Explicit block encodings of boundary value problems for many-body elliptic operators

Tyler Kharazi[1,2], Ahmad M. Alkadri[3], Jin-Peng Liu[4,5,6], Kranthi K. Mandadapu[3,7], and K. Birgitta Whaley[1,2]

[1]Department of Chemistry, University of California, Berkeley
[2]Berkeley Quantum Information and Computation Center, University of California, Berkeley
[3]Department of Chemical and Biomolecular Engineering, University of California, Berkeley
[4]Yau Mathematical Sciences Center, Tsinghua University
[5]Center for Theoretical Physics, Massachusetts Institute of Technology
[6]Simons Institute and Department of Mathematics, University of California, Berkeley
[7]Chemical Sciences Division, Lawrence Berkeley National Laboratory, Berkeley

May 27, 2025

Simulation of physical systems is one of the most promising use cases of future digital quantum computers. In this work we systematically analyze the quantum circuit complexities of block encoding the discretized elliptic operators that arise extensively in numerical simulations for partial differential equations, including high-dimensional instances for many-body simulations. When restricted to rectangular domains with separable boundary conditions, we provide explicit circuits to block encode the many-body Laplacian with separable periodic, Dirichlet, Neumann, and Robin boundary conditions, using standard discretization techniques from low-order finite difference methods. To obtain high-precision, we introduce a scheme based on periodic extensions to solve Dirichlet and Neumann boundary value problems using a high-order finite difference method, with only a constant increase in total circuit depth and subnormalization factor. We then present a scheme to implement block encodings of differential operators acting on more arbitrary domains, inspired by Cartesian immersed boundary methods. We then block encode the many-body convective operator, which describes interacting particles experiencing a force generated by a pair-wise potential given as an inverse power law of the interparticle distance. This work provides concrete recipes that are readily translated into quantum circuits, with depth logarithmic in the total Hilbert space dimension, that block encode operators arising broadly in applications involving the quantum simulation of quantum and classical many-body mechanics.

Tyler Kharazi: kharazitd@berkeley.edu

Contents

1 Introduction

Elliptic operators constitute a class of differential operators that generalize the Laplace operator and arise in applications ranging from electrostatics to continuum and quantum mechanics [1]. In the context of continuum mechanics, elliptic operators commonly arise as the spatial operators within a time-dependent partial differential equation (PDE), and the related steady-state problem can be solved using a variational principle [2]. In quantum mechanics, elliptic operators often arise in quantum chemistry and in condensed matter physics as the Hamiltonian operator of the Schrödinger equation. In classical settings, the spatial operators in the heat equation and more generally the advection-diffusion equation often correspond to an elliptic operator. Solving these problems often requires a large effort in all but the most simple examples, since numerical discretization methods are needed to obtain accurate approximations. Direct discretization methods such as finite differences have a cost that grows exponentially with the spatial dimension and the number of particles. If one discretizes each dimension with N grid points, for an interacting many-body system containing η particles in some d-dimensional domain $\Omega \subset \mathbb{R}^d$, the dimension of the vector space upon which the discretized operator acts will be of extremely high dimension $O(N^{\eta d})$, which becomes intractable for even modest numbers of particles. Quantum algorithms provide a promising approach, as the $N^{\eta d}$ grid points can be stored in the quantum state of $O(\eta d \log(N))$ qubits using the technique of first quantization. We also note that N is not the exponentially growing factor in these simulations, as N is chosen with respect to a fixed accuracy, however, $N^{\eta d}$ grows exponentially in the number of particles η and number of independent spatial dimensions d. Therefore, it is expected that quantum advantage will be most pronounced in the regime where ηd is large, since each additional particle has only a linear increase in memory on the quantum computer, whereas classically the resource requirements grow exponentially with each additional particle.

The analysis of quantum algorithms for solving linear and nonlinear ordinary [3–9] and partial [10–15] differential equations is an area of increasingly active development. A new family of approaches for solving linear ordinary differential equations, including those obtained by discretizing a PDE, has been recently developed that uses ideas of "linear combination of Hamiltonian simulations (LCHS)" or "Schrödingerization" to approximate the non-unitary time evolution. The Schrödingerization technique [16] has been used to analyze quantum simulations for solving the Fokker-Planck equation as well as many other kinds of PDEs [17–20]. Proposed at a similar time, [21] provides an algorithm to implement the LCHS as well as to implement fast initial state preparation. This approach was recently improved [22], exponentially improving the algorithmic dependence on accuracy over the original LCHS algorithms: the improvement is also asymptotically optimal for quantum simulation of non-unitary dynamics. Despite these advancements, there is nevertheless very little evidence for exponential quantum advantage for any end-to-end application in the quantum simulation of PDE's. One reason for this gap is a fundamental lack of resources which describe the basic quantum operations needed to implement the prescribed algorithm. Our goal in this work is to provide an accounting of the computational resources for the block encoding of various differential and multiplication operators. Using low level primitives such as incrementer and reflection circuits, we provide explicit circuit constructions of LCUs for the Laplacian and convective operators under different boundary conditions and estimate the number of Toffoli gates needed to implement the operations. We expect that these subroutines can be broadly employed in quantum algorithms for practical simulations of differential equations, quantum and classical.

Quantum algorithms using a block encoding input model will likely be the preferred approach in fault-tolerant quantum computation. Indeed, [22], and many other near optimal quantum algorithms, use block encoding as the input model. In most cases, the block encoding is also applied many times throughout the execution of the algorithm. Therefore, the cost to implement the block encoding is an important factor in determining the overall runtime of the proposed algorithm. A standard approach to implementing block encodings is linear combinations of unitaries (LCU) [23]. However, other approaches based on sparse access models, similar to classical sparse matrix formats, have also been explored [24–26]. Combined with quantum signal processing (QSP) [27, 28] or more generally, the quantum singular value transformation (QSVT) [29], a large class of matrix transformations can be enacted on the block encoded matrix. These include inversion, exponentiation (or time evolution), eigenvalue filtering , and many others[29, 30]. Such routines build up a polynomial transformation of the block encoding matrix by performing the block encoding circuit, interleaved with a rotation, a number of times directly related to the degree of the polynomial approximation to the desired function. Therefore, the efficiency of the block encoding is a critical component of the overall efficiency of the proposed quantum algorithm.

The major contribution of this work is the construction and cost analysis of efficient quantum circuits to block encode a broad family of differential operators with nearly arbitrary boundary conditions to high-precision and near-optimal gate complexity and subnormalization. We focus here on the particular problem of block encoding linear operators, however through the use of the Carleman linearization [31–35], these constructions can be readily extended to represent nonlinear PDEs. In addition to LCHS, these block encodings can be used in conjunction with time-stepping schemes [36] as well as with history state schemes [37] to solve time-dependent PDE's. The key results are summarized in Table 1. We find that the quantum circuits implementing these block encodings can be done with as few as $O(pd\eta \log(N))$ Toffoli or simpler gates for a separable differential operator and with as few as $\widetilde{O}((pd\eta)^2 N \log^2(N))$ Toffoli or simpler gates for pairwise interactions with a radially symmetric inverse power law potential V, where N is chosen so that for any $\epsilon > 0$, $N^{-p+1} < \epsilon$. These constructions provide an exponential speedup in memory and circuit complexity over classical representations of finite difference discretizations of differential operators. This further implies that quantum algorithms providing exponential speedups for solving a commonly encountered subset of elliptic PDEs using the block encoding framework are retained when including the circuit cost of the block encodings.

Elliptic operators can be defined in a very general way and encode a great many examples of physical problems. We use the following mathematical definition to clarify what is meant by an elliptic operator for the sake of this work.

Definition 1 (Linear elliptic operator of order m)**.** *Let L be a linear differential operator of order m on a domain $\Omega \subset \mathbb{R}^d$ given by*

$$Lu = \sum_{|\alpha| \leq m} a_\alpha(\mathbf{x}) \partial^\alpha u \,,$$

where $\alpha = (\alpha_1, \ldots, \alpha_d)$ denotes a multi-index, $\partial^\alpha u = \partial_1^{\alpha_1} \cdots \partial_d^{\alpha_d} u$, and here $|\alpha| = \alpha_1 + \cdots + \alpha_m$. Then L is called elliptic if for every $x \in \Omega$ and every $\boldsymbol{v} \neq \mathbf{0} \in \mathbb{R}^d$,

$$\sum_{|\alpha| = m} a_\alpha(x) \boldsymbol{v}^\alpha \neq 0 \,,$$

where $\boldsymbol{v}^\alpha := v_1^{\alpha_1} \ldots v_d^{\alpha_d}$.

Term	Laplacian (non-interacting)				Convective Operator
Boundary Condition	Periodic	Dirichlet	Neumann/Robin	Periodic Extension	Periodic
Toffoli complexity	$O(dp\log(N))$	$O(d\log(N))$	$O(d\log(N))$	$O(dp\log(N))$	$O((p\eta d)^2 N\log^2(N))$
subnormalization	$O(d)$	$O(d)$	$O(d)$	$O(d)$	$O(\eta^2 d\|\|\nabla V\|\|_\infty)$
#ancilla	$O(\log(dp))$	$O(\log(d))$	$O(\log(d))$	$\widetilde{O}(\log(dp))$	$O(\log(\log(N)\eta dp))$
accuracy	$O(N^{-p+1})$	$O(N^{-2})$	$O(N^{-1})$	$O(N^{-p+1})$	$O(N^{-p+1})$

Table 1: Features of the block encodings for the various schemes and boundary conditions employed in this work. Here N is the number of grid points in $d=1$ spatial dimensions, p is the number of grid points used in the FD stencil, and η is the number of particles. Accuracy refers to the rate at which a given discretization converges to the true solution, in terms of N. The column labels periodic, Dirichlet, and Neumann refer to the $d-$dimensional Laplacian with the corresponding boundary conditions. Periodic extension refers to the solution of boundary value problems by periodic extensions of functions that is introduced in subsection 3.4. The operator $\nabla V(x)\cdot\nabla$ is the convective operator given by a scalar pair-wise inverse power law potential V summed over the $\binom{\eta}{2}$ interactions between pairs of particles. The factor of N is related to the polynomial approximation rate to inverse power law potentials. The introduction of a cutoff parameter δ, which we assume scales as $\delta\sim 1/N$, controls the divergence near the singular portion of V and gives the associated scaling (see e.g. Lemma 5).

Of particular interest in the physical sciences are the second-order elliptic operators where $|\alpha|\leq 2$. In the most general second-order case, we may have the rank-2 tensor

$$A(x)=\sum_{i,j=0}^{d-1} a_{ij}(x)\,|i\rangle\langle j|\,,$$

with elements $a_{ij}(x)\in\mathcal{C}^2$, we may write the second-order elliptic operator in divergence form

$$Lu=-\nabla\cdot(A(x)\nabla u)\,, \tag{1}$$

or equivalently in divergence-free form

$$Lu=\sum_{ij}a_{ij}(x)\frac{\partial^2}{\partial x^i\partial x^j}u+\mathbf{b}(x)\cdot\nabla u\,, \tag{2}$$

where

$$\mathbf{b}_i(x)=\sum_{j=0}^{d-1}\frac{\partial a_{ij}(x)}{\partial x^j}\,.$$

In the isotropic case where $A(x)=V(x)$ where $V:\mathbb{R}^d\to\mathbb{R}$ is a multiplication operator corresponding to scalar multiplication by $V(x)$, this simplifies further to

$$Lu=V(x)\Delta+\nabla V(x)\cdot\nabla\equiv\sum_{i=0}^{d-1}\left(V(x)\frac{\partial^2}{\partial x^i\partial x^i}+\frac{\partial}{\partial x^i}V(x)\frac{\partial}{\partial x^i}\right)\,, \tag{3}$$

where we interpret $V(x)$ as a potential function acting on all of the particle position coordinates and $\nabla V(x)\sim\mathbf{b}(x)$ the associated force vector. We focus on the ubiquitous case where V is an scalar function from $\mathbb{R}^d$ to $\mathbb{R}$ and the order of the differential operator is 2, however, generalizations of the schemes we present in this case to vector or tensor functions are straightforward.

1.1 Prior Work

Quantum algorithms addressing the complexity of block encoding and simulating systems given by various kinds of elliptic operators have been an active area of research. In the work of [15], explicit block encodings were provided for the d dimensional Laplacian with periodic boundary conditions for arbitrary order finite difference stencils. There has also been some work exploring the application of pseudospectral methods [6], although implementing block encodings of the obtained linear system is likely to be more challenging as the obtained matrix does not enjoy the sparse and regularly repeating structure as the finite difference case. In the work of [38], it was shown that the Neumann and Dirichlet boundary value problems for the Laplacian on a grid can be implemented by providing an extra "loop" on boundary nodes of the associated graph Laplacian, however, it was unclear how to implement the corresponding block encoding. In addition, there have been explicit block encodings provided for some sparse matrices such as the graph Laplacian of a d-regular graph [26], as well as schemes for the so-called "hierarchical matrices" considered in [39]. In the context of real-space simulations, [40] considers the generic costs associated with simulating real-space dynamics resulting from direct domain discretization methods under the influence of a Coulomb potential, where the dependence on the accuracy was later improved in [41]. In the context of quantum chemistry algorithms employing real (or dual) space grids in first quantization, [42] provides a very detailed construction of the block encoding of the electronic Hamiltonian using a plane wave discretization.

The work of [43] considers the task of block encoding the pseudo-differential operator (PDO) representation of an elliptic operator. Similar to pseudo spectral methods, the number of Fourier modes used to approximate the PDO, N, can be taken as $O(\log(1/\epsilon))$, in contrast to standard high-order difference methods that have only polynomial convergence with the number of stencil points k, since $N \sim \epsilon^{-1/k}$. For example, section 6.1 of that work shows how to block encode a variable coefficient elliptic operator of the form $I - \nabla \cdot (V(x)\nabla)$ using $O(dnr(\log(1/\epsilon) + d + n))$ elementary operations, where r is the number of Fourier modes needed to approximate V to the desired accuracy, and $n = \log(N) = \log\log(1/\epsilon)$ is the number of qubits, and d is the spatial dimension. Therefore, when the number of Fourier modes needed to expand a scalar potential $V(x)$ is small, this can be very efficient. However, in the cases we consider in this work, the potential function V is given as an inverse power law of the interparticle distance and the number of Fourier modes needed to obtain high precision can be very large, $r \sim O(N)$. Although [43] only considers a periodic boundary condition, replacing the quantum Fourier transform of this algorithm with the discrete cosine and discrete sine transformations should in principle be able to diagonalize the operator with homogeneous Neumann and Dirichlet conditions respectively [44]. However, the authors of [43] also showed that in the most general cases where the associated PDO symbol is not separable, this approach can have exponentially small success probability. In the particular case of a separable symbol, they show, similar to the findings of the current work, that the success probability can be $\widetilde{O}(1)$. We have also recently become aware of [45], which provides explicit block encodings of Hamiltonian operators given as a linear combination of polynomials in the position and momentum operators in the context of the Schrödingerization technique.

1.2 Our Contributions

In the current work, we provide explicit block encodings of the differential and multiplication operators needed to block encode an elliptic operator on a $d-$dimensional cubic domain with Periodic, Neumann, Dirichlet, and Robin boundary conditions. We provide

circuit-level primitives for every part of this work, except for the evaluation of phase factors in block encodings that rely on QSP. We will also use controlled applications of some of the circuit constructions, but do not address their controlled implementation. In Appendix A, we provide explicit circuits for the shift and reflection operations used in our LCU, each of which can be implemented using $O(n)$ Toffoli or simpler gates to implement. We show how to efficiently modify the known block encoding of the Laplacian with periodic boundary conditions to enforce Dirichlet, Neumann, and Robin conditions. In d-dimensions if the boundary is rectangular and the forcing function is separable and satisfies consistency conditions where the boundaries meet, we show how these block encodings can be efficiently combined to block encode the differential equation and boundary value problem in d spatial dimensions. We provide explicit gate costs, as well as numerical and asymptotic estimates for the success probability of this procedure for the case of a simple Dirichlet boundary value problem and show that our method of enforcing Dirichlet conditions contributes only a small additive constant to the subnormalization factor over that of the periodic case. We then show how a natural extension of the technique we used for regular boundary conditions can be used for boundary conditions imposed on the "interior" of a computational domain, as might occur when studying flow around an obstacle, or geometries with holes in them (see Fig. 3). We additionally provide a method based on periodic extensions to block encode Dirichlet and Neumann boundary conditions to high precision and furthermore, obtain the same order of accuracy on the boundary as on the interior, at least quadratically improving the global convergence rate over one-sided schemes for approximating the boundary condition. As an immediate extension of this approach, we show how to block encode the Laplacian on a more arbitrary domain, assuming access to an oracle that can flag grid points as belonging to the either the interior, exterior, or boundary of the desired domain. We additionally provide a software demonstration of these methods, as well as circuit constructions implemented in Pennylane[46], which can be found on Github [47].

