hand, if $n \leqslant 2m$, we implement the adder operator using the phase shift and the adder operator.

APPENDIX B: UNIVERSAL LAYERED *ANSATZ*

The universal layered *Ansatz* is shown in Fig. 12.

APPENDIX C: 1D LINEAR AND NONLINEAR BLACK-SCHOLES EQUATION COST FUNCTIONS

We will henceforth denote the parameters of $|\chi\rangle$ by θ_0 and $\boldsymbol{\theta}$, and the parameters of $|V\rangle$ by λ_0 and $\boldsymbol{\lambda}$, such that $|\chi\rangle = \theta_0\hat{U}(\boldsymbol{\theta})|0\rangle = \theta_0|\phi\rangle$ and $|V\rangle = \lambda_0\hat{U}(\boldsymbol{\lambda})|0\rangle = \lambda_0|\psi\rangle$, where we

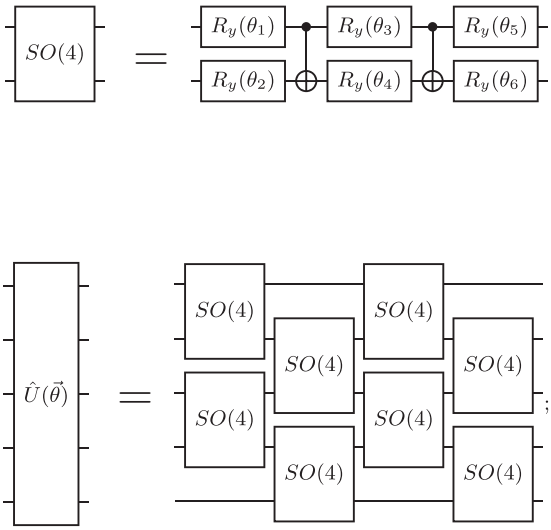

FIG. 12. ULA circuit diagram. Here the ULA with $n = 5$ qubits and depth $d = 2$ is displayed. Each SO(4) gate contains six different parameters, giving a total of $6(n - 1)d$ parameters.

have defined $|\phi\rangle = \hat{U}(\boldsymbol{\theta})|0\rangle$ and $|\psi\rangle = \hat{U}(\boldsymbol{\lambda})|0\rangle$ and we have dropped the explicit dependence on the rotational parameters. Once again denoting the parameters of the previous time step by a tilde, the cost function for $|\chi\rangle$ is

$$C_\chi = \left\| |\chi\rangle - \left(\frac{\hat{\partial}^2}{\partial x^2} - \frac{\hat{\partial}}{\partial x}\right)|\tilde{V}\rangle \right\|^2 = \left\langle \chi - \left(\frac{\hat{\partial}^2}{\partial x^2} - \frac{\hat{\partial}}{\partial x}\right)\tilde{V} \middle| \chi - \left(\frac{\hat{\partial}^2}{\partial x^2} - \frac{\hat{\partial}}{\partial x}\right)\tilde{V} \right\rangle, \quad \text{(C1)}$$

where we are once again using the naive distance measure between $|\chi\rangle$ and the desired state. After making the substitutions $\frac{\hat{\partial}^2}{\partial x^2} = \frac{4^n}{L^2}(\hat{A} + \hat{A}^\dagger - 2\mathbb{1})$, $\frac{\hat{\partial}}{\partial x} = \frac{2^{n-1}}{L}(\hat{A} - \hat{A}^\dagger)$, $|\chi\rangle = \theta_0|\phi\rangle$, and $|\tilde{V}\rangle = \tilde{\lambda}_0|\tilde{\psi}\rangle$ and expanding and simplifying, we get

$$C_\chi = \theta_0^2 + 2\theta_0\tilde{\lambda}_0 \mathrm{Re}\left[\left(\frac{2^{n-1}}{L} - \frac{4^n}{L^2}\right)\langle\phi|\hat{A}|\tilde{\psi}\rangle - \left(\frac{2^{n-1}}{L} + \frac{4^n}{L^2}\right)\langle\phi|\hat{A}^\dagger|\tilde{\psi}\rangle + 2\frac{4^n}{L^2}\langle\phi|\tilde{\psi}\rangle\right] + \text{const.} \quad \text{(C2)}$$

The three expectations in C_χ can be calculated efficiently using the Hadamard test. Now that we have the state $|\chi\rangle$, we can use the diagonal operator, as described in Ref. [1], to multiply by the nonlinear volatility. Using this operator, we can generate the quantum version of the differential operator $\hat{O}$,

$$\begin{aligned}\hat{O} &= r + \left(\frac{\sigma^2}{2} - r\right)\frac{\partial}{\partial x} + \frac{\sigma^2}{2}\frac{\partial^2}{\partial x^2} \\ &= r\mathbb{1} + \left(\frac{\sigma_0^2}{2}(1 + e^{r(T-t)}a^2\hat{D}_\chi) - r\right)\frac{\hat{\partial}}{\partial x} \\ &\quad - \frac{\sigma_0^2}{2}(1 + e^{r(T-t)}a^2\hat{D}_\chi)\frac{\hat{\partial}^2}{\partial x^2}. \end{aligned} \quad \text{(C3)}$$

Substituting into Eq. (1), the cost function for $|V\rangle$ takes the form

$$\begin{aligned} C_V = \Bigg\langle \Bigg(&(1 - r\tau)\mathbb{1} - \left\{\frac{1}{2}\left[\sigma_0^2(1 + e^{r(T-t)}a^2\hat{D}_\chi)\right] - r\right\}\tau\frac{\hat{\partial}}{\partial x} \\ &+ \frac{1}{2}\left[\sigma_0^2(1 + e^{r(T-t)}a^2\hat{D}_\chi)\right]\tau\frac{\hat{\partial}^2}{\partial x^2}\Bigg)V \\ &- \tilde{V}\Bigg| \Bigg((1 - r\tau)\mathbb{1} - \left\{\frac{1}{2}\left[\sigma_0^2(1 + e^{r(T-t)}a^2\hat{D}_\chi)\right] - r\right\}\tau\frac{\hat{\partial}}{\partial x} \\ &+ \frac{1}{2}\left[\sigma_0^2(1 + e^{r(T-t)}a^2\hat{D}_\chi)\right]\tau\frac{\hat{\partial}^2}{\partial x^2}\Bigg)V - \tilde{V}\Bigg\rangle. \end{aligned} \quad \text{(C4)}$$

Noting that $\hat{D}_\chi = \theta_0\hat{D}_\phi$, we get

$$C_V = \left\langle \left(\alpha\mathbb{1} - \beta\frac{\hat{\partial}}{\partial x} + \gamma\frac{\hat{\partial}^2}{\partial x^2} - \epsilon\hat{D}_\phi\frac{\hat{\partial}}{\partial x} + \epsilon\hat{D}_\phi\frac{\hat{\partial}^2}{\partial x^2}\right)V - \tilde{V} \middle| \left(\alpha\mathbb{1} - \beta\frac{\hat{\partial}}{\partial x} + \gamma\frac{\hat{\partial}^2}{\partial x^2} - \epsilon\hat{D}_\phi\frac{\hat{\partial}}{\partial x} + \epsilon\hat{D}_\phi\frac{\hat{\partial}^2}{\partial x^2}\right)V - \tilde{V} \right\rangle, \quad \text{(C5)}$$

where we have defined the constants $\alpha = 1 - r\tau$, $\beta = (\frac{1}{2}\sigma_0^2 - r)\tau$, $\gamma = \frac{1}{2}\sigma_0^2\tau$, and $\epsilon = \frac{1}{2}\sigma_0^2 e^{r(T-t)}a^2\theta_0\tau$. This cost function can be factored as

$$\begin{aligned} C_V = &\left\langle \left(\alpha\mathbb{1} - \beta\frac{\hat{\partial}}{\partial x} + \gamma\frac{\hat{\partial}^2}{\partial x^2}\right)V - \tilde{V} \middle| \left(\alpha\mathbb{1} - \beta\frac{\hat{\partial}}{\partial x} + \gamma\frac{\hat{\partial}^2}{\partial x^2}\right)V - \tilde{V} \right\rangle \\ &+ 2\,\mathrm{Re}\left[\left\langle \left(\alpha\mathbb{1} - \beta\frac{\hat{\partial}}{\partial x} + \gamma\frac{\hat{\partial}^2}{\partial x^2}\right)V \middle| \left(-\epsilon\hat{D}_\phi\frac{\hat{\partial}}{\partial x} + \epsilon\hat{D}_\phi\frac{\hat{\partial}^2}{\partial x^2}\right)V \right\rangle\right] \\ &+ \left[\left(-\epsilon\hat{D}_\phi\frac{\hat{\partial}}{\partial x} + \epsilon\hat{D}_\phi\frac{\hat{\partial}^2}{\partial x^2}\right)\middle| \left(-\epsilon\hat{D}_\phi\frac{\hat{\partial}}{\partial x} + \epsilon\hat{D}_\phi\frac{\hat{\partial}^2}{\partial x^2}\right)V\right] - 2\,\mathrm{Re}\left[\left\langle \left(-\epsilon\hat{D}_\phi\frac{\hat{\partial}}{\partial x} + \epsilon\hat{D}_\phi\frac{\hat{\partial}^2}{\partial x^2}\right)V \middle| \tilde{V} \right\rangle\right]. \end{aligned} \quad \text{(C6)}$$