We then address the block encoding of the so-called convective operator, which takes the form $\nabla V(\mathbf{x}) \cdot \nabla$, for a scalar pair-wise interparticle potential function V, evaluated over, $\mathbf{x}$ the multi-particle position coordinates. Because the differential operators involved in the convective term are discretized in a similar manner to the Laplacian terms, our contribution will first focus on constructing the block encoding of a multi-particle scalar potential V and evaluating the multi-particle gradient ∇V. We explicitly consider the case when $V(r)$ is a Lennard-Jones potential, a commonly used potential in chemical and molecular dynamics simulations. We note that this choice of V incorporates many of the salient features of other important scenarios, as it involves the evaluation of an inverse power law function of the inter-particle distance: such interaction potentials arise regularly in physical models. We first review the gate cost to block encode a discrete position operator, that we then can use to form a block encoding of a discrete distance function. We discuss the number of queries to this distance function based on the convergence rate of a polynomial approximation to the desired potential function V. We then detail the construction of the block encoding of the gradient operator ∇V, using a finite difference approach. Since from Def. 1, every elliptic operator for a given scalar potential can always be written as a Laplacian term plus a convective term. Combining the block encoding of the scalar operators V with the block encodings that we obtain for the differential operators is sufficient to implement block encodings for a very broad class of elliptic PDE's.

1.3 Overview

The remainder of the paper proceeds as follows. Section 2 introduces the notation we use

throughout the text and briefly reviews the finite difference method and how boundary conditions are enforced. We also introduce some of the quantum primitives that we use in the text, such as linear combination of unitaries (LCU) [23] and quantum signal processing (QSP) [27]. Section 3 shows how to block encode the $d-$dimensional Laplacian with various boundary conditions occurring on a separable Cartesian domain $\sim [0, L]^d$. We provide examples for block encoding Dirichlet, Neumann, Robin, and mixed boundary conditions and gate complexities to implement the corresponding LCUs. In addition, we show how to use the method of periodic embedding to obtain high-order accuracy schemes for Dirichlet and Neumann boundary conditions in d dimensions. We then present a method that can be used to block encode the Laplacian on more irregular domains. Section 4 introduces the primitives needed to block encode the convective term $\nabla V(x) \cdot \nabla$, including the evaluation of multi-particle potential functions given by an inverse power law. We provide circuits to compute the squared distance between any two grid points in d dimensions and then bound the polynomial degree needed to obtain an $\epsilon-$accurate approximation of V in terms of a cutoff parameter. Finally, we provide an alternative approach to uniform polynomial approximation by approximating the function with different polynomials close to and far away from the singular point in the potential that can be used to practically reduce the dependence on the cutoff in the numerical approximation of the desired function. Section 5 concludes with a discussion and outlook.

2 Background

2.1 Notation

We will use d to refer to the underlying spatial dimension of the differential operator, η to refer to the number of particles, and N the number of grid points used in each dimension. We will commonly use $p = 2a+1$ to be the number of grid points used in the stencil for the Laplacian and $2a$ the number of grid points for the first derivative. We use $r_j,\ 0 \leq |j| \leq a$ to refer to the central difference coefficients for the pth order finite difference stencil of the second derivative, and $c_j,\ 1 \leq |j| \leq a$ to refer to the central difference coefficients for the pth order finite difference stencil of the first derivative. We will also generally use the convention that upper indices are used to index particles and lower indices are used to index spatial dimensions. For example, if $\mathcal{L}$ is some linear differential operator for $d = 1$, then $\mathcal{L}_j^i$ refers to the action of $\mathcal{L}$ on particle $i \in [\eta]$ in dimension $j \in [d]$ where $[N] := \{0, \ldots, N-1\}$.

We will use the notation $\|\cdot\|$ to refer to the standard 2-norm when the argument is a vector and to the induced 2-norm if the argument is a matrix. We use the notation $\boldsymbol{\alpha}$ to refer to a multi-index. We will write $\alpha_i^j \in [N]$ to refer to an arbitrary grid point for particle j along direction i, so $\boldsymbol{\alpha} \in [N^{\eta d}]$. When the meaning is obvious, we may also use the simplified indicial notation $i \in [N^{\eta d}]$ to refer to the index given by the map $\alpha_j^i \in [N] \to \alpha_j^i N^{di+j} \in [N^{\eta d}]$. We use the notation $|x\rangle\,|y\rangle$ to refer to the tensor product of two n-qubit states. The notation $|\boldsymbol{\alpha}\rangle, \boldsymbol{\alpha} \in [N]^d$ should be interpreted to mean $|\alpha_0\rangle\,|\alpha_1\rangle \cdots |\alpha_{d-1}\rangle$, where each $\alpha_i \in [N]$. Therefore, the η particle d-dimensional Hilbert space is given by the span over states of the form $|\boldsymbol{\alpha}^0\rangle\,|\boldsymbol{\alpha}^1\rangle \cdots |\boldsymbol{\alpha}^{\eta-1}\rangle$, where each $|\boldsymbol{\alpha}^i\rangle$ is given as a N^d-dimensional state vector so that the total dimension of the underlying Hilbert space is $N^{\eta d}$.

We also use the following asymptotic notation

$$\begin{aligned} f \in O(g) &\iff \frac{f(n)}{g(n)} \to 0, \\ f \in \widetilde{O}(g) &\iff f \in O(g\text{polylog}(g)), \end{aligned} \tag{4}$$

and write $x(n) \to y$ to mean that $x(n)$ tends to y as n grows large.

2.2 Discretization Scheme

Finite difference methods (FDM)s are a robust technique for discretizing spatial derivative operators on Cartesian domains. We use the FDM by first discretizing the problem domain $\Omega \subset \mathbb{R}^d$ into regular grid points. If each of the d spatial dimensions are discretized with N grid points, then the domain Ω becomes associated with the set of grid points $x_{\boldsymbol{\alpha}} \in \mathbb{Z}_N^d$, where $\boldsymbol{\alpha}$ is a multi-index indexing all the grid points in the N^d dimensional discretized position space and $x_{\boldsymbol{\alpha}}$ is the map from the grid index onto the physical domain.

For example, in one spatial dimension, the first derivative operator can be approximated to second order using a central-difference discretization scheme as

$$\frac{d}{dx} f(x) \to \frac{f(x + h/2) - f(x - h/2)}{h/2} + O(h^2), \tag{5}$$

where $h = 1/N$. In addition, an arbitrary function f evaluated on the induced grid results in a N-dimensional vector

$$f(x) \to \sum_{i \in [N]} f(x_i) \, |i\rangle \, . \tag{6}$$

If we consider the second-order PDE with periodic boundary conditions

$$\begin{aligned} \frac{d^2}{dx^2} u(x) &= f(x) \;\; 0 < x < 1 \\ u(0) &= u(1), \end{aligned}$$

discretization of the interval $[0, 1]$ into N grid points and using a 3 point central-difference scheme to approximate the second derivative results in the linear system

$$\underbrace{\frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 1 & \cdots & 0 & 0 & 1 & -2 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \end{pmatrix}}_{\mathbf{u}} = \underbrace{\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-3} \\ f_{N-2} \\ f_0 \end{pmatrix}}_{\mathbf{f}} \tag{7}$$

We define L to be the matrix on the left hand side of (7) and $\mathbf{u}$, $\mathbf{f}$ are vectors corresponding to $u(x)$ and $f(x)$ respectively in the sense of (6).

We can generalize the above definitions of differential operators on lattices to higher-order derivatives, by using additional grid points in the evaluation of the differentials. The number of stencil points is closely connected to the convergence rate of the discretization

scheme, and thereby the number of grid points needed to solve a particular problem to a fixed error tolerance. The following theorem characterizes the error in implementing a $p = 2a + 1$ point finite difference stencil on N grid points in terms of the smoothness of the solution.

Lemma 1. *[Error in 2a+1 point stencil on N grid point Laplacian ([40] Theorem 7)] Let $\psi(x) \in \mathcal{C}^{2a+1}$, the space of functions with continuous derivatives up to order $2a+1$, be a solution to the PDE, then the error in the (2a+1)-point centered difference formula for the second derivative of $\psi(x)$ evaluated on a uniform mesh with spacing $h = 1/N$ is at most*

$$\frac{\pi^{3/2}}{9} e^{2a(1-\ln(2))} h^{2a-1} \max_x \left| \frac{\partial \psi(x)}{\partial x^{2a+1}} \right|. \tag{8}$$

Computing finite difference stencil coefficients for regularly spaced grid points is fairly routine and an analytic form of the coefficients are known to arbitrary order; a tabulated resource can even be found on Wikipedia [48]. Research on the computation of finite difference coefficients in more arbitrary settings can be found, e.g., in Ref. [49], and in general can be obtained via the theory of interpolating polynomials. For the second derivative with a p-point central difference scheme, the coefficients are given by

$$r_j \equiv \begin{cases} \frac{2(-1)^{j+1}(k!)^2}{j^2(a-j)!(a+j)!} & j \in \{1, \ldots, a\} \\ -2\sum_{j=1}^{a} r_j & j = 0 \\ r_{-j} & j \in \{-a, \ldots, -1\}. \end{cases} \tag{9}$$

Similarly, for the first derivative we have

$$c_j \equiv \begin{cases} \frac{(-1)^{j+1}((2a)!)^2}{j(2a-j)!(2a+j)!} & 1 \leq |j| \leq a \\ 0 & j = 0. \end{cases} \tag{10}$$

In addition to the promise on the accuracy of high-order finite difference stencils given the smoothness of the solution, we are also given a promise on the smoothness of the solution provided a smooth forcing function f and potential function V. This is characterized by the *elliptic regularity theorem*[1] for linear, second-order elliptic PDEs.

Lemma 2. *[Elliptic Regularity Theorem, Ref. [1] Section 6.3 Theorem 5] Let m be a nonnegative integer and assume $\mathbf{b}(x) \in C^{m+1}(\Omega)$ and $f \in H^m(\Omega)$, the Sobolev space of function satisfying $\|f\|_{H^m(\Omega)} < \infty$ where $\|\cdot\|_{H^m(\Omega)}$,*

$$\|u\|_{H^m(\Omega)} := \left(\sum_{|\alpha| \leq m} \|D^\alpha u\|^2_{L^2(\Omega)} \right)^{1/2} \tag{11}$$

and

$$D^\alpha := \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \ldots \partial x_1^{\alpha_d}} \quad , \quad |\alpha| = \alpha_1 + \cdots + \alpha_d \,. \tag{12}$$

Now, suppose u solves the second order linear boundary value problem

$$\begin{aligned} (\Delta + \mathbf{b}(x) \cdot \nabla) u &= f, \quad \text{in } \Omega \\ u &= 0 \quad \text{in } \partial\Omega, \end{aligned} \tag{13}$$

where the boundary $\partial\Omega \in C^{m+2}$*. Then, the solution* $u \in H^{m+2}(\Omega)$*, and we have the estimate that*

$$\|u\|_{H^{m+2}(\Omega)} \leq C\left(\|f\|_{H^m(\Omega)} + \|u\|_{L^2(\Omega)}\right) \tag{14}$$

for some constant C *depending only on* m*,* Ω*, and* $\mathbf{b}$*.*

Given a smooth $\mathbf{b} = \nabla V$ and well-behaved right hand side forcing function f over a domain with smooth boundary, we are guaranteed a solution to the PDE that is also smooth, so that the estimate given in Lemma 1 provides a useful bound on the norm of the largest derivative of the solution. Using high-precision schemes on a quantum computer may seem counter intuitive at first, as we can represent exponentially many grid points with a linear number of qubits. However, in many cases the underlying differential operator has a norm or condition number dependence that is polynomial in the number of grid points N. Therefore, simply using exponentially many grid points is not an effective strategy as the cost to perform a linear solve or time-stepping scheme will go as a polynomial of N, therefore it is unlikely for quantum computers to achieve an exponential speedup with respect to the number of grid points in realistic applications. By contrast, a high-precision scheme only increases the total norm and conditioning by a constant, while exponentially decreasing the total number of grid points needed to approximate the operator to a given accuracy. Therefore, it is useful to consider the higher-order schemes for approximating the differential operator with finite differences on a quantum computer. When considering high-dimensional applications, the advantages of using a high-precision scheme on a quantum computer are more pronounced.

In many spatial dimensions and for multiple particles, the operators can be constructed using a one-local tensor product structure. That is, the differential operator in d spatial dimensions (with each dimension satisfying the same boundary conditions and number of grid points) can be represented as a direct sum over d copies of the $d = 1$ case

$$\mathcal{D} = \sum_{i\in[d]} \mathcal{D}_i, \tag{15}$$

where

$$\mathcal{D}_i = I_N^{\otimes i} \otimes D \otimes I_N^{\otimes(d-i-1)}. \tag{16}$$

Although the dimension of each matrix is $O(N^d)$, the total matrix norm scales linearly with d through a straightforward application of the triangle inequality applied to the above relations. This construction is readily generalized to the case where the dimensions have different boundary conditions and different numbers of nodes, as long as the boundary conditions are consistent and can be expressed in terms of each coordinate in a separable manner. We similarly define the second derivative scalar operator in d dimensions by

$$\mathcal{L} = \sum_{i\in[d]} \mathcal{L}_i, \tag{17}$$

where

$$\mathcal{L}_i = I_N^{\otimes i} \otimes L \otimes I_N^{\otimes(d-i-1)}. \tag{18}$$

In the situation of many particles, η, occupying positions in d-dimensions, the finite difference operators satisfy a similar generalization as the continuum many-particle differential operators. In this case, each particle is associated with an N^d-dimensional subspace of the $N^{d\eta}$-dimensional Hilbert space, where the particle-localized differential operators for particle j and direction i satisfy

$$\begin{aligned} \mathcal{D}_i^j &= I_{N^d}^{\otimes j} \otimes \mathcal{D}_i \otimes I_{N^d}^{\otimes(\eta-j-1)} \\ \mathcal{L}_i^j &= I_{N^d}^{\otimes j} \otimes \mathcal{L}_i \otimes I_{N^d}^{\otimes(\eta-j-1)}. \end{aligned} \tag{19}$$

Therefore, the block diagonal matrix encoding all of the first and second derivatives can be expressed as

$$\begin{aligned} \boldsymbol{D} &= \sum_{j\in[\eta]} \sum_{i\in[d]} \mathcal{D}_i^j \otimes |ij\rangle \langle ij| \\ \boldsymbol{L} &= \sum_{j\in[\eta]} \sum_{i\in[d]} \mathcal{L}_i^j \otimes |ij\rangle \langle ij| \,, \end{aligned} \tag{20}$$

and the associated scalar operators can be obtained by computing the partial trace over the indices i and j.

2.3 Encoding into quantum states

We consider an encoding where each basis state is associated with a particular location on the lattice. In each dimension we have that the N grid points are associated with $n \equiv \lceil \log(N) \rceil$ qubits. Therefore in d dimensions, we need dn qubits to represent the state space of a single particle, and ηdn qubits to represent η particles in d dimensions. As an example, we can define a single particle position basis state in d-dimensions as the tensor product

$$\left|\boldsymbol{\alpha}^0\right\rangle = \left|\alpha_0^0\right\rangle \left|\alpha_1^0\right\rangle \cdots \left|\alpha_{d-1}^0\right\rangle, \tag{21}$$

where $\alpha_i^0 \in [N]$ and $i \in [d]$. The elements in this tensor product space are evaluated corresponding to the indexing relations

$$\boldsymbol{\alpha}^0 \in [N]^d \to \sum_{i=0}^{d-1} \alpha_i^0 N^i \in [N^d], \tag{22}$$

where $[N]^d$ is the set of d-tuples with entries in $[N]$.

Similarly, for η-particles in d-dimensions, we represent the state space as

$$|\boldsymbol{\alpha}\rangle = \left|\boldsymbol{\alpha}^0\right\rangle \left|\boldsymbol{\alpha}^1\right\rangle \cdots \left|\boldsymbol{\alpha}^{\eta-1}\right\rangle, \tag{23}$$

where $\boldsymbol{\alpha}^i \in [N]^d$ and $i \in [\eta]$. We treat $|\boldsymbol{\alpha}\rangle$ as an arbitrary basis state in our many-body Hilbert space.

Furthermore, we can map any function $f : \mathbb{R}^{\eta d} \to \mathbb{R}$ to a corresponding state on the lattice. Therefore upon discretization, the function f obtains the following representation as a vector

$$|f\rangle = \sum_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) \, |\boldsymbol{\alpha}\rangle \,, \tag{24}$$

which would need to be suitably l^2-normalized in order to correspond to a valid quantum state. Similarly, a function $f : \mathbb{R}^{\eta d} \to \mathbb{R}$ corresponding to the operation of scalar multiplication is implemented as the diagonal matrix

$$\mathbf{F} = \sum_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) \, |\boldsymbol{\alpha}\rangle \langle \boldsymbol{\alpha}| \,, \tag{25}$$

which to be implemented as a block encoding will need to be normalized by the factor $\max_{\boldsymbol{\alpha}} |f(\boldsymbol{\alpha})|$, to ensure $\|\mathbf{F}\|_1 \leq 1$.