Note that the first term is identically equal to the linear backward Euler Black-Scholes cost function. All three terms can be simplified by substituting the operator identities for the derivatives and expanding the inner product. In the end, we get

$$\begin{aligned} C_{\text{linear}} = &\tilde{\lambda}_0^2 + \lambda_0^2\left(\alpha^2 + \frac{2\beta^2 4^{n-1}}{L^2} + \frac{6\gamma^2 16^n}{L^4} - \frac{4\alpha\gamma 4^n}{L^2}\right) \\ &+ 2\lambda_0\mathrm{Re}\left[\lambda_0\left(\frac{\gamma^2 16^n}{L^4} - \frac{\beta^2 4^{n-1}}{L^2}\right)\langle\psi|\hat{A}^2|\psi\rangle + \lambda_0\left(\frac{2\alpha\gamma 4^n}{L^2} - \frac{4\gamma^2 16^n}{L^4}\right)\langle\psi|\hat{A}|\psi\rangle \right. \\ &\left. - \tilde{\lambda}_0\left(\frac{\beta 2^{n-1}}{L} + \frac{\gamma 4^n}{L^2}\right)\langle\psi|\hat{A}|\tilde{\psi}\rangle + \tilde{\lambda}_0\left(\frac{\beta 2^{n-1}}{L} - \frac{\gamma 4^n}{L^2}\right)\langle\psi|\hat{A}^\dagger|\tilde{\psi}\rangle + \tilde{\lambda}_0\left(\frac{2\gamma 4^n}{L^2} - \alpha\right)\langle\psi|\tilde{\psi}\rangle\right], \end{aligned} \quad \text{(C7)}$$

$$\begin{aligned} C_2 = &2\lambda_0\mathrm{Re}\left[\lambda_0\left(\frac{\alpha\epsilon 4^n}{L^2} - \frac{\alpha\epsilon 2^{n-1}}{L} + \frac{2\gamma\epsilon 2^{n-1}4^n}{L^3} - \frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi\hat{A}|\psi\rangle \right. \\ &\left. + \lambda_0\left(\frac{\alpha\epsilon 4^n}{L^2} + \frac{\alpha\epsilon 2^{n-1}}{L} - \frac{2\gamma\epsilon 2^{n-1}4^n}{L^3} - \frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi\hat{A}^\dagger|\psi\rangle + \lambda_0\left(\frac{4\gamma\epsilon 16^n}{L^4} - \frac{2\alpha\epsilon 4^n}{L^2}\right)\langle\psi|\hat{D}_\phi|\psi\rangle\right. \end{aligned}$$

$$
\begin{aligned}
&+\lambda_0\left(\frac{\beta\epsilon 4^{n-1}}{L^2}-\frac{\beta\epsilon 2^{n-1}4^n}{L^3}+\frac{\gamma\epsilon 16^n}{L^4}-\frac{\gamma\epsilon 2^{n-1}4^n}{L^3}\right)\langle\psi|\hat{A}^\dagger\hat{D}_\phi\hat{A}|\psi\rangle\\
&+\lambda_0\left(-\frac{\beta\epsilon 4^{n-1}}{L^2}-\frac{\beta\epsilon 2^{n-1}4^n}{L^3}+\frac{\gamma\epsilon 16^n}{L^4}+\frac{\gamma\epsilon 2^{n-1}4^n}{L^3}\right)\langle\psi|\hat{A}^\dagger\hat{D}_\phi\hat{A}^\dagger|\psi\rangle\\
&+\lambda_0\left(-\frac{\beta\epsilon 4^{n-1}}{L^2}+\frac{\beta\epsilon 2^{n-1}4^n}{L^3}+\frac{\gamma\epsilon 16^n}{L^4}-\frac{\gamma\epsilon 2^{n-1}4^n}{L^3}\right)\langle\psi|\hat{A}\hat{D}_\phi\hat{A}|\psi\rangle\\
&+\lambda_0\left(\frac{\beta\epsilon 4^{n-1}}{L^2}+\frac{\beta\epsilon 2^{n-1}4^n}{L^3}+\frac{\gamma\epsilon 16^n}{L^4}+\frac{\gamma\epsilon 2^{n-1}4^n}{L^3}\right)\langle\psi|\hat{A}\hat{D}_\phi\hat{A}^\dagger|\psi\rangle\\
&+\lambda_0\left(\frac{2\beta\epsilon 2^{n-1}4^n}{L^3}-\frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{A}^\dagger\hat{D}_\phi|\psi\rangle+\lambda_0\left(-\frac{2\beta\epsilon 2^{n-1}4^n}{L^3}-\frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{A}\hat{D}_\phi|\psi\rangle\Bigg],
\end{aligned}
\tag{C8}
$$

$$
\begin{aligned}
C_3=2\lambda_0\mathrm{Re}\Bigg[&\lambda_0\left(-\frac{\epsilon^2 4^{n-1}}{L^2}+\frac{\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{A}\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}|\psi\rangle+\lambda_0\left(\frac{\epsilon^2 4^{n-1}}{2L^2}+\frac{\epsilon^2 4^n 2^{n-1}}{L^3}+\frac{\epsilon^2 16^n}{2L^4}\right)\langle\psi|\hat{A}\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}^\dagger|\psi\rangle\\
&+\lambda_0\left(\frac{\epsilon^2 4^{n-1}}{2L^2}-\frac{\epsilon^2 4^n 2^{n-1}}{L^3}+\frac{\epsilon^2 16^n}{2L^4}\right)\langle\psi|\hat{A}^\dagger\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}|\psi\rangle+\lambda_0\left(\frac{2\epsilon^2 4^n 2^{n-1}}{L^3}-\frac{2\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}|\psi\rangle\\
&+\lambda_0\left(-\frac{2\epsilon^2 4^n 2^{n-1}}{L^3}-\frac{2\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}^\dagger|\psi\rangle+\lambda_0\left(\frac{2\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi^\dagger\hat{D}_\phi|\psi\rangle\Bigg],
\end{aligned}
\tag{C9}
$$

$$
C_4=2\lambda_0\mathrm{Re}\left[\tilde{\lambda}_0\left(\frac{\epsilon^2 2^{n-1}}{L}-\frac{\epsilon^2 4^n}{L^2}\right)\langle\tilde{\psi}|\hat{D}_\phi\hat{A}|\psi\rangle+\tilde{\lambda}_0\left(-\frac{\epsilon^2 2^{n-1}}{L}-\frac{\epsilon^2 4^n}{L^2}\right)\langle\tilde{\psi}|\hat{D}_\phi\hat{A}^\dagger|\psi\rangle+\tilde{\lambda}_0\left(\frac{2\epsilon^2 4^n}{L^2}\right)\langle\tilde{\psi}|\hat{D}_\phi|\psi\rangle\right],
\tag{C10}
$$

giving us our final nonlinear cost function

$$
\begin{aligned}
C_V=&\tilde{\lambda}_0^2+\lambda_0^2\left(\alpha^2+\frac{2\beta^2 4^{n-1}}{L^2}+\frac{6\gamma^2 16^n}{L^4}-\frac{4\alpha\gamma 4^n}{L^2}\right)+2\lambda_0\mathrm{Re}\Bigg[\lambda_0\left(\frac{\gamma^2 16^n}{L^4}-\frac{\beta^2 4^{n-1}}{L^2}\right)\langle\psi|\hat{A}^2|\psi\rangle\\
&+\lambda_0\left(\frac{2\alpha\gamma 4^n}{L^2}-\frac{4\gamma^2 16^n}{L^4}\right)\langle\psi|\hat{A}|\psi\rangle-\tilde{\lambda}_0\left(\frac{\beta 2^{n-1}}{L}+\frac{\gamma 4^n}{L^2}\right)\langle\psi|\hat{A}|\tilde{\psi}\rangle+\tilde{\lambda}_0\left(\frac{\beta 2^{n-1}}{L}-\frac{\gamma 4^n}{L^2}\right)\langle\psi|\hat{A}^\dagger|\tilde{\psi}\rangle\\
&+\tilde{\lambda}_0\left(\frac{2\gamma 4^n}{L^2}-\alpha\right)\langle\psi|\tilde{\psi}\rangle+\lambda_0\left(\frac{\alpha\epsilon 4^n}{L^2}-\frac{\alpha\epsilon 2^{n-1}}{L}+\frac{2\gamma\epsilon 2^{n-1}4^n}{L^3}-\frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi\hat{A}|\psi\rangle\\
&+\lambda_0\left(\frac{\alpha\epsilon 4^n}{L^2}+\frac{\alpha\epsilon 2^{n-1}}{L}-\frac{2\gamma\epsilon 2^{n-1}4^n}{L^3}-\frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi\hat{A}^\dagger|\psi\rangle\\
&+\lambda_0\left(\frac{4\gamma\epsilon 16^n}{L^4}-\frac{2\alpha\epsilon 4^n}{L^2}+\frac{2\beta\epsilon 4^{n-1}}{L^2}+\frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi|\psi\rangle\\
&+\lambda_0\left(-\frac{\beta\epsilon 4^{n-1}}{L^2}-\frac{\beta\epsilon 2^{n-1}4^n}{L^3}+\frac{\gamma\epsilon 16^n}{L^4}+\frac{\gamma\epsilon 2^{n-1}4^n}{L^3}\right)\langle\psi|\hat{A}^\dagger\hat{D}_\phi\hat{A}^\dagger|\psi\rangle\\
&+\lambda_0\left(-\frac{\beta\epsilon 4^{n-1}}{L^2}+\frac{\beta\epsilon 2^{n-1}4^n}{L^3}+\frac{\gamma\epsilon 16^n}{L^4}-\frac{\gamma\epsilon 2^{n-1}4^n}{L^3}\right)\langle\psi|\hat{A}\hat{D}_\phi\hat{A}|\psi\rangle+\lambda_0\left(\frac{2\beta\epsilon 2^{n-1}4^n}{L^3}-\frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{A}^\dagger\hat{D}_\phi|\psi\rangle\\
&+\lambda_0\left(-\frac{2\beta\epsilon 2^{n-1}4^n}{L^3}-\frac{2\gamma\epsilon 16^n}{L^4}\right)\langle\psi|\hat{A}\hat{D}_\phi|\psi\rangle+\lambda_0\left(-\frac{\epsilon^2 4^{n-1}}{L^2}+\frac{\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{A}\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}|\psi\rangle\\
&+\lambda_0\left(\frac{2\epsilon^2 4^n 2^{n-1}}{L^3}-\frac{2\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}|\psi\rangle+\lambda_0\left(-\frac{2\epsilon^2 4^n 2^{n-1}}{L^3}-\frac{2\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi^\dagger\hat{D}_\phi\hat{A}^\dagger|\psi\rangle\\
&+\lambda_0\left(\frac{2\epsilon^2 16^n}{L^4}+\frac{\epsilon^2 4^{n-1}}{L^2}+\frac{\epsilon^2 16^n}{L^4}\right)\langle\psi|\hat{D}_\phi^\dagger\hat{D}_\phi|\psi\rangle+\tilde{\lambda}_0\left(\frac{\epsilon^2 2^{n-1}}{L}-\frac{\epsilon^2 4^n}{L^2}\right)\langle\tilde{\psi}|\hat{D}_\phi\hat{A}|\psi\rangle\\
&+\tilde{\lambda}_0\left(-\frac{\epsilon^2 2^{n-1}}{L}-\frac{\epsilon^2 4^n}{L^2}\right)\langle\tilde{\psi}|\hat{D}_\phi\hat{A}^\dagger|\psi\rangle+\tilde{\lambda}_0\left(\frac{2\epsilon^2 4^n}{L^2}\right)\langle\tilde{\psi}|\hat{D}_\phi|\psi\rangle\Bigg].
\end{aligned}
\tag{C11}
$$