2.4 Discrete boundary value problems

We focus on the three major kinds of boundary conditions encountered in physical problems, namely Dirichlet, Neumann, and Robin conditions. The Dirichlet condition specifies the value of the solution at particular subsets of the domain, the Neumann condition specifies the value of the derivative of the solution at particular subsets of the domain, and the Robin condition specifies a weighted linear combination of the value and the derivative of the function. Using a discretization of $[0,1]$ into $N = 1/h$ grid points we can express the linear system corresponding to the Dirichlet boundary value problem as follows. For simply connected domains where the boundary conditions are specified on the end-points, we can "fold in" the boundary value problem into the interior to obtain a higher-order accuracy scheme. Letting a and b be arbitrary real-valued constants describing the boundary values, and applying this approach to the Dirichlet boundary value problem can be expressed as the symmetric linear system

$$\frac{1}{h^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{pmatrix}\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} f_0 - \frac{a}{h^2} \\ f_1 \\ \vdots \\ f_{N-2} \\ f_{N-1} - \frac{b}{h^2} \end{pmatrix}. \tag{26}$$

Using a and b to also refer to the boundary values for the Neumann condition, i.e. $u'(0) = -a$ (where the minus sign reflects the convention of outward facing normal vectors) and $u'(1) = b$, we extend the system by two grid points on either end and we obtain the system

$$\frac{1}{h^2}\begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & \cdots & -1 & 1 \end{pmatrix}\begin{pmatrix} u_{-1} \\ u_0 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} \frac{a}{h} \\ -f_0 \\ \vdots \\ -f_{N-1} \\ -\frac{b}{h} \end{pmatrix}, \tag{27}$$

and, more generally, a Robin condition of the form $au(0) + bu'(0) = -\alpha$ and $cu(L) + du'(L) = \beta$ with $\alpha, \beta \in \mathbb{R}$ can be expressed by

$$\frac{1}{h^2}\begin{pmatrix} 1 - ah/b & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & \cdots & -1 & 1 + ch/d \end{pmatrix}\begin{pmatrix} u_{-1} \\ u_0 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} \frac{\alpha}{b} \\ -f_0 \\ \vdots \\ -f_{N-1} \\ -\frac{\alpha}{d} \end{pmatrix}. \tag{28}$$

The extension to so-called "mixed" boundary conditions, where different types of boundary conditions are specified on independent subsets of the domain is straightforward. In $d = 1$ for example, we could enforce a mixed boundary value problem with Dirichlet condition on the left-most endpoint and Neumann condition on the right-most

endpoint by adjusting the periodic matrix to match the Dirichlet matrix in the first row and the Neumann matrix in the last row. However, one difficulty of applying boundary conditions using the above approach is that it is unclear how to approximate them to high-precision, and so the above schemes do not readily generalize to high-precision case as easily as the periodic case. We address this concern in more detail in sec 3.3 when we consider high-order schemes for Dirichlet and Neumann boundary value problems.

Since we assume that the boundary conditions can be described in each dimension independently, the corresponding discretization can be constructed through the local tensor product structure we discuss in 2.2, where each $1d$ matrix encodes a boundary value problem similar to those given above. That is, for each dimension we can construct an operator taking the form of one of the matrices in (26)-(28) including the specified boundary conditions, then we can combine the operators from each dimension with a direct sum to represent the entire system matrix with the desired boundary conditions overall. We focus on the particular case of block encoding the matrix operator in d dimensions subjected to various boundary conditions and assume access to an efficient quantum circuit that prepares normalized versions of the right-hand-side vectors in (26) -(28).

2.5 Block encoding and LCU implementation

Block encoding is an access model for matrix operations on a quantum computer. A block encoding is characterized by 3 parameters, namely, the subnormalization factor λ, the number of ancilla qubits q, and the accuracy of the block encoding ϵ.

Definition 2 ((λ, q, ϵ) Block Encoding)**.** *For an n qubit matrix A, we say that the $n+q$ qubit unitary matrix U_A is a (λ, q, ϵ) block encoding of A if*

$$||A - \lambda(\langle 0|^{\otimes q} \otimes I_n) U_A (|0\rangle^{\otimes q} \otimes I_n)|| \leq \epsilon\,. \tag{29}$$

If we can implement the above unitary U_A exactly, then we call it a $(\lambda, q) - BE(A)$ block encoding of A.

In this work, we will assume that the block encoding of the matrix is obtained exactly, as the only place where an error can be introduced is in the preparation of a quantum state implementing the finite difference coefficients, which is essentially linear in the number of coefficients and subdominant. *Unless otherwise noted, we will abuse the standard notation used above and take ϵ to be the global truncation error of the finite difference scheme.*

We will use the method of linear combination of unitaries (LCU) [23] to implement the desired block encodings. LCU begins by assuming that one has a matrix operation A that has a decomposition into a sum of k unitary matrices:

$$A = \sum_{l=0}^{k-1} c_l U_l. \tag{30}$$

The goal is to embed the matrix operation given by A into a subspace of a larger matrix U_A which acts as a unitary on the system and ancilla registers. Because of the restriction to unitary dynamics, we require that $||A|| \leq 1$. Therefore we introduce a rescaling factor λ that upper bounds the $1-$norm of A, which we call the *subnormalization* factor so that $||A/\lambda|| \leq 1$. The block encoding U_A is expressed in matrix form as

$$U_A = \begin{pmatrix} A/\lambda & * \\ * & * \end{pmatrix}. \tag{31}$$

With LCU, it is straightforward to obtain an upper bound on $||A||$, since $\lambda = \sum_{l=0}^{k-1} |c_l|$. Then, scaling the LCU by $1/\lambda$,

$$\frac{A}{\lambda} = \sum_{l=0}^{k-1} \frac{c_l}{\lambda} U_l \equiv \sum_{l=0}^{k-1} \beta_l U_l, \tag{32}$$

we have a matrix that can be block encoded using LCU.

The unitary description of LCU is as follows. We use an ancilla register of $q = \lceil \log(k) \rceil$ qubits and define the oracle which "prepares" the LCU coefficients by

$$\text{PREP} \, |0\rangle^{\otimes q} \to \sum_{l=0}^{k-1} \sqrt{\beta_l} \, |l\rangle \,, \tag{33}$$

which defines a normalized quantum state and corresponds to the first column of a unitary matrix. Next, we implement the controlled-application of the desired unitaries using the "select" routine

$$\text{SEL} := \sum_{l=0}^{k-1} |l\rangle \langle l| \otimes U_l. \tag{34}$$

The block encoding can then be expressed as the $n + q$ qubit unitary

$$U_A := \left(\text{PREP}^\dagger \otimes I_n\right) \cdot \text{SEL} \cdot \left(\text{PREP} \otimes I_n\right). \tag{35}$$

The circuit diagram corresponding to this unitary transformation is shown in Fig. 1.

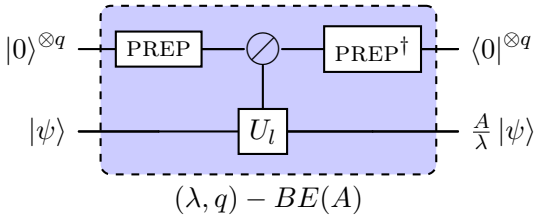


Figure 1: Circuit for linear combination of unitaries (LCU). PREP prepares the LCU coefficients β_l into a quantum state $\sum_{l=0}^{k-1} \sqrt{\beta_l} \, |l\rangle_q$, and the SEL operator applies the unitaries in the LCU in a controlled manner on the state of the ancilla register. The $\oslash$ notation is to be interpreted as a control on all states in the ancilla register. After uncomputing the ancilla register with PREP†, the block encoded operator is guaranteed to be applied to the system register upon measuring the ancillary system of $q = \lceil \log(k) \rceil$ qubits being measured in the all-zero state, where the notation $\langle 0|^{\otimes q}$ at the end of the ancilla wire refers to postselection on the outcome of the ancilla register.

2.6 Shift operators

An important class of unitary matrices that we will use regularly in this work are the *shift operators*. The shift operators are well studied [26, 50, 51], and we give a construction of the incrementer circuit in appendix A for those unfamiliar. The unitary shift operator $S \in \mathbb{R}^N$ performs a modular increment of the input state:

$$S : |i\rangle \mapsto |(i+1) \mod N\rangle \,. \tag{36}$$

and for an n qubit state $|i\rangle$, the increment by j operator satisfies

$$S^j : |i\rangle_n \mapsto |(i+j) \mod 2^n\rangle_n \,; \;\; j \in [2^n]. \tag{37}$$

These operators are 1-sparse and their matrix representations correspond to periodic bands about the diagonal. As an example, the modular increment operator S^1 corresponds to the matrix

$$S^1 = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}. \tag{38}$$

In Appendix A, we show that these operators can be implemented using $O(n)$ multi-controlled Toffolis, which can be expressed as a quantum circuit using $O(n^2)$ Toffoli gates [51]. Using an additional ancilla qubit, this cost can further be reduced to $O(n)$ using the techniques given in [52, 53], which is the scaling we will use in this paper. As the ancilla qubit can be reused for each shift operator we employ in our block encodings, we will ignore this additional global ancilla qubit in our cost estimates to simplify the discussion.

2.7 Quantum Signal Processing

Quantum signal processing (QSP) and its variants [27–29, 54] provides a systematic procedure for implementing a class of polynomial transformations to block encoded matrices. For Hermitian matrices A, the action of a scalar function f can be determined by the eigendecomposition of $A = UDU^\dagger$, as

$$f(A) := U \sum_i f(\lambda_i) |i\rangle \langle i| U^\dagger, \tag{39}$$

where λ_i is the ith eigenvalue, and $U|i\rangle$ is the ith eigenvector. When A is not Hermitian, a generalized procedure based on the singular value transform [29] can be used to implement these functions. To implement a degree d polynomial, QSP uses $O(d)$ applications of the block encoding and therefore its efficiency depends closely on the rate at which a given polynomial approximation converges to the function of interest, as well as on the circuit complexity required to implement the block encoding.

QSP uses repeated applications of the block encoding circuit to implement a polynomial of the block encoded matrix. We assume that A is a Hermitian matrix that has been suitably subnormalized by some factor λ so that $\|A/\lambda\| \leq 1$. QSP exploits the fact that the block encoding U_A can be expressed as a direct sum over one and two-dimensional invariant subspaces which correspond directly to the eigensystem of A, without knowledge of the eigendecomposition. For each eigenvector $|x\rangle$ of A, there is a 2×2 rotation matrix O_x representing the $2d$ subspace associated with $|x\rangle$. In turn this allows one to *obliviously* apply polynomial functions to the eigenvalues in each subspace, which builds up the polynomial transformation of the block encoded matrix A/λ overall. QSP is characterized by the following theorem, which is a rephrasing of Theorem 4 of Ref. [27].

Lemma 3 (Quantum Signal Processing (Theorem 7.21 [25]))**.** *There exists a set of phase*

factors $\mathbf{\Phi} := (\phi_0, \ldots, \phi_d) \in \mathbb{R}^{d+1}$ *such that*

$$\begin{aligned} U_\phi(x) &= e^{i\phi_0 Z} \prod_{j=1}^{d} [O(x) e^{i\phi_j Z}] \\ &= \begin{pmatrix} P(x) & -Q(x)\sqrt{1-x^2} \\ Q^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix}, \end{aligned} \tag{40}$$

where

$$O(x) = \begin{pmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{pmatrix}, \tag{41}$$

if and only if $P, Q \in \mathbb{C}[x]$ *satisfy*

1. $deg(P) \leq d, deg(Q) \leq d-1$,
2. P *has parity* $d \mod 2$ *and* Q *has parity* $d-1 \mod 2$, *and*
3. $|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1 \ \ \forall x \in [-1, 1]$.

Given a scalar function $f(x)$, we can use QSP to approximately implement $f(A/\lambda)$ by using the block encoding of A, a polynomial approximation to the desired scalar function, and a set of phase factors corresponding to the approximating polynomial. If f is not of definite parity, we can use the technique of linear combination of block encodings to obtain a polynomial approximation to f that is also of indefinite parity. This corresponds to implementing QSP for the even and odd parts respectively and then using an additional ancilla qubit to construct their linear combination. The last requirement, that $P(x) + (1 - x^2)|Q(x)| = 1$ for every x, is the most restrictive and requires normalization of the desired function, which can be severe in some cases. However, the polynomial P can be specified independent of the polynomial Q so long as $|P(x)| \leq 1 \ \ \forall x \in [-1, 1]$ and $P(x) \in \mathbb{R}[x]$ (see Corollary 5 of Ref. [29]), which is indeed the case for the algorithms presented in this work.

We note that there are efficient algorithms for computing the phase factors of QSP for very high degree polynomials. Algorithms for finding phase factors for polynomials of degree $d = O(10^7)$ have been reported in the literature (see e.g. [54, 55]) and are surprisingly numerically stable even to such high degree. Given that it is straightforward to obtain polynomial approximations to scalar functions and to obtain the corresponding phase factors for QSP, our explicit block encodings combined with the QSP framework immediately provides a nearly fully explicit circuit description of the block encodings in this work.

3 Block encoding of Laplacian with boundary conditions

In this section, we will provide constructions for the block encoding of the Laplacian. Throughout this section, we will assume that the overall scaling constant of $1/h^2$ has been multiplied through, and is enforced in the state preparation portion of the algorithm. Although this greatly improves the subnormalization factor, it may induce additional complexity in the preparation of the initial state. We will first briefly review the block encoding of the Laplacian with periodic boundary conditions using a linear combination of shift operators in 3.1. We then show how to obtain a block encoding of the periodic

operator in d dimensions. We then turn to the implementation of non-periodic boundary conditions in 3.2 and show how the construction for the periodic operator can be efficiently updated to represent Dirichlet, Neumann, and Robin boundary value problems on generalized and non-simply connected rectangular domains in d-dimensions using a low-order central difference scheme. We then show how the Dirichlet and Neumann boundary value problems can be discretized using a high-precision scheme using a domain extension in 3.3. Finally, in 3.4 we address the block encoding of differential operators with boundary data on irregular domains that cannot be described as a simple Cartesian product in d dimensions.

3.1 Periodic Boundary

For the Laplacian with a periodic boundary condition, approximated with a 3-point central-difference stencil with $d = 1$, the discrete Laplacian matrix will take the form

$$L = \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -2 & 1 \\ 1 & 0 & 0 & \dots & 1 & -2 \end{pmatrix}. \tag{42}$$

For $d = 1$, the Laplacian term Δ with periodic boundary and 3-point finite difference approximation can be expressed as the linear combination of shift matrices introduced in 2.6. It is straightforward to show that the 3-point stencil finite difference scheme applied to the continuum Laplacian operator $\mathcal{L}$ with periodic boundary conditions can be expressed as the linear combination

$$L = S^{-1} - 2I + S^{1}. \tag{43}$$

In general for a $2a+1$ point finite difference stencil, the periodic Laplacian can be expressed with $2a$ shift matrices, namely $\{S^{-1}, S^{-2}, \dots, S^{-a}\}$, $\{S^{1}, S^{2}, \dots, S^{a}\}$ and an identity matrix. The block encoding for the high-order scheme for approximating the Laplacian with periodic boundary conditions is characterized by the following theorem.

Theorem 1 (Block Encoding Periodic Laplacian)**.** *Using the shift operators and the identity matrix, an $p = 2a + 1$ point finite difference stencil with N discretization points can be block-encoded with a linear combination of p shift matrices, each of which can be implemented with $O(\log(N))$ Toffoli or simpler gates for a total of $\widetilde{O}(p \log(N))$ Toffoli or simpler gates.*

Proof. The central finite difference approximation of the second derivative of order a has $2a + 1$ coefficients that take on the form given in (9). Using the central difference formula for the second derivative, we may write

$$L \approx \frac{1}{h^2} \sum_{j=-a}^{a} r_j S^j + O(N^{-2a}), \tag{44}$$

we immediately obtain a decomposition of the central finite difference approximation of the Laplacian into a linear combination of unitary matrices. We choose λ equal to the

1-norm of the coefficients, namely

$$\lambda = h^2 \sum_{j=-a}^{a} |r_j|. \tag{45}$$

Furthermore, by use of the inequality $(a-j)!(a+j)! \geq (a!)^2$, for $0 \leq j \leq a$ and taking the limit as $a \to \infty$, the sum is given by a Basel problem and consequently we can bound the coefficients in the sum by a constant

$$\frac{\lambda}{h^2} = \sum_{j=-a}^{a} |r_j| \leq \frac{2\pi^2}{3}, \tag{46}$$

as shown in Lemma 6 of Ref. [40]. Since the construction of Ref. [52] promises that each of the shift operators can be implemented with $O(n)$ Toffoli and T gates with a single ancilla qubit that can be recycled, the Laplacian with periodic boundary conditions can be block encoded to precision $\epsilon \sim N^{-p+1}$ with gate cost $\widetilde{O}(np)$ Toffoli or simpler gates gates, where logarithmic factors arise from the compilation of the rotation angles in the LCU into Clifford + T gates, employing $\lceil \log(p) \rceil + 1$ ancilla qubits. □

This LCU of the periodic Laplacian is the core of the block encodings we will construct for the Laplacian with various boundary conditions and in arbitrary dimension. The remainder of this section will show how this LCU can be efficiently updated using $O(1)$ additional quantum resources to block encode many other kinds of boundary conditions for $d > 1$ dimension.

3.1.1 d dimensional periodic boundary

We assume that the d-dimensional boundary corresponds to a $d-$dimensional torus, so that the periodic boundary condition can be specified for each dimension independently. In this case, the corresponding operator will satisfy the local tensor product structure discussed in Sec. 2.3, so that the $d-$dimensional Laplacian can be constructed as the direct sum of $1d$ Laplacian matrices. With this structure, we can obtain an $(O(d), \log(d) + \log(p), \frac{d}{N^{p-1}})-$ block encoding of the approximate periodic Laplacian operator.