APPENDIX D: 2D LINEAR BLACK-SCHOLES EQUATION COST FUNCTION

First, we define $\alpha = 1 - r\tau$, $\beta_x = (\frac{1}{2}\sigma_x^2 - r)\tau$, $\beta_y = (\frac{1}{2}\sigma_y^2 - r)\tau$, $\gamma_x = \frac{1}{2}\sigma_x^2\tau$, $\gamma_y = \frac{1}{2}\sigma_y^2\tau$, $\gamma_{xy} = \frac{1}{2}\rho\sigma_x\sigma_y\tau$, and

$$C_V = \left\langle \left(\alpha\mathbb{1} - \beta_x\frac{\hat{\partial}}{\partial x} - \beta_y\frac{\hat{\partial}}{\partial y} + \gamma_x\frac{\hat{\partial}^2}{\partial x^2} + \gamma_y\frac{\hat{\partial}^2}{\partial y^2} + \gamma_{xy}\frac{\hat{\partial}^2}{\partial x\partial y}\right)V \right. \\ \left. - \tilde{V}\left|\left(\alpha\mathbb{1} - \beta_x\frac{\hat{\partial}}{\partial x} - \beta_y\frac{\hat{\partial}}{\partial y} + \gamma_x\frac{\hat{\partial}^2}{\partial x^2} + \gamma_y\frac{\hat{\partial}^2}{\partial y^2} + \gamma_{xy}\frac{\hat{\partial}^2}{\partial x\partial y}\right)V - \tilde{V}\right.\right\rangle \tag{D1}$$

following from Eq. (1). After plugging in the definitions of $|V\rangle$, $|\tilde{V}\rangle$, and the derivatives, we are ultimately left with