We can construct the block encoding of this matrix as follows. Let

$$\text{PREP}_r |0\rangle = \sum_j \sqrt{\frac{|r_j|}{\lambda}} |j\rangle , \tag{47}$$

where $\lambda = \sum_j |r_j| \leq \frac{2\pi^2}{3}$. In practice, $p = O(1)$ and so the sign information of the coefficients is assumed to be easily accessible and can efficiently be included in the unitary. Therefore, in the state preparation we take the r_j's to be non-negative and will incorporate the sign information into the unitaries by multiplying them by an appropriate phase. Now, we let

$$\text{SEL} = \sum_{i=-a}^{a} \text{sign}(r_i) S^i \otimes |i\rangle \langle i| . \tag{48}$$

Then, we introduce the notation S_l^j to refer to the increment by j operator applied to register l, acting trivially on the other registers. In circuit form, this corresponds to the

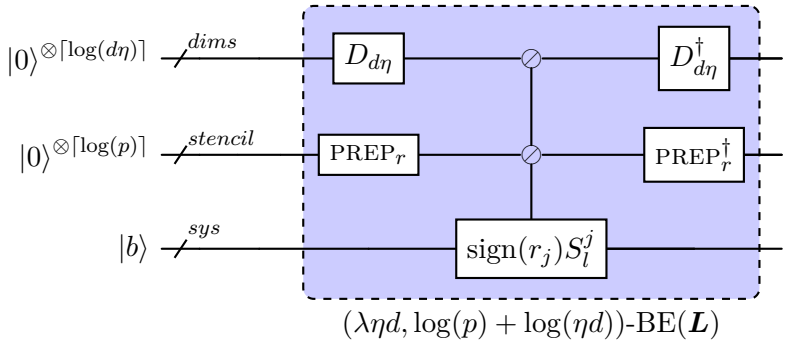
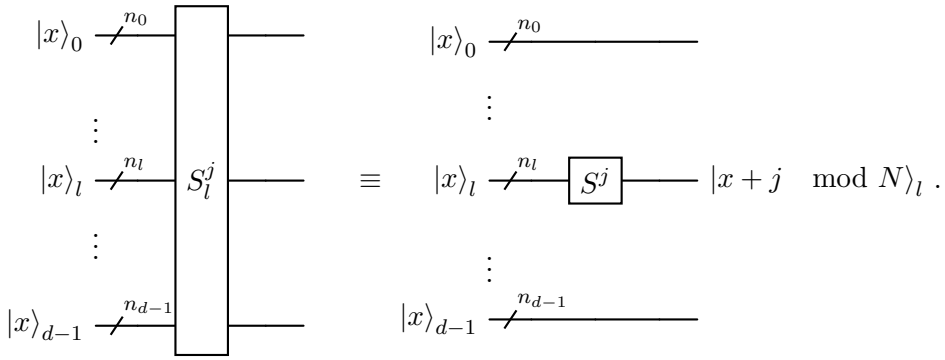


Figure 2: Quantum circuit for LCU of Laplacian $\boldsymbol{L}$ with periodic boundary conditions in d spatial dimensions using a stencil of arbitrary order $p = 2a + 1$ and for η particles. PREP is the unitary which prepares the unsigned finite difference coefficients, $\mathrm{sign}(r_j)r_j$ on the *coeff* register of size $O(\log(p))$ and D_d is the unitary which prepares the equal superposition over d states on the *dims* register of size $O(\log(\eta d))$. The system register *sys* holds ηd registers of n qubits representing a $N^{\eta d}$ dimensional state $|x\rangle$. S_l^j refers to the modular increment-by-j operator in register l and $\mathrm{sign}(r_j)S^j$, multiplies the overall unitary by a phase ± 1. We use the $\oslash$ convention to mean the application of a unitary controlled on any $|j\rangle$ in the *stencil* register for every $j \in [\eta]$. Similarly, for the *dim* register, $\oslash$ controls on $|l\rangle$ for every $l \in [d]$.

following equivalence:



Now, we observe that the d-dimensional system can be represented by the following LCU:

$$\begin{aligned}
&\frac{1}{\lambda d}\sum_{i\in[d]}\sum_{j=-a}^{a} r_j S_i^j \\
&= \frac{1}{\lambda d}\sum_{i\in[d]}\sum_{j=-a}^{a} r_j I^{\otimes i}\otimes S^j \otimes I^{\otimes(d-i-1)} \\
&= \frac{1}{\lambda d}\sum_{i\in[d]} \mathcal{L}_i^{(d)} \\
&= \frac{1}{\lambda d}\mathcal{L},
\end{aligned}$$

where $\mathcal{L}_i^{(d)} = \sum_{j=-a}^{a} r_j S_i^j$ is the one-dimensional periodic Laplacian on register i, acting as the identity on all other registers. Note that this block encoding does not use controlled applications of the block encodings of the $d = 1$ dimensional operator.

By preparing the uniform superposition over d states, providing indices to control which register we apply the shifts on, we can implement the d dimensional Laplacian

matrix with comparable efficiency to the $1d$ case, because the number of additional ancilla qubits is only an additional $O(\log(d))$ larger than the $1d$ case, the circuit depth is $O(dpn)$ and the subnormalization is $d\lambda$. The quantum circuit implementing this block encoding is shown in Fig. 2. The generalization to η particles in d dimensions follows the same pattern as the generalization from 1 to d-dimensions.

3.2 Dirichlet, Neumann, and Robin Boundary conditions

In this section, we show how Dirichlet, Neumann, and Robin boundary conditions can be implemented using the low order scheme to enforce common boundary value problems that we introduced in section 2. These block encodings of the Laplacian with non-periodic boundary conditions do not enjoy the same convergence rates as the higher-order finite difference scheme used in the periodic operator. This is because the evaluation of the differential operator near the boundary restricts the order of the finite difference stencil that can be used, which lowers the global convergence rate. Nevertheless, these schemes are standard in classical methods for the numerical solution of differential equations, and so we write the circuits for these block encodings explicitly in this section, which may be useful for simple applications. Later, we will show how Neumann and Dirichlet conditions can be efficiently implemented with high-precision using an extended domain and forcing function.

In $d = 1$ the boundary conditions only affect the rows of the corresponding matrix which describe the behavior of the solution on the boundary. When the boundary value problem is on the extremal points of an interval, upon discretization, the boundary conditions change the matrix elements of the periodic Laplacian on only the first and last row. Here we demonstrate how to implement this change for the simple case of a Dirichlet boundary condition that is specified on both endpoints of the domain. Recall from (26) that up to a global scaling factor, the 3-point stencil for the Laplacian with Dirichlet boundary corresponds to the tridiagonal matrix

$$L_D = \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{pmatrix}.$$

We can construct this matrix with a similar LCU to the periodic case, but with the addition of reflection operations. This is based on the simple observation that a sum or difference of a reflection and an identity matrix forms the projector onto the reflected subspace or its complement. Notice that we may express L_D as,

$$L_D = -2I + \Pi_0^{\perp} S^{-1} + \Pi_{N-1}^{\perp} S^{1}.$$

The projectors $\Pi_0^{\perp} = I - |0\rangle\langle 0|$ and $\Pi_{N-1}^{\perp} = I - |N-1\rangle\langle N-1|$ can be expanded using a linear combination of the reflections $R_0 = I - 2\Pi_0$, $R_{N-1} = I - 2\Pi_{N-1}$ and the identity as,

$$L_D = -2I + \frac{1}{2}(R_0 + I) S^{-1} + \frac{1}{2}(R_{N-1} + I) S^{1}. \tag{49}$$

This LCU can be immediately implemented as an $(\lambda_D = 4, m = 3, \epsilon = O(N^{-2}))$ block encoding of the Laplacian with Dirichlet boundary conditions on N grid points. The

quantum circuit implementing this block encoding is given in Fig. 4, and involves only a constant number of additional gates and only a single additional ancilla qubit.

The Neumann and Robin boundary value problems are similar. For the on dimensional case, the matrices we need to block encode take the form

$$L_N = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 1 & -1 \end{pmatrix}, \quad L_R = \begin{pmatrix} -1 - ah/b & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & \cdots & 1 & -1 + ch/d \end{pmatrix}, \tag{50}$$

Here we need to adjust three matrix elements on the boundary rows. Using a similar approach with a linear combination of shifts and shifts multiplying reflections, we can also construct the matrix corresponding to the Neumann boundary value problem. For the Neumann case, corresponding LCU can be expressed

$$L_N = \frac{-3}{2} I - \frac{1}{2} R_0 R_{N-1} + \frac{1}{2} \left(R_0 S^{-1} + S^{-1} \right) + \frac{1}{2} \left(R_{N-1} S^1 + S^1 \right). \tag{51}$$

This constructs a $(\lambda_N = 4, m = 3, \epsilon = O(N^{-1}))$ block encoding of the Neumann boundary value problem. The Robin condition can now be encoded by adjusting the the first and last diagonal entries of the block encoding of the Neumann matrix. This can be accomplished by adding the following LCU terms

$$-\frac{ah}{2b} \left(I - R_{N-1} \right) + \frac{ch}{2b} \left(R_0 - I \right).$$

Therefore, the Robin LCU can be expressed

$$L_R = \frac{(c-a)h - 3b}{2b} I + \frac{ah}{2b} R_{N-1} + \frac{ch}{2b} R_0 - \frac{1}{2} R_0 R_{N-1} + \frac{1}{2} \left(R_0 S^{-1} + S^{-1} \right) + \frac{1}{2} \left(R_{N-1} S^1 + S^1 \right). \tag{52}$$

This results in an $(\lambda_R \leq 4 + \frac{c+a}{Nb}, m = 4)$ block encoding of the linear system corresponding to Robin boundary condition. In many cases with Neumann or Robin conditions, the stability of the corresponding linear system may have a non-trivial dependence on N, typically $O(N)$, which will lower the global truncation error of the scheme to $\epsilon \sim O(N^{-1})$. Furthermore, the stability of the system depends on the coefficients a and b as well as the associated boundary values. Additionally, since there are terms with different powers of N in the summation, the factor of N^2 cannot be viewed as a global scaling factor and must be incorporated into the coefficients. Although, the subnormalization factor for the Neumann/Robin case becomes is similar to the Neumann case $\lambda_R \leq \lambda_N + O(1/N)$, the preparation of the LCU coefficients is more involved.

This scheme can be straightforwardly extended to dimension $d > 1$ using a similar circuit to that in Fig. (2), as long as the underlying boundary conditions are separable. In addition, this approach also allows for the evaluation of boundary conditions imposed, for example, on the interior of the grid. To illustrate this, consider the following example. Let $[0, L]^d$ be the domain and $A = A_0 \cup A_1 \cdots \cup \cdots A_{d-1}$ be a set of points corresponding to the boundary in each dimension so that Cartesian product $(A_0 \times A_1 \times \cdots \times A_{d-1}) \subset [0, L]^d$ corresponds to a generalized rectangle in d-dimensions. Because each of the faces of the rectangle are orthogonal, the projection of the rectangle onto the axes in each dimension

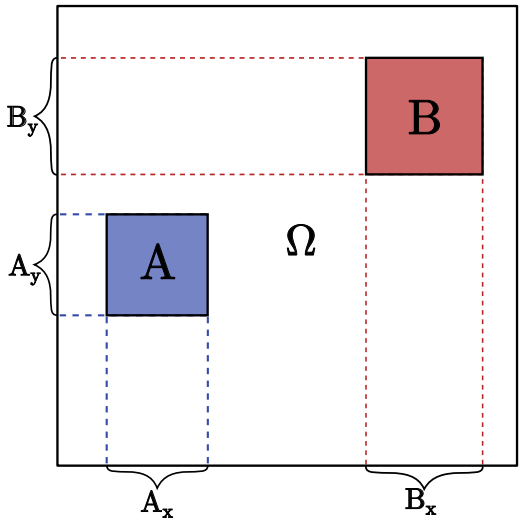


Figure 3: An example of a non-simply connected domain. For this portion of the discussion, we assume that the boundary conditions are given in each dimension independently so that the projection of the shapes of the boundary onto the principal axes is sufficient to completely characterize the hole. We solve the Dirichlet problem corresponding to the holes A and B by solving the $d = 1$ Dirichlet problems on $A_x \cup B_x$ and $A_y \cup B_y$, assuming the solution takes a constant value on the interior of the holes. Because the underlying operator is still separable, knowledge of the projections of A and B and the block encodings for the Dirichlet problem for $d = 1$ are sufficient to block encode the Laplacian on this non-simply connected domain.

is sufficient to describe the shape. Exploiting the separable structure, in each dimension i, we can block encode the Dirichlet boundary value problem with the LCU

$$\mathcal{L}_i^{(d)} = 2I - \frac{1}{2}\left(S^{-1}R_{A_i+1} + S^{-1} + S^1 + S^1 R_{A_i-1}\right), \tag{53}$$

with $A_i + j \equiv \{j + k \mod N : \forall k \in A_i\}$ and $R_A = I - 2\sum_{j\in A} |j\rangle\langle j|$. This LCU produces a matrix that approximates the Laplacian on $[0, \min(A_i)) \cup (\max(A_i), L]$ and is proportional to the identity matrix on grid points supported in $[\min(A_i), \max(A_i)]$, so that the prescribed value on the interior can be enforced by simply placing the boundary values in the elements of right hand side vector that correspond to the grid points on the boundary. The Neumann boundary value problem can be block encoded similarly.

Because the LCU involves a linear combination of shift operators and shift operators multiplying reflection matrices, which have known circuit decompositions into one and two qubit gates, we can immediately obtain a decomposition of the LCU into one two qubit gates. In particular, the shift operators have been discussed many times in the literature, and an optimized implementation is provided in Ref. [52], using an additional ancilla qubit and $n - 1$ Toffoli gates. The reflection matrices can be implemented by n-qubit control-Z gates and single-qubit Pauli gates, which can also be implemented by $O(n)$ Toffoli gates. Therefore, if the boundary conditions are set on the exterior grid points and satisfy the above assumptions, we can use the periodic extension to enforce the Neumann and Dirichlet to high precision in d-dimensions. In particular, using the 3 point scheme in d-dimensions the expected number of shift and reflection operators needed is $O(2d)$ to block encode the Neumann and Dirichlet matrices as opposed to $O(d)$ needed for the periodic operator. Since each shift and reflection operation requires at most $O(n)$ Toffoli gates to implement, the total Toffoli count for the periodic case is $O(dn)$ and the case with Dirichlet and Neumann boundary is $O(2dn)$.

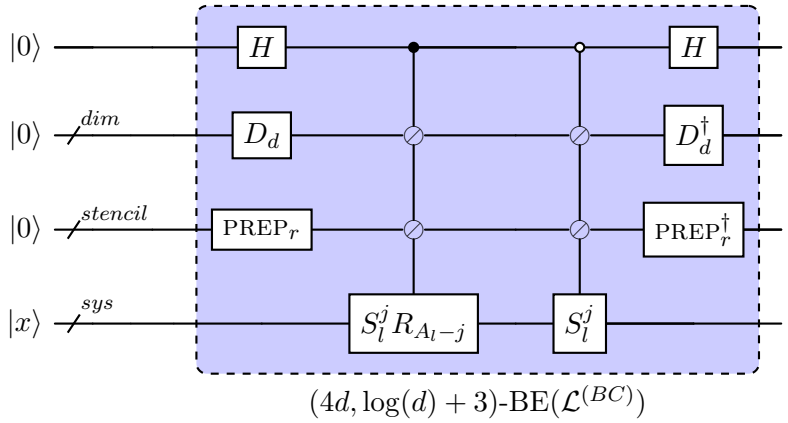


Figure 4: Quantum circuit for LCU of second derivative operator with boundary conditions on a simply or non-simply connected domain in d-dimensions using low-order stencil $\mathcal{L}^{(BC)}$. The subnormalization $4d$ results from $\lambda = 4$ for the 3 point stencil, which we then combine d times. Here we use the notation PREP as the unitary which prepares the finite difference coefficients on the *stencil* register and D_d as the unitary which prepares the equal superposition over d states on the *dim* register where S_l^j refers to the modular increment-by-j operator in direction l. We use the $\oslash$ convention to mean controlled over every state. Appending an additional register of $\lceil \log(\eta) \rceil$ qubits and selecting over shift operators both in dimension and particle indices will implement the η-particle Laplacian over d-dimensions.

The total number of Toffoli or simpler gates is $O(dn)$, with $\log(d) + 1$ ancilla qubits required to block encode the d-dimensional Laplacian with periodic boundary conditions on a rectangular grid. Replacing the periodic Laplacian with the Dirichlet block encoding will provide a block encoding of the d-dimensional Dirichlet problem. Doing this for η particles, we can simply append another register of size $\log(\eta)$ and proceed similarly. This will in turn require $O(dn\eta)$ Toffoli or simpler gates to implement and $O(\log(dp\eta))$ ancilla qubits.

We note that the above schemes for Dirichlet and Neumann/Robin boundary conditions are generically limited to second, $O(N^{-2})$, and first order, $O(N^{-1})$, accuracy with respect to the discretization, respectively. We can improve these orders of accuracy by using a high-order scheme, but this is complicated by the ambiguity of how to evaluate the high-order scheme near the boundary. The introduction of ghost points is a common technique for handling Neumann boundary value problems, but it is unclear how to extend the ghost point method to arbitrary order. However, if we can uniquely and smoothly extend the PDE onto a larger domain, then we can use a higher order method to evaluate the boundary value problem. With some restriction on the domain and boundary conditions, this can be accomplished, at least in theory, using the method of images. Although the method of images also holds for the Robin boundary condition, it is more complicated than the Dirichlet and Neumann boundary conditions, so we will focus on their implementation. We now show how this can be efficiently implemented in practice on a quantum computer using a method of periodic extensions for the Dirichlet and Neumann case.