$$\begin{aligned}
C_V =& \tilde{\lambda}_0^2 + \lambda_0^2\left(\alpha^2 + \frac{6\gamma_x^2 16^{n_x}}{L_x^4} + \frac{6\gamma_y^2 16^{n_y}}{L_y^4} + \frac{4^{n_x}\beta_x^2}{2L_x^2} + \frac{4^{n_y}\beta_y^2}{2L_y^2} - \frac{4\alpha\gamma_x 4^{n_x}}{L_x^2} - \frac{4\alpha\gamma_y 4^{n_y}}{L_y^2} + \frac{4^{n_x}4^{n_y}\gamma_{xy}^2}{4L_x^2L_y^2} + \frac{8\gamma_x\gamma_y 4^{n_x}4^{n_y}}{L_x^2L_y^2}\right) \\
&+ 2\lambda_0\mathrm{Re}\left[\tilde{\lambda}_0\left(-\alpha + \frac{2\gamma_x 4^{n_x}}{L_x^2} + \frac{2\gamma_y 4^{n_y}}{L_y^2}\right)\langle\psi|\tilde{\psi}\rangle + \tilde{\lambda}_0\left(-\frac{2^{n_x-1}\beta_x}{L_x} - \frac{4^{n_x}\gamma_x}{L_x^2}\right)\langle\psi|\hat{A}_x|\tilde{\psi}\rangle\right. \\
&+ \tilde{\lambda}_0\left(\frac{2^{n_x-1}\beta_x}{L_x} - \frac{4^{n_x}\gamma_x}{L_x^2}\right)\langle\psi|\hat{A}_x^\dagger|\tilde{\psi}\rangle + \tilde{\lambda}_0\left(-\frac{2^{n_y-1}\beta_y}{L_y} - \frac{4^{n_y}\gamma_y}{L_y^2}\right)\langle\psi|\hat{A}_y|\tilde{\psi}\rangle \\
&+ \tilde{\lambda}_0\left(\frac{2^{n_y-1}\beta_y}{L_y} - \frac{4^{n_y}\gamma_y}{L_y^2}\right)\langle\psi|\hat{A}_y^\dagger|\tilde{\psi}\rangle + \lambda_0\left(-\frac{4^{n_x}4^{n_y}\gamma_{xy}^2}{8L_x^2L_y^2} - \frac{4^{n_x}\beta_x^2}{4L_x^2} + \frac{16^{n_x}\gamma_x^2}{L_x^4}\right)\langle\psi|\hat{A}_x^2|\psi\rangle \\
&+ \lambda_0\left(\frac{8^{n_x}2^{n_y}\gamma_x\gamma_{xy}}{2L_x^3L_y}\right)\langle\psi|\hat{A}_x^2\hat{A}_y|\psi\rangle + \lambda_0\left(\frac{4^{n_x}4^{n_y}\gamma_{xy}^2}{16L_x^2L_y^2}\right)\langle\psi|\hat{A}_x^2\hat{A}_y^2|\psi\rangle \\
&+ \lambda_0\left(-\frac{8^{n_x}2^{n_y}\gamma_x\gamma_{xy}}{2L_x^3L_y}\right)\langle\psi|\hat{A}_x^2\hat{A}_y^\dagger|\psi\rangle + \lambda_0\left(\frac{4^{n_x}4^{n_y}\gamma_{xy}^2}{16L_x^2L_y^2}\right)\langle\psi|\hat{A}_x^2(\hat{A}_y^\dagger)^2|\psi\rangle \\
&+ \lambda_0\left(-\frac{4^{n_x}4^{n_y}\gamma_{xy}^2}{8L_x^2L_y^2} - \frac{4^{n_y}\beta_y^2}{4L_y^2} + \frac{16^{n_y}\gamma_y^2}{L_y^4}\right)\langle\psi|\hat{A}_y^2|\psi\rangle \\
&+ \lambda_0\left(\frac{2^{n_x}8^{n_y}\gamma_y\gamma_{xy}}{2L_xL_y^3}\right)\langle\psi|\hat{A}_y^2\hat{A}_x|\psi\rangle + \lambda_0\left(-\frac{2^{n_x}8^{n_y}\gamma_y\gamma_{xy}}{2L_xL_y^3}\right)\langle\psi|\hat{A}_y^2\hat{A}_x^\dagger|\psi\rangle \\
&+ \lambda_0\left(-\frac{4\times 16^{n_x}\gamma_x^2}{L_x^4} - \frac{4\times 4^{n_x}4^{n_y}\gamma_x\gamma_y}{L_x^2L_y^2} + \frac{2\times 4^{n_x}\alpha\gamma_x}{L_x^2}\right)\langle\psi|\hat{A}_x|\psi\rangle \\
&+ \lambda_0\left(-\frac{8^{n_x}2^{n_y}\gamma_x\gamma_{xy}}{L_x^3L_y} + \frac{2\times 4^{n_x}4^{n_y}\gamma_x\gamma_y}{L_x^2L_y^2} - \frac{2^{n_x}8^{n_y}\gamma_y\gamma_{xy}}{L_xL_y^3} + \frac{2^{n_x}2^{n_y}\alpha\gamma_{xy}}{2L_xL_y} - \frac{2^{n_x}2^{n_y}\beta_x\beta_y}{2L_xL_y}\right)\langle\psi|\hat{A}_x\hat{A}_y|\psi\rangle \\
&+ \lambda_0\left(\frac{8^{n_x}2^{n_y}\gamma_x\gamma_{xy}}{L_x^3L_y} + \frac{2\times 4^{n_x}4^{n_y}\gamma_x\gamma_y}{L_x^2L_y^2} + \frac{2^{n_x}8^{n_y}\gamma_y\gamma_{xy}}{L_xL_y^3} - \frac{2^{n_x}2^{n_y}\alpha\gamma_{xy}}{2L_xL_y} + \frac{2^{n_x}2^{n_y}\beta_x\beta_y}{2L_xL_y}\right)\langle\psi|\hat{A}_x\hat{A}_y^\dagger|\psi\rangle \\
&+ \left.\lambda_0\left(-\frac{4\times 16^{n_y}\gamma_y^2}{L_y^4} - \frac{4\times 4^{n_x}4^{n_y}\gamma_x\gamma_y}{L_x^2L_y^2} + \frac{2\times 4^{n_y}\alpha\gamma_y}{L_y^2}\right)\langle\psi|\hat{A}_y|\psi\rangle\right].
\end{aligned} \tag{D2}$$

APPENDIX E: BUCKMASTER EQUATION COST FUNCTION

The quantum analog of the Buckmaster differential operator is $\hat{O}|\tilde{f}\rangle = (\frac{\hat{\partial}^2}{\partial x^2}\hat{D}_{\tilde{f}}^3 + \frac{\hat{\partial}}{\partial x}\hat{D}_{\tilde{f}}^2)|\tilde{f}\rangle$. With the forward Euler scheme, our cost function is

$$C = \||f\rangle - (\mathbb{1} + \tau\hat{O})|\tilde{f}\rangle\|^2. \tag{E1}$$

After expanding with the inner product, substituting $|f\rangle = \lambda_0\hat{U}(\lambda)|0\rangle = \lambda_0|\psi\rangle$ and the adder expressions for the derivatives, and simplifying, we get