3.3 Neumann and Dirichlet boundary conditions via periodic extensions

One practical way to implement the method of images over a finite interval, say $[0, 1]$, is to use the even or odd extensions of the forcing function f onto $[-1, 1]$ and then discretize the corresponding differential equation over the enlarged interval. Although this doubles the number of grid points, on a quantum computer this can be implemented with the introduction of just a single additional qubit. This idea was first explored as part of [15],

but left to future work its detailed implementation. We provide one set of the required implementation details here. For this section, we will assume that the boundary value problem is homogeneous and that the corresponding extended forcing function $\widetilde{f}$ is smooth across the boundary.

From a geometric perspective, this can be viewed as embedding some system with specified non-periodic boundary conditions within a larger system having a periodic boundary. The degrees of freedom in the extended portion of the domain are constructed to produce the correct solution on the original domain of definition. This extended system allows the use of central difference schemes, even on boundary nodes, to obtain approximations at the boundary to the same precision as in the interior. Furthermore, constructing such an extension is straightforward since only the original operator, the domain of definition, and the prescribed boundary conditions are sufficient to uniquely extend the operator onto a new domain and to construct the appropriate periodic extension of the forcing function f. To proceed, we first define even and odd periodic extensions of the forcing function on the right hand side f.

Definition 3 (Even and odd periodic extensions)**.** *Given a function f defined over some compact interval, say $[0, L]$, $f : [0, L] \to \mathbb{R}$, we can define the even periodic extension to the interval $[-L, L]$ $\widetilde{f}_e : [-L, L] \to \mathbb{R}$*

$$\widetilde{f}_e(x) = \begin{cases} f(-x) & x \in [-L, 0) \\ f(x) & x \in [0, L) \end{cases} \tag{54}$$

so that $f(-L) = f(L), f(0) = f(2L)$. Similarly, we define the related odd periodic extension $\widetilde{f}_o : [-L, L] \to \mathbb{R}$ as

$$\widetilde{f}_o(x) = \begin{cases} -f(-x) & x \in [-L, 0) \\ 0 & x = 0 \\ f(x) & x \in [0, L) \end{cases} \tag{55}$$

so that $\widetilde{f}_o(-x) = -\widetilde{f}_o(x)$ is an odd function. In addition, we assume that $\widetilde{f}$ is smooth in $[-L, L]$, meaning that for any $p > 3$, and $x \in [-L, L]$, $\max_x |f^{(p)}(x)| < C$, for some constant C.

The assumption of smoothness over the extended domain is critical to obtaining theoretical guarantees of higher order accuracy at the boundary to achieve an higher-order globally accurate scheme. If such smoothness is not automatically guaranteed by the original problem definition, one can mollify the forcing function through a convolution with a smooth bump function centered at the boundary point to smooth connect the even and odd extensions of the problem. Then we simply solve the equation with the smoothed version of the periodically extended forcing function, accepting the small controllable error introduced from modifying the original function on the domain. Intuitively, by extending the domain of the original function and of the differential operator, the stencil will involve points extending across the boundary, causing interference between the solution on the original domain and that on the extended domain. The odd periodic extension enforces a kind of destructive interference of the solution near the boundary, while the even periodic extension enforces the destructive interference of the *derivatives* of the solution near the boundary. This behavior is related to why the Laplacian with Dirichlet and Neumann boundary value problems (BVPs) are diagonalized by the discrete sine and cosine transformations respectively. Extending this approach to Robin boundary conditions is less straightforward. In principle it should be possible, using, for example, the

approach found in [56], but it is unclear if this method can be efficiently implemented using a straightforward adjustment of the scheme we present for Dirichlet and Neumann BVPs.

For a nonhomogenous Dirichlet problem

$$\begin{aligned} \Delta[u](x) &= f(x); \quad 0 < x < L, \\ u(0) &= f(0), \\ u(L) &= f(L), \end{aligned}$$

we must first homogenize the system to apply the method of images. To homogenize the boundary value problem, we first introduce a new solution $V = u(x) - \frac{f(L)-f(0)}{L}x - f(0)$ to homogenize the original boundary value problem so that

$$\begin{aligned} \Delta[V] &= f(x); \quad 0 < x < L, \\ V(0) &= 0, \\ V(L) &= 0. \end{aligned}$$

Now, we consider the odd periodic extension of the function f about $x = 0$ and $x = L$

$$\widetilde{f}_o(x) = \begin{cases} -f(-x) & -L < x < 0 \\ 0 & x = 0 \\ f(x) & 0 < x < L \\ 0 & x = L. \end{cases} \tag{56}$$

In general, since $f(0) \neq f(L) \neq 0$, we may have the even or odd extension may be discontinuous or have discontinuous derivatives across the boundary that would lower the smoothness of the forcing function at the boundary, destroying the smoothness of the solution as promised by lemma 2 and thereby the global convergence rate of the high order scheme. In the case where f decays sufficiently quickly so that we need only to perform the periodic extension about a single boundary point, we may convolving f with a smooth bump function $\varphi_k(x) = e^{\frac{1}{(kx)^2-1}}$, for $k > 1$, to mollify the forcing function across the boundary. We then choose k large enough that we approximate the periodically extended function within the desired precision and solve the problem on the mollified forcing function given by,

$$\hat{f}(x) := \frac{1}{\mathcal{N}} \int_{-1}^{1} \varphi_k(x-y) f(y) dy, \tag{57}$$

where $\mathcal{N} = \int_{-1}^{1} \varphi_k(x)dx$ is a normalizing constant. For this scheme to be efficient, we require that $k = O(1)$, as the nth derivatives of the bump function go as $O(k^n)$. If k is large, one may need to balance of the order of the scheme with the smoothness of f across the boundary. However, we do not expect for k to be so large in practice. In Fig. 5, we demonstrate a particularly challenging example of periodically extending the exponential function across the origin. In the figure, we show close qualitative accuracy for mollifying constant $k = 5$ in the even extension case and $k = 10$ in the odd extension case. For the rest of this section, we will assume that smoothness across the boundary is guaranteed by the original problem since the mollification process can be efficiently performed classically if needed.

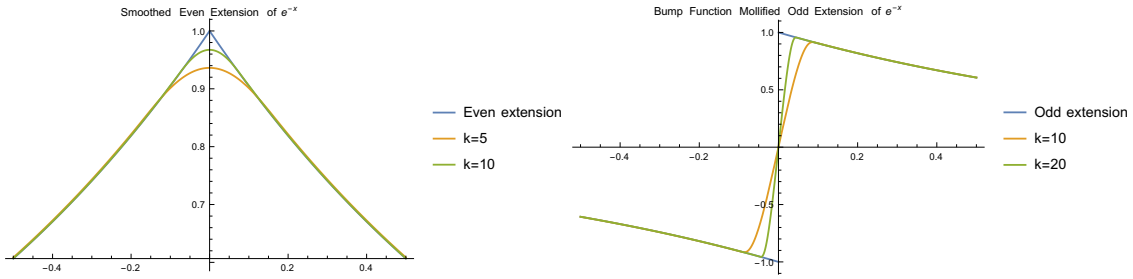


Figure 5: Smoothed even and odd extensions of an exponential forcing function $f(x) = e^{-x}$. This case represents a particularly challenging example, as the even extension of e^{-x} has a cusp at $x = 0$ and the odd extension of e^{-x} is discontinuous at $x = 0$. For the even extension, we convolve $\widetilde{f}_e(x)$ with $e^{\frac{1}{(kx)^2-1}}$ for $k = 5$ and $k = 10$ in the figure on the left. For the odd extension, we convolve $\widetilde{f}_o(x)$ with the same mollifying function using the mollifying constants $k = 10$ and $k = 20$ in the figure on the right. The convolution can be accomplished according to Eq. (57). These figures demonstrate how a mollified version of the periodic extension can be used to obtain an infinitely differentiable periodic extension of the original forcing function onto the extended domain, at the cost of some error that depends on mollifying constant k. This allows us to use a central difference scheme spanning the extended domain, similar to the standard periodic boundary condition case. This greatly improves the smoothness of the solution and the overall convergence rate, correspondingly. We provide numerical examples for the solution of periodically extended domains that are available on Github [47].

Now, consider the differential equation with respect to the linearly shifted solution given above

$$\begin{aligned} \Delta[V] &= \widetilde{f}_o(x), \quad x \in (-L, L) \\ V(-L) &= V(L) \end{aligned}$$

with periodic boundary conditions. Now, we notice that

$$\Delta[V](-x) = -f(x) = -\Delta[V](x)$$

and by linearity, we conclude that

$$\Delta[V](0) = \sum_{j=1}^{a} r_j((V(j) + V(-j)) + r_0 V(0) \sim 0 + O(h^p),$$

which implies $V(0) = O(h^p)$. Therefore, we can use the block encoding of the periodic Laplacian on $2N-2$ grid points applied to the smooth odd extension of f to obtain a block encoding of Dirichlet Laplacian which converges globally as $O(h^{p-1})$. This can then be implemented via the LCU (43) and correspondingly as an $(\lambda = O(1), \lceil \log(p) \rceil, \epsilon \sim N^{-p+1})$ block encoding of the Dirichlet operator.

The Neumann case is similar. Consider the periodic boundary value problem with an even function $\widetilde{f}$ about $x = 0$ and $x = L$:

$$\begin{aligned} \Delta[V] &= \widetilde{f}_e(x) \quad x \in (-L, L) \\ V(-L) &= V(L). \end{aligned}$$

Now suppose u is some particular solution to the above. Then,

$$\begin{aligned} u'(0) &\sim \sum_{j=1}^{a} c_j u(-j) - c_j u(j) + O(h^p) \\ &= O(h^p) \end{aligned}$$

satisfies the required boundary condition to high precision. A similar result can be found at $x = L$. Therefore, once again, this matrix can be block encoded using a high order scheme for the Laplacian satisfying periodic boundary conditions on the extended domain. However, in this case, the subnormalization factor will again depend on N, but the factor of N in the global truncation error arising from the stability of the linear system is also only $O(1)$ since the rows with Neumann conditions R satisfy $R|x\rangle = 0$ for every x in the row space of R. Therefore the overall global truncation error for the Neumann scheme is similar to that of the Dirichlet case $\epsilon \sim O(N^{-p+1})$.

Extending to d-dimensions with the periodic embedding is also straightforward whenever $f : \mathbb{R}^d \to \mathbb{R}$ can be expressed as a separable function of its arguments. We simply periodically extend each portion of the domain and the function corresponding to each dimension, and then constitute the direct sum to obtain the whole operator. Periodic extensions in d dimensions will increase the number of grid points by a factor of 2^d so that $N^d \to N' \sim O(2^d N^d)$, but this only comes at the cost of $\sim d\log(2N)$ qubits, which is essentially equivalent. However, for a fixed N, the periodic embedding technique promises $\epsilon \sim N^{-(p-1)}$ whereas the standard schemes only provide $\epsilon \sim N^{-(p-1)/2}$, assuming some high order scheme has been consistently implemented at the boundary, so that one would need at least *quadratically* more grid points to obtain the same accuracy as the periodic embedding scheme. Furthermore, it is much less clear how to extend the above constructions in an efficient manner to achieve the standard accuracy $O\left(N^{-(p-1)/2}\right)$ in the first place, as whole new coefficients would be needed on the boundary rows for the scheme. Finally, and perhaps most beneficially, the constructed matrix can be diagonalized by the quantum Fourier transform, allowing for faster algorithms to solve the underlying linear system or perform dynamical simulations [15, 57].

3.4 Block encoding operators on irregular domains

The main technique used to implement the above boundary conditions is to use a linear combination of a unitary with itself, multiplied by a reflection about the boundary nodes to zero out specific matrix elements. Unfortunately, our technique of constructing explicit block encodings relied on the assumption of separability to efficiently exploit the tensor product structure and so cannot be immediately applied in less structured domains. However, a very similar approach can be used for block encoding domains where an explicit tensor product structure is not apparent, if we can efficiently compute a membership oracle for the domain. We illustrate this by considering the following scenario. We have some $d-$dimensional physical domain Ω that is contained within some $d-$dimensional cube that we call the computational domain, i.e., there is some $L > 0$ such that $\Omega \subseteq \Omega_L^d \sim [0, L]^d$. This is similar to classical embedding methods which embed the domain of interest within a subset of a Cartesian domain. These techniques are a starting point for the numerical simulation of curved boundaries [58–61], which could be an avenue to explore in future work. We will discretize Ω_L^d as usual into N^d many grid points, and in addition, we will assume access to an oracle that can compute the following quantities

$$\textsc{dom}\,|x\rangle\,|0\rangle = \begin{cases} |x\rangle\,|0\rangle & \text{if } x \in \text{int}(\Omega) \\ |x\rangle\,|1\rangle & \text{if } x \in \text{bndry}(\Omega) \\ |x\rangle\,|2\rangle & \text{if } x \in \Omega_L^d \backslash \Omega. \end{cases} \tag{58}$$

We have used the shorthand notation $|x\rangle = |x_0\rangle\,|x_1\rangle \cdots |x_{d-1}\rangle$ to refer to the N^d-dimensional state vector indexing a grid point in Ω_L^d. We further assume that inclusion in the set Ω can be efficiently computed. Some examples where this can occur are i) if the function

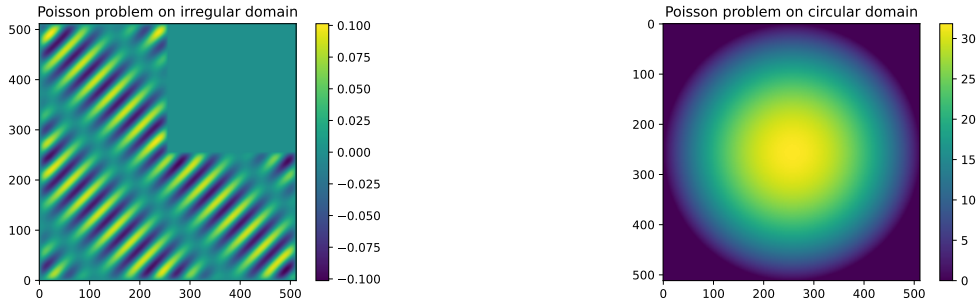


Figure 6: Here we demonstrate solving the obtained linear system using the projection method on an L-shaped and circular domain, two examples where the problem cannot be expressed as the direct sum over two one-dimensional domains. The linear system is constructed by first constructing the two-dimensional Laplacian matrix with periodic boundary conditions, and then multiplying on the left by a diagonal projection matrix containing 1 in entries corresponding to interior grid points. We then add to this matrix a projector onto the boundary to enforce a homogeneous Dirichlet boundary condition. We then add a projector onto the exterior portion, so that obtained linear system is not singular, placing zeros in the corresponding locations of the right hand side vector. For the L-shaped domain we a product of $\sin$ and $\cos$, and use a constant for the circular domain to generate the rhs vector. The code generating these plots can be found on Github [47].

parameterizing the boundary of Ω can be expressed as a smooth function or as a small $\widetilde{O}(1)$ family of piecewise smooth curves, or ii) if the boundary is given by the convex hull of a simple set, in which case such an oracle could be efficiently realized using the method of inequality testing [62]. The resulting block encoding corresponds to an operator with boundary conditions on a "rasterized" domain. Because this block encoding is still inherently restricted to a Cartesian grid, the DOM oracle could induce "jagged" edges when the domain is a smooth curve like a circle. Although it may be possible to choose N large enough to minimize this effect, this may introduce uncontrolled numerical errors which we will not account for here.

Now let L be a linear differential operator defined with periodic boundary conditions in each dimension on Ω_c^d, which is then discretized using standard finite difference methods, and let U_L be its (λ, m)-block encoding. Because the periodic boundary conditions are separable, this block encoding can be obtained by the methods given above. For a generic Robin boundary condition $au(x) + b\nabla u(x) \cdot \mathbf{n}(x) = f$, we may express the operator in difference form as $B = b\sum_{1\leq|j|\leq a} S^j c_j + aI = \mathbf{f}$. Let U_B be the (β, l)-block encoding of these boundary conditions. We can then implement these operations on the relevant subsets of the domain and sum them together to obtain the encoding of the desired operator on the physically relevant subset.

Now, we show how more general boundary conditions can be realized using this oracle and a phase kickback. Consider some arbitrary, suitably normalized initial starting state,

$$|x\rangle = \sum_{i\in[N^d]} x_i |i\rangle , \tag{59}$$

and append an additional register of 2 ancilla qubits $|b\rangle \to |b\rangle |00\rangle$. Then, applying the oracle DOM to $|b\rangle |00\rangle$ we find

$$\text{DOM} |x\rangle |00\rangle = |x_{int}\rangle |00\rangle + |x_\partial\rangle |01\rangle + |x_{ext}\rangle |10\rangle . \tag{60}$$

To obtain a block encoding of the operator on the interior, we will use a phase kickback oracle to cancel out matrix elements on the boundary and exterior. Since the boundary and

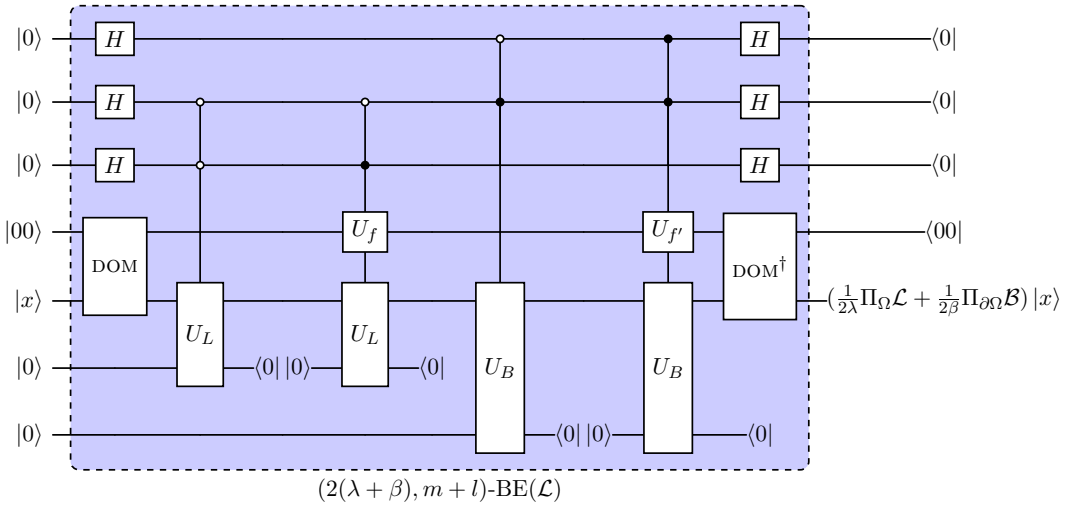


Figure 7: Quantum circuit to implement $(2(\lambda+\beta), m+l)-$block encoding of a finite difference approximation to the differential operator with arbitrary boundary conditions. DOM is an oracle which marks each point $|x\rangle$ in the $[N^d]$ with a two-bit label $|x\rangle|b_0, b_1\rangle$, with $'00'$ indicating interior points, $'01'$ indicating boundary points and $'10'$ indicating exterior points. Here, U_L is a block encoding of the differential operator defined with periodic boundary conditions on $[0, L]^d$ in matrix form and U_B is a block encoding of the Dirichlet, Neumann, or Robin boundary condition in matrix form. U_f implements the unitary transformation $|b_0, b_1\rangle \to (-1)^{b_0 \oplus b_1}|b_0, b_1\rangle$ and $U_{f'}$ implements $|b_0, b_1\rangle \to (-1)^{b_0 \oplus 1 \oplus b_1}|b_0, b_1\rangle$, to cause the summed operators to be projected onto the appropriate boundary modes. Because DOM is not restricted to marking points that correspond to a domain with orthogonal boundary faces, this procedure can be used to implement a broad family of boundary conditions.

exterior nodes are marked with a set bit, we can implement the desired sign flip by checking the parity of the input string. We will assume access to the function $f(b_0, b_1) = b_0 \oplus b_1$ which we implement via the unitary

$$U_f|b_0, b_1\rangle = (-1)^{f(b_0,b_1)}|b_0, b_1\rangle, \tag{61}$$

where $|\perp\rangle \in \text{span}\{|01\rangle, |10\rangle\}$. Now, assuming access to the block encoding of the interior L operator with periodic boundary conditions on $[0, L]^d$, we can implement the operator restricted to the interior points.

Now, consider the controlled operation

$$\text{SEL} = I\otimes U_L\otimes|00\rangle\langle 00| + U_f\otimes U_L\otimes|01\rangle\langle 01| + I\otimes U_B\otimes|10\rangle\langle 10| + U_{f'}\otimes U_B\otimes|11\rangle\langle 11| \tag{62}$$

where U_L is the block encoding of the operator on the interior and U_B is the block encoding of the operator given on the boundary. Then, we have that

$$\text{SEL}\cdot\text{DOM}|b\rangle|00\rangle = \sum_{i\in\Omega}\frac{b_i}{\lambda}L|i\rangle|0\rangle(|0\rangle+|\perp\rangle) + \sum_{i\in\partial\Omega}\frac{b_i}{\beta}B|i\rangle|1\rangle(|0\rangle+|\perp\rangle) + \sum_{i\notin\Omega\cup\partial\Omega} b_i|i\rangle|2\rangle, \tag{63}$$

which we write in the simplified form

$$(H\otimes H\otimes\text{DOM}^\dagger)\cdot\text{SEL}\cdot(H\otimes H\otimes\text{DOM})|b\rangle|00\rangle = \sum_{i\in\Omega} b_i L|i\rangle|0\rangle|0\rangle + \sum_{i\in\partial\Omega} b_i B|i\rangle|1\rangle|0\rangle + \sum_{i\notin\Omega\cup\partial\Omega} b_i|i\rangle|2\rangle + |\perp\rangle, \tag{64}$$

where the $|0\rangle$ is the flag to indicate the successful application of the block encoding unitary.

Writing the ancilla states in binary notation, we notice that the states indicating the boundary and exterior nodes correspond to $|01\rangle$ and to $|10\rangle$ respectively. We will assume

access to the function $f(b_0, b_1) = b_0 \oplus b_1$, which we implement via the phase kickback oracle

$$U_f |b_0, b_1\rangle = (-1)^{f(b_0,b_1)} |b_0, b_1\rangle . \tag{65}$$

Then, to obtain a block encoding of the interior portion of the operator, we can undertake a linear combination of block encodings $I \otimes U_L$ and $U_f \otimes U_L$, with $\text{PREP} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\text{SEL} = (I \otimes U_L) \otimes |0\rangle\langle 0| + (U_f \otimes U_L) \otimes |1\rangle\langle 1|$. For an arbitrary state $|x\rangle$, we may then write

$$\begin{aligned}
|x\rangle |00\rangle |0\rangle &\overset{I\otimes I\otimes H}{\to} \frac{1}{\sqrt{2}} |x\rangle |00\rangle (|0\rangle + |1\rangle) \\
&\overset{\text{DOM}}{\to} \frac{1}{\sqrt{2}} \frac{1}{N_x} |x_\Omega\rangle |00\rangle + |x_\partial\rangle |01\rangle + |x_{out}\rangle |10\rangle)(|0\rangle + |1\rangle) \\
&\overset{\text{SEL}}{\to} \frac{1}{\sqrt{2}} \frac{1}{N_x} U_L(|x_\Omega\rangle |00\rangle + |x_\partial\rangle |01\rangle + |x_{out}\rangle |10\rangle) |0\rangle + U_L(|x_\Omega\rangle |00\rangle - |x_\partial\rangle |01\rangle - |x_{out}\rangle |10\rangle) |1\rangle \\
&\overset{I\otimes I\otimes H}{\to} \frac{1}{2} U_L |x_\Omega\rangle |00\rangle |0\rangle + |\perp\rangle .
\end{aligned}$$

Therefore, these operations implement a $(2\lambda, m+3)$ block encoding of the interior operator. We note that this operator only implements the differential operator on the interior nodes and projects everything else to zero. We now need to add the boundary operators.

This can be done similarly, replacing U_f with $U_{f'}$ (which can be implemented by conjugating the control with X-gates on the least significant bit) which performs $U_f |b_0, b_1\rangle = (-1)^{f(b_0 \oplus 1, b_1)} |b_0, b_1\rangle$, where the effect of the $\oplus 1$ on b_0 bit is to place a minus sign on the nodes on both the interior and the exterior, while leaving the nodes on the boundary unaffected. If we consider a generic boundary condition and block encode the corresponding differential operator U_B, then we can implement the operator for just the boundary via the linear combination of block encodings $\frac{1}{2}\left(I \otimes U_B + U_{f'} \otimes U_B\right)$, analogously to the interior operator. Therefore, the interior and boundary operator can now be implemented via the linear combination of block encodings according to

$$U_{\mathcal{L}} \propto \frac{1}{2\lambda}(U_L \otimes I + U_L \otimes U_f) + \frac{1}{2\beta}(U_B \otimes I + U_B \otimes U_{f'}). \tag{66}$$

This formula is analogous to and generalizes those obtained in the above sections, but the implementation of the reflection operation in this case required an additional register and oracle to label each grid point. This procedure allows one to begin with the efficient block encodings of the periodic operators that we found in the above section, and to project them onto the corresponding subsets of $[0, L]^d$ so that in total the obtained operator acts as the desired differential equation and boundary conditions on the "physical" domain $\Omega \cup \partial\Omega$, while acting trivially on $\Omega_L^d \backslash \{\Omega \cup \partial\Omega\}$. Finally, we note that if the initial state $|b\rangle$ has no support on $[0, L]^d \backslash \Omega$, which would correspond to a physically relevant starting state, then the success probability of implementing this block encoding can be as large as $O(1/d)$.

4 Block encoding elliptic operator with multi-particle potential

To block encode a general elliptic operator for a many-body system, we need to be able to evaluate gradients of potential functions. In particular, we wish to construct a block encoding of the operator $\nabla V \cdot \nabla$. This term is sometimes referred to as the "convective term" and describes the transport of quantities such as mass under the influence of a force

∇V. Such operators arise in dynamical simulations described by the Fokker-Planck and Smoluchowski equations, and steady-state formulations of these, i.e., the backward and forward Kolmogorov equations, as well as in the closely related Black-Scholes equation. We focus here on the particular case where the potential function V is given as the sum over two-particle Lennard-Jones potentials

$$V_{LJ}(r_{ij}) = 4\varepsilon \left(\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right). \tag{67}$$

Here ε is the depth of the potential, σ is the particle size, and $r_{ij} = \left\| \mathbf{x}^i - \mathbf{x}^j \right\|$ is the 2-norm distance between particles i and j in $[\eta]$. We choose this potential because it incorporates the features of a wide range of potentials, in that it requires the evaluation of inverse powers of the interparticle distance r and has both attractive and repulsive regimes. The Lennard-Jones potential is a very common effective potential used in molecular dynamics simulations, and is therefore a natural setting for the study of quantum algorithms for many-body simulations. However, the constructions we present in this section could be applied to many other multi-particle potential functions in a natural way.

This section will proceed as follows. First we will briefly review a block encoding of the discrete position operator, i.e. the operator which returns the location of a grid point

$$X \left| i \right\rangle = i \left| i \right\rangle. \tag{68}$$

Then, we show how to compute the difference of position of two particles using, i.e.

$$R \left| i \right\rangle \left| j \right\rangle = (i - j) \left| i \right\rangle \left| j \right\rangle. \tag{69}$$

Then, we show how to compute the squared distance between two particles in d-dimensions $|\boldsymbol{\alpha}\rangle$, $|\boldsymbol{\beta}\rangle$, i.e.

$$R^2 \left| \boldsymbol{\alpha} \right\rangle \left| \boldsymbol{\beta} \right\rangle = \sum_{i \in [d]} (\alpha_i - \beta_i)^2 \left| \boldsymbol{\alpha} \right\rangle \left| \boldsymbol{\beta} \right\rangle \tag{70}$$

which is the squared Euclidean distance between the two particles. Once this diagonal operator is block encoded, we can then perform a polynomial transformation of these diagonal matrices using QSP to implement a polynomial approximation to multiplication operator V. Then, we show a technique to implement a block encoding of the term $\nabla V \cdot \nabla$ combining the techniques of this section with the previous section.

4.1 Block encoding position operators

Our first goal is to block encode the operator

$$X \equiv \sum_{i=0}^{N-1} x(i) \left| i \right\rangle \left\langle i \right|, \tag{71}$$

where $x(i)$ maps the point i on the grid to the corresponding point $x(i)$ in real space. In the simplest case on $[0, 1]$, we take $x(i) = i/(N-1)$. With a periodic boundary, we take $x(i) = \min(1 - i/(N-1), i/(N-1))$. Block encoding the corresponding operator is characterized by the following lemma, which can be found in [63], Appendix F Lemma 24.

Lemma 4 (LCU of diagonal position matrix [63] Lemma 24)**.** *Let the discrete "position operator"*
$X = \mathrm{diag}(0, 1, \ldots, N-1)$ *be an* $N \times N$ *matrix with* $N = 2^n$*. Then*

$$X = \frac{N-1}{2} I - \frac{1}{2} \sum_{i=0}^{n-1} 2^i Z_{(i)} \tag{72}$$

where

$$Z_{(i)} = I^{\otimes(n-i-1)} \otimes Z \otimes I^{\otimes i}, \tag{73}$$

gives an $(\lambda = N-1, m = \lceil \log(n+1) \rceil)$*-BE(X), the discrete position operator using the LCU oracles*

$$\begin{aligned} PREP_X \left|0\right\rangle &= \frac{1}{\sqrt{2}} \left|0\right\rangle + \frac{1}{\sqrt{2(2^n-1)}} \sum_{i=0}^{n-1} \sqrt{2^i} \left|i+1\right\rangle \\ SEL_X &= \sum_{i=1}^{n} Z_i \otimes \left|i\right\rangle \left\langle i\right| . \end{aligned} \tag{74}$$

With access to the block encoding of the position operator, it is straightforward to obtain a block encoding of the difference of positions operator for one dimension, using the circuit in Fig. 8. This is a subroutine we will use many times when evaluating potentials that are defined as the symmetric distance between two particles.

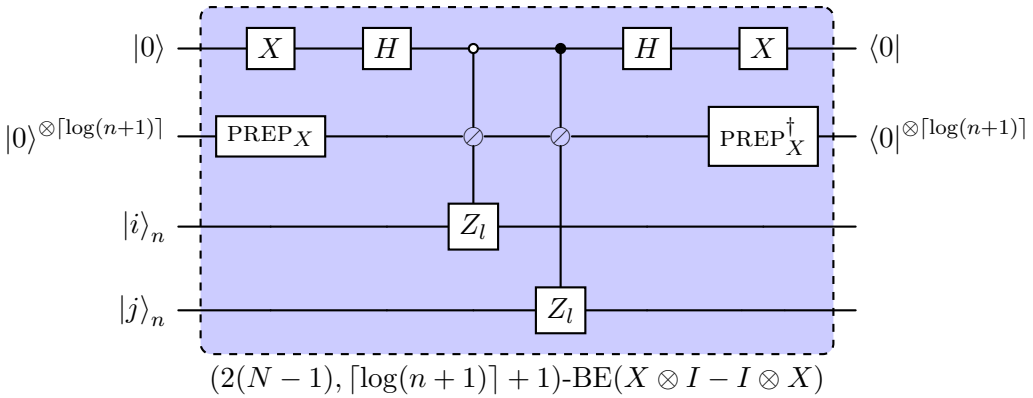


Figure 8: Circuit for the block encoding of the difference of positions operator utilizing $\lceil \log(n+1) \rceil + 1$ ancilla qubits and the LCU subroutines from the single particle positions operators. Controlled on the state of the $\lceil \log(n+1) \rceil + 1$ ancilla register, we apply a single qubit Pauli Z operator to the lth qubit of the n qubit registers encoding the positions in the grid. This block encoding applies the operation $\left|i\right\rangle \left|j\right\rangle \rightarrow \frac{(i-j)}{2(N-1)} \left|i\right\rangle \left|j\right\rangle$ whenever all of the ancilla qubits are in the $\left|0\right\rangle$ state.

We can construct block encodings of the squared distance $\left(R_k^{i,j}\right)^2$, between particles i and j in dimension k, using the block encodings of the difference of position operators. Since by definition the operator X_i^j is just X on every register except the register encoding the ith's particle's jth coordinate, and acts trivially elsewhere, it is mathematically equivalent to a block encoding of X applied only to that register. It is therefore sufficient to obtain a block encoding of X_i^j, given access to a block encoding of X. Using an additional ancilla qubit to form the LCU

$$U_{X_k^i - X_k^j} = \frac{1}{2} \left(U_{X_k^i} - U_{X_k^j} \right), \tag{75}$$

which is implemented as an $(2(N-1), \lceil \log(n+1) \rceil + 1)$ block encoding, see e.g. Fig. 8.

An immediate corollary of this is the block encoding of the squared distance between two particles in d dimensions. We denote the matrix $\mathbf{R}_{i,j}^2$ to mean the finite-dimensional operator which acts on the $2nd$ qubit registers containing particles i and j which performs the following operation

$$\mathbf{R}_{i,j}^2 \left|\boldsymbol{\alpha}^i\right\rangle \left|\boldsymbol{\alpha}^j\right\rangle = \sum_{k \in [d]} \left(\alpha_k^i - \alpha_k^j\right)^2 \left|\boldsymbol{\alpha}^i\right\rangle \left|\boldsymbol{\alpha}^j\right\rangle ,$$

for every basis state of the $2nd$ qubits. This matrix can be block encoded by a linear combination of a product of block encodings

$$\frac{1}{4}\sum_{k=0}^{d-1}\left(U_{X_k^i}-U_{X_k^j}\right)^2 \tag{76}$$

where $U_{X_k^i}$ is the block encoding of the position operator that acts non-trivially on the ith particle's kth register. This corresponds to first performing the linear combination of block encodings $U_{X_k^i}-U_{X_k^j}$, then forming the product of block encodings, then summing this up for each dimension. The cost to form this is characterized by the following theorem.

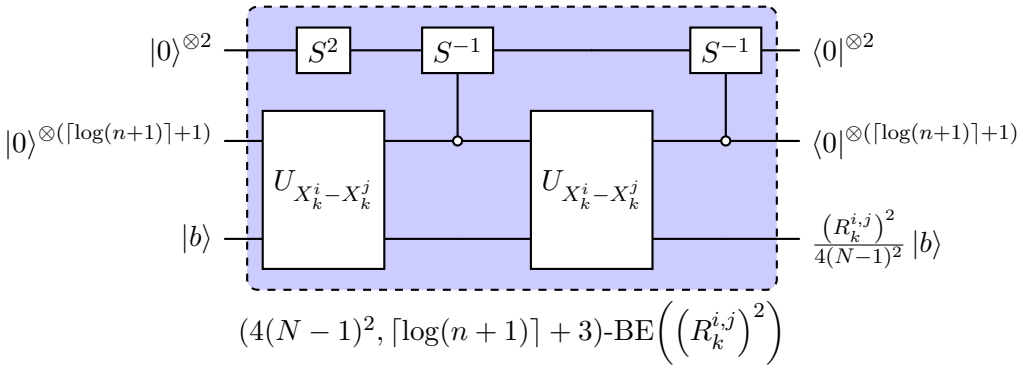


Figure 9: Quantum circuit for the product of two block encodings applied to the difference-of-positions operator. Post-selecting on the ancilla qubits being measured in the all zero state, the squared difference of position operator for particles i and j in direction $k \in [d]$ is applied to the input state. Using an LCU over $k \in [d]$ of block encodings of $(R_k^{i,j})^2$, we obtain the squared Euclidean distance in d dimensions.