$$\begin{aligned}
C =& \lambda_0^2 + 2\lambda_0\mathrm{Re}\left(-\tilde{\lambda}_0\langle\psi|\tilde{\psi}\rangle + \frac{2\times 4^n\tilde{\lambda}_0^4\tau}{L^2}\langle\psi|\hat{D}_{\tilde{\psi}}^3|\tilde{\psi}\rangle\right. \\
&- \frac{4^n\tilde{\lambda}_0^4\tau}{L^2}\left(\langle\psi|\hat{A}\hat{D}_{\tilde{\psi}}^3|\tilde{\psi}\rangle + \langle\psi|\hat{A}^\dagger\hat{D}_{\tilde{\psi}}^3|\tilde{\psi}\rangle\right) \\
&+ \left.\frac{\alpha 2^{n-1}\tilde{\lambda}_0^3\tau}{L}\left(-\langle\psi|\hat{A}\hat{D}_{\tilde{\psi}}^2|\tilde{\psi}\rangle + \langle\psi|\hat{A}^\dagger\hat{D}_{\tilde{\psi}}^2|\tilde{\psi}\rangle\right)\right).
\end{aligned} \tag{E2}$$

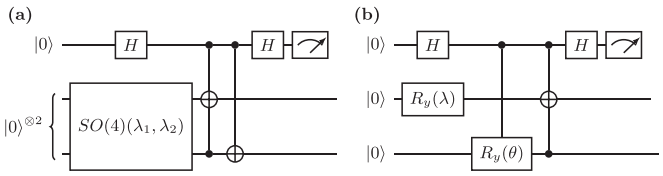

FIG. 13. Circuit diagrams for Fig. 10: (a) two-qubit-controlled adder implemented by the Toffoli and CNOT gates for expectation value $\langle\psi|\hat{A}|\psi\rangle$ and (b) bottom ancilla qubit used to implement the controlled diagonal gate for expectation value $\langle\psi|\hat{D}_\phi|\psi\rangle$. Both expectation values are calculated using the Hadamard test.

APPENDIX F: DETERMINISTIC KARDAR-PARISI-ZHANG EQUATION COST FUNCTION

To time evolve this equation we must first train an auxiliary state $|\chi\rangle$ to represent $\frac{\hat{\partial}}{\partial x}|\tilde{f}\rangle$. Our cost function is thus

$$C = \left\| |\chi\rangle - \frac{\hat{\partial}}{\partial x}|\tilde{f}\rangle \right\|^2. \tag{F1}$$

Letting $|\chi\rangle = \theta_0\hat{U}(\theta)|0\rangle = \theta_0|\phi\rangle$ and $|\tilde{f}\rangle = \tilde{\lambda}_0\hat{U}(\tilde{\boldsymbol{\lambda}})|0\rangle = \tilde{\lambda}_0|\tilde{\psi}\rangle$ and substituting the spatial derivatives, we get

$$C_\chi = \theta_0^2 - \frac{2^n}{L}\tilde{\lambda}_0\theta_0\mathrm{Re}(\langle\phi|\hat{A}|\tilde{\psi}\rangle - \langle\phi|\hat{A}^\dagger|\tilde{\psi}\rangle). \tag{F2}$$

Now we can construct our quantum operator $\hat{O}|\tilde{f}\rangle = \alpha\frac{\hat{\partial}^2}{\partial x^2}|\tilde{f}\rangle + \beta\hat{D}_\chi|\chi\rangle$. Letting $|f\rangle = \lambda_0\hat{U}(\lambda)|0\rangle = \lambda_0|\psi\rangle$ and substituting into Eq. (E1), we get

$$\begin{aligned} C_f = \lambda_0^2 - 2\lambda_0\mathrm{Re}\Bigg[&\tilde{\lambda}_0\left(1 - \frac{2\alpha\tau 4^n}{L^2}\right)\langle\psi|\tilde{\psi}\rangle \\ &+ \tilde{\lambda}_0\alpha\tau\frac{4^n}{L^2}(\langle\psi|\hat{A}|\tilde{\psi}\rangle + \langle\psi|\hat{A}^\dagger|\tilde{\psi}\rangle) \\ &+ \theta_0^2\tau\beta\langle\psi|\hat{D}_\phi|\phi\rangle\Bigg]. \end{aligned} \tag{F3}$$

APPENDIX G: HARDWARE DEMONSTRATION OF CIRCUIT DIAGRAMS

The circuits for Fig. 10 are shown in Fig. 13.

[1] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, and D. Jaksch, Variational quantum algorithms for nonlinear problems, Phys. Rev. A **101**, 010301(R) (2020).

[2] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, New York, 2010).

[4] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[5] O. Kyriienko, A. E. Paine, and V. E. Elfving, Solving nonlinear differential equations with differentiable quantum circuits, Phys. Rev. A **103**, 052416 (2021).

[6] H.-L. Liu, Y.-S. Wu, L.-C. Wan, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, Variational quantum algorithm for the Poisson equation, Phys. Rev. A **104**, 022418 (2021).

[7] P. Mocz and A. Szasz, Toward cosmological simulations of dark matter on quantum computers, Astrophys. J. **910**, 29 (2021).

[8] Y. Sato, R. Kondo, S. Koide, H. Takamatsu, and N. Imoto, Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation, Phys. Rev. A **104**, 052409 (2021).

[9] P. García-Molina, J. Rodríguez-Mediavilla, and J. J. García-Ripoll, Quantum Fourier analysis for multivariate functions and applications to a class of Schrödinger-type partial differential equations, Phys. Rev. A **105**, 012433 (2022).

[10] F. Yew Leong, W.-B. Ewe, and D. Enshan Koh, Variational quantum evolution equation solver, Sci. Rep. **12**, 10817 (2022).

[11] F. Yew Leong, D. Enshan Koh, W.-B. Ewe, and J. F. Kong, Variational quantum simulation of partial differential equations: Applications in colloidal transport, arXiv:2307.07173.

[12] W.-B. Ewe, D. Enshan Koh, S. T. Goh, H.-S. Chu, and C. E. Png, Variational quantum-based simulation of waveguide modes, IEEE Trans. Microwave Theory Techn. **70**, 2517 (2022).

[13] L. Mouton, F. Reiter, Y. Chen, and P. Rebentrost, Towards deep learning-based quantum algorithms for solving nonlinear partial differential equations, arXiv:2305.02019.

[14] Ó. Amaro and D. Cruz, A living review of quantum computing for plasma physics, arXiv:2302.00001.

[15] B. D. Clader, B. C. Jacobs, and C. R. Sprouse, Preconditioned quantum linear system algorithm, Phys. Rev. Lett. **110**, 250504 (2013).

[16] 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).

[17] A. Montanaro and S. Pallister, Quantum algorithms and the finite element method, Phys. Rev. A **93**, 032324 (2016).

[18] J. M. Arrazola, T. Kalajdzievski, C. Weedbrook, and S. Lloyd, Quantum algorithm for nonhomogeneous linear partial differential equations, Phys. Rev. A **100**, 032306 (2019).

[19] A. M. Childs, J.-P. Liu, and A. Ostrander, High-precision quantum algorithms for partial differential equations, Quantum **5**, 574 (2021).

[20] S. Jin, N. Liu, and Y. Yu, Quantum simulation of partial differential equations via Schrodingerisation: Technical details, arXiv:2212.14703.

[21] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. **103**, 150502 (2009).

[22] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J. S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier *et al.*, Benchmarking an 11-qubit quantum computer, Nat. Commun. **10**, 5464 (2019).

[23] J. Gonzalez-Conde, Á. Rodríguez-Rozas, E. Solano, and M. Sanz, Simulating option price dynamics with exponential quantum speedup, Phys. Rev. Res. **5**, 043220 (2023).

[24] M. Moosa, T. W. Watts, Y. Chen, A. Sarma, and P. L. McMahon, Linear-depth quantum circuits for loading Fourier

approximations of arbitrary functions, Quantum Sci. Technol. **9**, 015002 (2024).

[25] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, Transformation of quantum states using uniformly controlled rotations, Quantum Inf. Comput. **5**, 467 (2005).

[26] B. T. Kiani, G. De Palma, D. Englund, W. Kaminsky, M. Marvian, and S. Lloyd, Quantum advantage for differential equation analysis, Phys. Rev. A **105**, 022415 (2022).

[27] K. Nakaji and N. Yamamoto, Expressibility of the alternating layered ansatz for quantum computation, Quantum **5**, 434 (2021).

[28] K. Sharma, A. Arrasmith, and P. J. Coles, Variational quantum state eigensolver, npj Quantum Inf. **8**, 113 (2022).

[29] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*, 5th ed. (Springer, Berlin, 1998), pp. 266–283.

[30] J. Ankudinova and M. Ehrhardt, On the numerical solution of nonlinear Black–Scholes equations, Comput. Math. Appl. **56**, 799 (2008).

[31] S. Chatterjee, Universality of deterministic KPZ, arXiv:2102.13131.

[32] Y. Y. Liu, Z. Chen, C. Shu, S. C. Chew, B. C. Khoo, X. Zhao, and Y. D. Cui, Application of a variational hybrid quantum-classical algorithm to heat conduction equation and analysis of time complexity, Phys. Fluids **34**, 117121 (2022).

[33] E. Rosenberg, P. Ginsparg, and P. L. McMahon, Experimental error mitigation using linear rescaling for variational quantum eigensolving with up to 20 qubits, Quantum Sci. Technol. **7**, 015024 (2022).

[34] N. M. Guseynov, A. A. Zhukov, W. V. Pogosov, and A. V. Lebedev, Depths analysis of variational quantum algorithms for heat equation, Phys. Rev. A **107**, 052422 (2023).

[35] A. J. Pool, A. D. Somoza, C. Mc Keever, M. Lubasch, and B. Horstmann, Nonlinear dynamics as a ground-state solution on quantum computers, arXiv:2403.16791.

[36] A. Sarma, T. W. Watts, M. Moosa, Y. Liu, and P. L. McMahon, Quantum variational solving of nonlinear and multi-dimensional partial differential equations (2023), https://doi.org/10.5281/zenodo.10035285.