Corollary 1 (Cost to block encode $\mathbf{R}_{i,j}^2$ operator)**.** *There exists a block encoding of the squared Euclidean distance between every pair of points on two grids of N^d nodes, with subnormalization $\lambda = 4d(N-1)^2$ using $m = \lceil\log(n+1)\rceil + \lceil\log(d)\rceil + 3$ ancilla qubits and this block encoding can be implemented with $O(dn)$ Toffoli or simpler gates.*

Proof. The cost to block encode this operator can be obtained by combining the costs to construct the circuits in figures 8,9, and 10. The circuit in Fig. 8 requires $O(n)$ Toffoli gates to implement the individual position operators, which applied in a controlled manner approximately doubles the number of gates, for a total of $O(4n)$ Toffoli gates. The circuit in Fig. 9 applies the circuit in Fig. 8 twice, for a total of $O(8n)$ Toffoli gates. These are then applied in a controlled manner d times to construct the circuit in Fig. 10 which would therefore require $O(16dn)$ Toffoli gates to construct the d-dimensional squared Euclidean distance operator. Since we can reuse the ancilla qubits involved in the block encodings of the individual terms in the positions operators, the circuit in Fig. 10 only requires an additional $\lceil\log(d)\rceil$ ancilla qubits over that of Fig. 9, for a total of $\lceil\log(n+1)\rceil+\lceil\log(d)\rceil+3$ ancilla qubits. □

4.2 Block encoding potential function

The operator we block encoded in the previous section is a diagonal matrix which encodes the squared Euclidean distance between two particles occupying positions in the d-dimensional grid. Combining the block encoding of corollary 1 with a polynomial approximation to the Lennard-Jones 3-6 potential, we can use QSP to operate on this diagonal matrix to construct a polynomial approximation to the Lennard-Jones 6-12 potential.

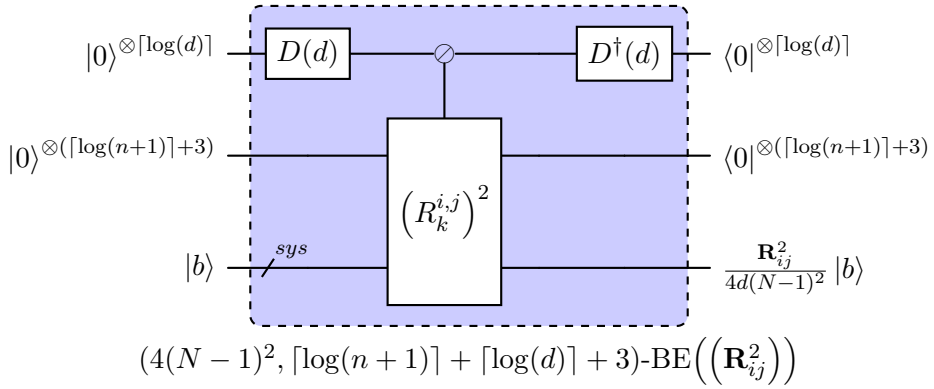


Figure 10: Quantum circuit for the d-dimensional distance of positions operator between two particles i and j. The top ancilla register prepares the uniform superposition over d states using the Diffusion operator $D(d)$. Controlled on the state of this register, we apply the one-dimensional squared difference of position operator on the system register. Then, upon measuring the all-zero state for all of the ancilla qubits, we will have applied the d-dimensional Euclidean distance operator between particles i and j to the system register. Overall, this results in an $\alpha = 4d(N-1)^2$, $m = \lceil\log(n+1)\rceil + \lceil\log(d)\rceil + 3$ block encoding of R in the product of two block encodings applied to the difference-of-positions operator. Post-selecting on the ancilla qubits being measured in the all zero state, the squared difference of position operator for particles i and j in direction $k \in [d]$ is applied to the input state.

Generally, our goal is to implement a polynomial transformation to the block encoding of the distance matrix to approximate the Lennard-Jones potential,

$$\mathbf{R}_{ij}^2 \xrightarrow{\text{poly}} \widetilde{V}(\mathbf{R}_{ij}^2), \tag{77}$$

where poly is some polynomial approximation to $\frac{1}{r^6} - \frac{1}{r^3}$ we call $\widetilde{V}$. $\widetilde{V}$ is a multiplication operator and is diagonal in the standard basis,

$$\widetilde{V}(\mathbf{R}_{ij}^2) = \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}\in[N^d]} \widetilde{V}\left(\|\boldsymbol{\alpha}-\boldsymbol{\beta}\|^2\right) |\boldsymbol{\alpha}\rangle_i |\boldsymbol{\beta}\rangle_j \langle\boldsymbol{\alpha}|_i \langle\boldsymbol{\beta}|_j . \tag{78}$$

Using the QSP subroutine on the block encodings for the squared distance, we can obtain an approximation to the potential operator as a function of the squared distance. For inverse power law potentials, the following lemma characterizes the convergence in terms of polynomial degree and a cut-off parameter $\delta > 0$ used to make the inverse power function well-defined near zero.

Lemma 5 (Convergence rate for polynomial approximations to inverse power law potentials (Ref. [29] Corollary 67))**.** *Let $\delta, \epsilon \in (0, 1/2], c > 0$ and $f(x) = \frac{\delta^c}{2} x^{-c}$, then there exist even/odd polynomials $P, P' \in \mathbb{R}[x]$ such that $||P - f||_{[\delta,1]} \leq \epsilon$, $||P||_{[-1,1]} \leq 1$, and similarly $||P' - f||_{[\delta,1]} \leq \epsilon$ and $||P'||_{[-1,1]} \leq 1$, moreover the degree of the polynomials are $O\left(\frac{max[1,c]}{\delta}\log(1/\epsilon)\right)$.*

A common choice of the cutoff is $\delta = O(1/N)$, so that the polynomial approximation rescales the Lennard-Jones potential by a factor of $O(N^{-12})$. This factor of $O(N^{-12})$ will need to be accounted for, as it does not correspond to a global scaling factor. Lemma 5 also implies that the polynomial degree to block encode the rescaled Lennard-Jones potential $\frac{\delta^{12}}{2} V_{LJ}$, is $O\left(12N\log(1/\epsilon)\right)$. In some scenarios it may be more natural to choose a cutoff parameter $\delta \sim \sigma/C$ where $C > 1$ is a constant. Since the polynomial degree is directly related to the number of queries to the block encoding of $\mathbf{R}^2$, the cost to block

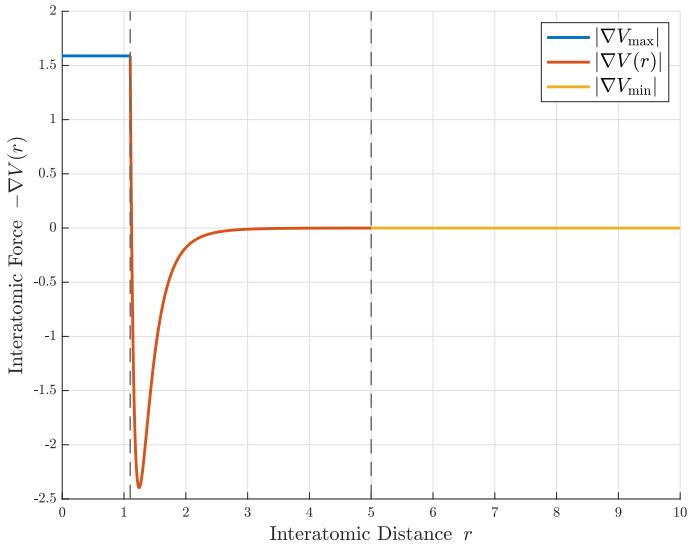


Figure 11: Depiction of the gradient of the piecewise Lennard-Jones potential in $1d$. The dashed lines demarcate the interval where we compute the polynomial approximation to the potential $V(r)$, outside of which we continuously extend as a constant.

encode a single term of the potential is $\widetilde{O}(N)$ queries to the block encoding of $\mathbf{R}^2$, the block encoding of $\mathbf{R}^2$ required $O(d)$ queries to the difference of position operator, which can each be implemented with $O(\log(N))$ Toffoli or simpler gates. Therefore, the total cost to implement a single term in the potential function $V_{LJ}(\mathbf{R}_{ij}^2)$ to ϵ accuracy is

$$\widetilde{O}(dN \log(N) \log(1/\epsilon)) \tag{79}$$

Toffoli or simpler quantum gates.

Clearly the Lennard-Jones rapidly diverges as $r \downarrow 0$. Because this divergence occurs with such a large polynomial degree, it is difficult to obtain uniform and numerically stable polynomial approximations, even with a cutoff. In practice, very high degree polynomials are required. It is worthwhile to consider other avenues beyond finding a global polynomial approximation, such as using different approximating polynomials for different subsets of the function's domain. We present an algorithm that allows one to selectively apply various transformations to subspaces of the block encoded matrix, and present an algorithm to mark the relevant subspaces using the method of inequality testing.

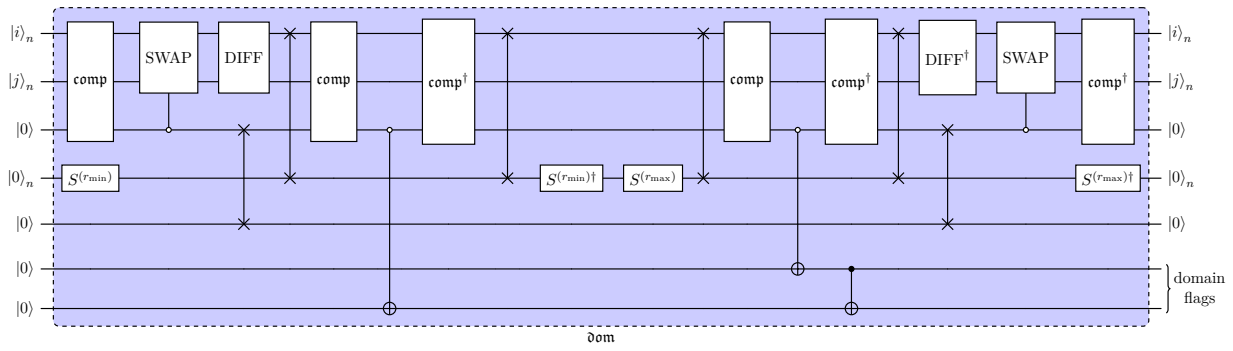


Figure 12: An explicit circuit for the domain oracle circuit $\mathfrak{dom}$ utilizing $n+2$ ancillary qubits. The two flag qubits return $|00\rangle$ if $|i-j| \in [0, r_{\text{min}}]$, $|01\rangle$ if $|i-j| \in (r_{\text{min}}, r_{\text{max}}]$, and $|10\rangle$ if $|i-j| \in (r_{\text{max}}, 1]$.

Using a comparator circuit, we build a circuit that we denote as $\mathfrak{dom}$ for assessing

which part of the piecewise domain our input is in. First, we first partition our domain as

$$\Omega = [0,1] = [0, r_{\min}) \sqcup [r_{\min}, r_{\max}] \sqcup (r_{\max}, 1]\,. \tag{80}$$

Within the interval $[r_{\min}, r_{\max}]$, we compute a polynomial approximation to $V(r)$ that converges to V at an exponential rate in the polynomial degree. Outside this interval, we continuously extend $V(r)$ to a constant function (see Figure 11). The use of comparator circuits in quantum computing was introduced in Ref. [62]. In that work a quantum circuit was constructed which implements the following inequality relations:

$$\mathfrak{comp}\,|i\rangle\,|j\rangle\,|0\rangle := \begin{cases} |i\rangle\,|j\rangle\,|0\rangle & \text{if } i < j\,, \\ |i\rangle\,|j\rangle\,|1\rangle & \text{if } i \geq j\,, \end{cases} \tag{81}$$

where $|i\rangle$ and $|j\rangle$ are n-qubit computational basis states. This is commonly refered to as "inequality testing" in the literature. The comparator circuit is obtained by modifying a quantum addition circuit to compute the most significant bit of the bit-wise subtraction $i - j$. In both one's and two's complement (i.e. unsigned and signed) binary arithmetic, bit subtraction can be rewritten in terms of bit addition using the identity [64]

$$j - i = (j' + i)'\,, \tag{82}$$

where $(\cdot)'$ is bitwise complementation. Using the quantum addition circuit given in [53], the $\mathfrak{comp}$ routine can be implemented using only n Toffoli gates.

We implement the piecewise polynomial using a domain oracle defined as

$$\mathfrak{dom}\,|i\rangle_n\,|j\rangle_n\,|0\rangle_2 := \begin{cases} |i\rangle_n\,|j\rangle_n\,|0\rangle_2 & \text{if } 0 \leq |i-j| \leq r_{\min}\,, \\ |i\rangle_n\,|j\rangle_n\,|1\rangle_2 & \text{if } r_{\min} < |i-j| \leq r_{\max}\,, \\ |i\rangle_n\,|j\rangle_n\,|2\rangle_2 & \text{if } r_{\max} < |i-j| \leq 1\,, \end{cases} \tag{83}$$

where $r_{\min}, r_{\max} \in [N]$ are the closest bit-string representations to, respectively, the minimum and maximum cut-off radii. This is done by combining quantum addition with the $\mathfrak{comp}$ oracle. First, let DIFF be defined as

$$\text{DIFF} : |i\rangle_n\,|j\rangle_n \to |i\rangle_n\,|i-j\rangle_n\,. \tag{84}$$

which can be implemented using a quantum addition circuit. The piecewise operations are then implemented using the circuit in Figure 13. The unitaries that implement the potential function gradients are the subject of the following sections.

Lemma 6 (Cost to construct $\mathfrak{dom}$ oracle)**.** *The number of Toffoli or simpler gates needed to implement the piecewise domain oracle is $O(dn)$.*

Proof. From Ref. [62], we know that $\mathfrak{comp}$, when applied to two, n qubit registers requires n Toffoli gates to implement. The controlled swap operation between two n qubit registers will also require $O(n)$ Toffoli, and DIFF as well. Therefore, all the n qubit operations can be performed with $O(n)$ Toffoli gates. Since we need to perform these comparisons in d dimensions, we therefore require $O(dn)$ Toffoli or simpler gates to implement the $\mathfrak{dom}$ oracle. □

Although this introduces some additional gate cost in the construction of the block encoding, this method allows us to greatly improve the polynomial degree in practice, and can improve the dependence of the $\delta \sim N^{-1}$ parameter in the polynomial degree. We note that a similar approach is used to block encode the Coulomb potential in Ref. [42].

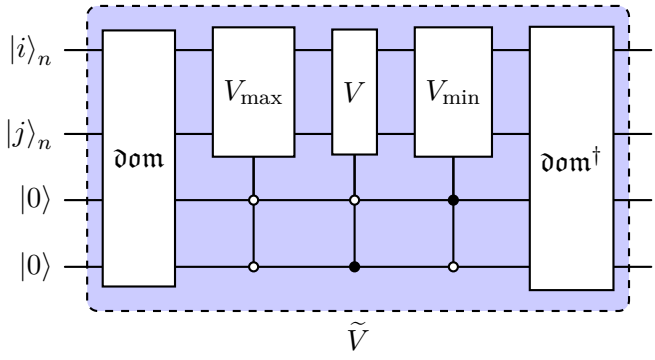


Figure 13: Circuit implementation of the domain-dependent application of the block-encodings. $\mathfrak{dom}$ represents the domain oracle, (83), and the gates $V_{\max}$, V, and $V_{\min}$ denote the block encodings of the upper potential cutoff $V_{\max}$, the two-particle potential V, and the lower potential cutoff $V_{\min}$, respectively.

4.3 Block encoding gradient operator

Once we have obtained our approximate block encoding of the potential $\widetilde{V}$, we turn to the construction of the gradient. When the partial derivatives of the potential do not have a clean closed form expression, we will need to evaluate the partial derivatives numerically. To do so we will use a central difference scheme and perform multiple evaluations of the block encoding of $\widetilde{V}$ at shifted coordinates. This approach may seem suboptimal at first, since the block encoding of V is by far the costliest operation we perform. It may be tempting to instead perform matrix multiplication of the block encodings of differential operators obtained in Sec. 3 applied to the block encoding of $\widetilde{V}$. However, this does not encode the correct operations, as multiplying the diagonal matrix $\widetilde{V}$ on the left with a finite difference matrix does not reproduce the desired behavior. We instead interpret $\nabla V = \mathbf{b}(x)$ as a vector operator, where each element is a diagonal multiplication operator $\partial_i V$ that we form by a linear combination of $\widetilde{V}$ evaluated at shifted values of the distance function weighted by the finite difference coefficients.

We construct the appropriate finite differences by directly shifting the individual position coordinate operators for each dimension and particle register. The appropriate shift can be realized via the similarity transformation

$$S^j X S^{-j} = \frac{1}{N-1}\mathrm{diag}(j, j+1, \ldots, N-1, 0, \ldots, j-2, j-1), \tag{85}$$

using the techniques from 4.1, we can straightforwardly form the partially shifted squared difference-of-position operator in d-dimensions,

$$(\mathbf{R}^{i,j}_{k,l})^2 = (S^l_k X^i_k S^{-l}_k - X^j_k)^2 + \sum_{m \neq k \in [d]} (X^i_m - X^j_m)^2. \tag{86}$$

This corresponds to shifting the position operator of particle i in direction k by l when computing the squared distance between particles i and j. We do this for every particle $i, j \in [\eta]$ and for every $k \in [d]$ and for every $1 \leq |l| \leq a$, so that we have block encodings of the squared Euclidean distance operator, evaluated at the shifted coordinates. The approximate gradient in d-dimensions, evaluated using the finite difference coefficients c_l, can therefore be expressed

$$\nabla^i V \approx \sum_{k \in [d]} \sum_{1 \leq |l| \leq a} c_l V\left((\mathbf{R}^{i,j}_{k,l})^2\right) \otimes |k\rangle\,. \tag{87}$$

We may also write,

$$V\left((\mathbf{R}_{k,l}^{i,j})^2\right) = \widetilde{V}\left(S_k^l \boldsymbol{\alpha}^i S_k^{-l}, \boldsymbol{\alpha}^j\right). \tag{88}$$

This naturally corresponds to a linear combination of block encodings, and we now detail its implementation.

We append two ancilla registers to encode the dimension and finite difference coefficients of dimension $\lceil \log(d) \rceil$ and $\lceil \log(p) \rceil$ respectively. To perform this block encoding we will use QSP over evaluations of the shifted distance matrix in conjunction with an LCU with state preparation oracles,

$$\begin{aligned} \text{PREP}_D |0\rangle &= \sum_{1 \leq |l| \leq a} \sqrt{\frac{c_l}{\beta}} |l\rangle \\ D(d) |0\rangle &= \frac{1}{\sqrt{d}} \sum_{j \in [d]} |j\rangle \end{aligned} \tag{89}$$

and where $\beta \in O(1)$ is the sum of the absolute values of the first derivative finite difference coefficients. To simplify, we introduce the notation

$$\partial_k^i \widetilde{V}(\boldsymbol{\alpha}^i, \boldsymbol{\alpha}^j) \equiv \sum_{1 \leq |l| \leq a} c_l \widetilde{V}(S_k^l \boldsymbol{\alpha}^i S_k^{-l}, \boldsymbol{\alpha}^j) \tag{90}$$

to refer to the diagonal matrix that encodes the approximate partial derivative to V_{ij} applied to particle i in direction k using a $2a$-point finite difference stencil with c_l the finite difference coefficients for the first derivative (see Eq. (10)). The single-particle d-dimensional gradient is,

$$\nabla^i \widetilde{V} = \sum_{k \in [d]} \partial_k^i \widetilde{V}(\boldsymbol{\alpha}^i, \boldsymbol{\alpha}^j) \otimes |k\rangle. \tag{91}$$

We show the quantum circuit implementing the block encoding of this matrix as Fig. 14. The following lemma characterizes the cost to implement this block encoding.

Lemma 7 (Cost to block encode ∇V for Lennard-Jones Potential)**.** *Let V be the two-particle Lennard-Jones potential, let d be the underlying spatial dimension of the system, and $p = 2a - 1$ the order of the finite difference stencil we use to evaluate the gradient. Then, there exists a quantum circuit preparing an $(O(dN^{12}), \lceil \log(n+1) \rceil + 2\lceil \log(d) \rceil + \lceil \log(p) \rceil + 4)$-BE$(\nabla \widetilde{V})$ using $O(d^2 p^2 N \log^2(N))$ Toffoli or simpler gates.*

Proof. We begin by first analyzing the gate complexity to construct Eq. (90). Constructing the shifted position operator requires two evaluations of the shift operators per term in the finite difference scheme, which requires $O(n)$ Toffoli or simpler gates each, and one evaluation of the position operator which also requires $O(n)$ Toffoli or simpler gates, so a total of $\sim 3n$ Toffoli gates are needed to block encode the one-dimensional shifted position operator. We then construct the squared Euclidean distance matrix with respect to the shifted coordinate, which will requires $\sim 3dn$ Toffoli or simpler gates. We then evaluate use QSP to approximate V, we would like our approximation $\widetilde{V}$ to be accurate to the same order as the finite difference scheme, which using estimate in Eq. 79 for the polynomial degree, will require $O(pNn)$ queries to the block encoding of the squared Euclidean distance function. Therefore, to block encode V evaluated at a single shifted coordinate requires $\sim 3dNn^2$ Toffoli or simpler gates. Since there are $2(p-1)$ shifts that need to be evaluated to approximate Eq. (90) for both particles, we will require an additional $O(6dp^2Nn^2)$ Toffoli or simpler gates. Since the gradient will require repeating this for

$2d$ times for both pairs of particles, we obtain a gate cost that is $O(12d^2p^2Nn^2)$ Toffoli or simpler gates. Substituting $n = \log(N)$ and removing the constant factor we obtain reported Toffoli complexity. The number of ancilla qubits can be immediately read-out by the circuit in Fig. 14, and the subnormalization factor directly results from the N^{12} factor needed to normalize the Lennard-Jones potential and the factor of d from the LCU over the d terms. □

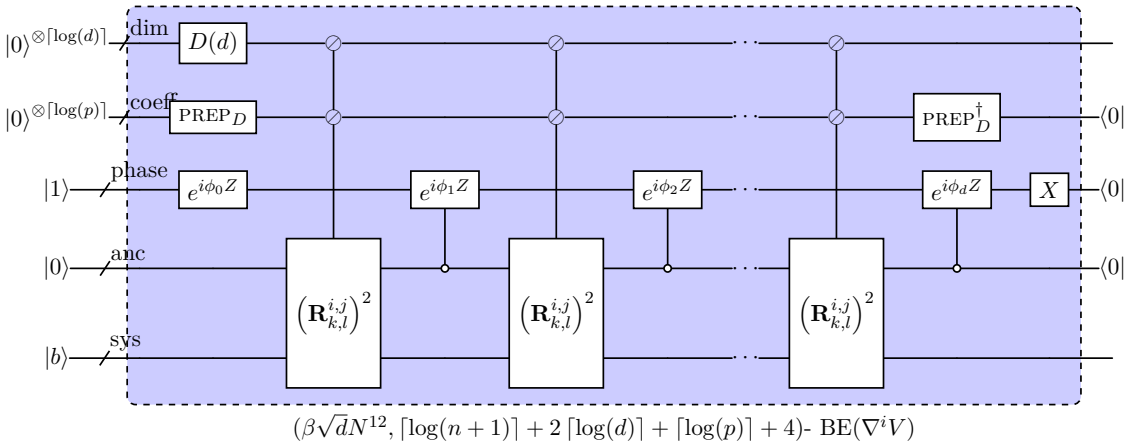


Figure 14: Quantum circuit for block encoding d-dimensional gradient for a single term of the potential function evaluated at the shifted coordinates. The phase register is used with QSP to induce the appropriate phase factors ϕ_k, corresponding to the approximating polynomial $\widetilde{V}$. Upon measuring the coeff, anc, and phase ancilla registers in the all zero state, we will have prepared a quantum state of the form $\sum_{k\in[d]}\sum_{1\leq|l|\leq a}\frac{1}{N^{12}\beta\sqrt{d}}c_l\widetilde{V}\left(S_k^l\boldsymbol{\alpha}^i S_k^{-l},\boldsymbol{\alpha}^j\right)|b\rangle|k\rangle \approx \nabla^i\widetilde{V}$, which provides a finite difference approximation to the d-dimensional gradient of particle i for the potential V_{ij}. Notice that at this stage we do not yet uncompute the dim register. We verify that this circuit produces the desired output in theorem 5 of appendix B.

4.4 Quantum circuit for convective operator

Given the above constructions for block encoding the potential gradient in the two particle case, we now turn to the block encoding of the full convective operator $(\nabla V)\cdot\nabla$,

$$(\nabla V)\cdot\nabla = \sum_{k\in[\eta]}\sum_{l\leq k}\sum_{i\in[d]}\partial_i^k V_{lk}\boldsymbol{D}_i^k, \tag{92}$$

with $V_{lk} = \widetilde{V}(\mathbf{R}_{lk}^2)$. The partial derivatives $\boldsymbol{D}_i^k$ on the right are just finite difference approximations to the first derivative and can be formed through the block encodings we construct in Sec. 3, but replacing the finite difference coefficients r_j for the Laplacian with the stencil coefficients c_j for the first derivative. We can now combine the quantum circuit in Fig. 14 and the block encoding of the d-dimensional gradient operator with the quantum circuit for a multiplication of block encodings as shown in 15. We first consider the block encoding of a term in Eq. (92), of the form

$$\nabla^i V\cdot\nabla^i = \sum_{k\in[d]}\partial_k^i V_{ij}\partial_k^i \tag{93}$$

With central difference coefficients for the first derivative operator c_j, we define the LCU oracles,

$$\text{PREP}_D|0\rangle = \sum_{j=-a,j\neq 0}^{a}\sqrt{\frac{|c_j|}{\beta}}|j\rangle, \tag{94}$$

where β is the subnormalization factor for the first derivative finite difference coefficients. We also define

$$\begin{aligned} \mathrm{SEL}_k^{i,j}(V) &= \mathrm{sign}(c_j) \sum_{1\leq|j|\leq a} U_V(S_{i,k}^{-j}\boldsymbol{\alpha}^i S_{i,k}^j, \boldsymbol{\alpha}^j) \otimes |j\rangle\langle j|\,, \\ \mathrm{SEL}_k^{i}(D) &= \sum_{1\leq|j|\leq a} \mathrm{sign}(c_j) S_{i,k}^j \otimes |j\rangle\langle j|\,. \end{aligned} \tag{95}$$

The above protocol implements a quantum circuit which block encodes a single term of

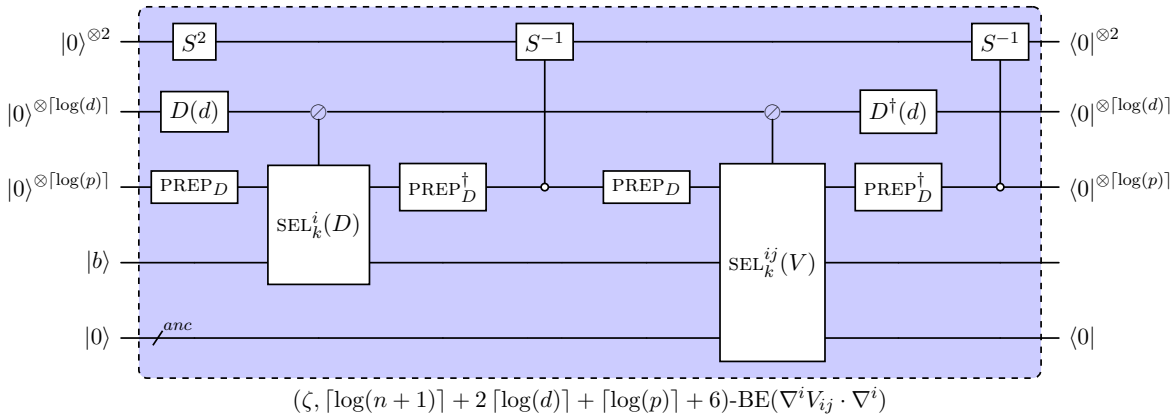


Figure 15: Quantum circuit for block encoding the finite difference approximation to $\partial_k^i V_{ij} \partial_k^i$ with subnormalization factor $\zeta = \beta^2 dN^{12} \in O(\frac{1}{N}\|\nabla V\|)$. We note that we have swapped the locations of the system and ancilla registers, denoted by the bottom two wires, as compared to Fig. 14. For an input state $|b\rangle$ in the system register, this circuit prepares $\frac{1}{\zeta}\sum_{k\in[d]} \partial_k^i V_{ij} \mathcal{D}_k^i |b\rangle$ whenever all of the ancilla qubits are measured to be in the 0 state. We remind the reader that ∂_k^i refers to the partial derivative operator for particle $i \in [\eta]$ in spatial direction $k \in [d]$, implemented as a linear combination of applications of block encodings of the potential function V_{ij}. This block encoding implements a single "diagonal" term in the convective operator, summing up over all of the terms $C^i = \nabla^i \sum_{j<i} V_{ij} \cdot \nabla^i$ would give the convective operator for particle i. Doing this for all pairs, i, j would give the full convective operator.

the convective operator.

Lemma 8 (Block encoding of a single convective term)**.** *The block encoding of a single term of the form with the approximate Lennard-Jones potential* $\widetilde{V}$

$$C^{ij} := \sum_{k\in[d]} \partial_k^i \widetilde{V}\left(\mathbf{R}_{ij}^2\right) \boldsymbol{D}_k^i \tag{96}$$

using $p-1$ *stencil points to approximate the derivative, requires* $O(dp)$ *queries to the block encoding of* $\widetilde{V}$ *and* $O(dp)$ *applications of the shift operators using* $\lceil\log(n+1)\rceil + 2\lceil\log(d)\rceil + \lceil\log(p)\rceil + 6$ *ancilla qubits and* $\widetilde{O}(Nd^2p^2\log^2(N))$ *Toffoli or simpler gates, with subnormalization factor* $O\left(dN^{12}\right)$.

Proof. The proof is an almost direct application of the quantum circuit in Fig.15 and lemma 7 and theorem 1. From lemma 7, the gate cost to form the block encoding of the two particle gradient operator of the Lennard-Jones potential in d-dimensions was $O(d^2p^2N\log^2(N))$ Toffoli. The cost to block encode the first derivative operator with $p-1$ stencil points is essentially equivalent to that of the Laplacian matrix. By theorem 1, we need $O(dp\log(N))$ Toffoli or simpler gates to perform the block encoding. We can employ two additional ancilla qubits to perform their product of block encodings, and reusing the

ancilla registers for the LCU over the dimension and finite difference coefficients. In all, this circuit therefore requires $\widetilde{O}(d^2p^2N\log^2(N) + dp\log(N))$ Toffoli or simpler gates and $\lceil\log(n+1)\rceil + 2\lceil\log(d)\rceil + \lceil\log(p)\rceil + 6$ ancilla qubits and implements a block encoding with subnormalization $\zeta = \beta^2 dN^{12} \in O(dN^{12})$ as $\beta^2 \in O(1)$. □

Now, if we consider the many particle case, with V given as a radially symmetric function of its arguments, the full potential function is given by

$$V_{tot} = \sum_{i\in[\eta]}\sum_{j<i} V(\boldsymbol{\alpha}^i, \boldsymbol{\alpha}^j), \tag{97}$$

so that the diagonal portion of the convective operator takes the form

$$C_{tot} = \sum_{i\in[\eta]}\sum_{j<i}\sum_{k\in[d]} \partial_k^i V(\boldsymbol{\alpha}^i, \boldsymbol{\alpha}^j)\partial_k^i. \tag{98}$$

With access to the block encoding of $C^{i,j}$, given as Fig. 15 we can obtain a block encoding of the many body convective operator by using controlled applications of the $C^{i,j}$ circuit over different registers to obtain a block encoding of C_{tot}.

We demonstrate how to construct this block encoding using an linear combination of block encodings, similar to above. Let

$$U_k^{ij} \in (\zeta/d, k, \epsilon)\text{- BE}(\partial_k^i V(\boldsymbol{x}^i, \boldsymbol{x}^j)\partial_k^i). \tag{99}$$

We can implement the linear combination of block encodings by selecting over the block encodings of the individual terms in the summand using the select oracle

$$\text{SEL}_C \equiv \sum_{i\in[\eta]}\sum_{j<i}\sum_{k\in[d]} U_k^{ij} \otimes |ijk\rangle\langle ijk| . \tag{100}$$

Then, to contract over the ancilla indices and block encode the corresponding scalar operator, we prepare the uniform superposition over the ancilla register with,

$$\text{PREP}_C |0\rangle|0\rangle|0\rangle = \sqrt{\frac{1}{d\binom{\eta}{2}}} \sum_{i\in[\eta]}\sum_{j<i}\sum_{k\in[d]} |ijk\rangle . \tag{101}$$

This therefore gives

$$U_C = [I_{N^{\eta d}} \otimes \text{PREP}_C^\dagger][\text{SEL}_C][I_{N^{\eta d}} \otimes \text{PREP}_C] \tag{102}$$

as a block encoding of the diagonal terms in the convective operator. We detail the costs to construct this block encoding as 2.

Theorem 2 (Block encoding many-body convective term)**.** *There exists a quantum circuit which implements a* $(O(\eta^2 dN^{12}), \lceil\log\binom{\eta}{2}\rceil + \lceil\log(n+1)\rceil + 2\lceil\log(d)\rceil + \lceil\log(p)\rceil + 6)$*-block encoding of the finite difference approximation to the convective operator of Eq.* (92) *with Lennard-Jones potential* V *using* $\widetilde{O}\left((\eta dp)^2 N\log^2(N)\right)$ *Toffoli or simpler gates.*

Proof. The proof can be obtained by direct application of 8. Simply apply the quantum circuit in Fig. 15 for all $\binom{\eta}{2}$ pairs of particles. This can be accomplished by appending an additional register of $\lceil\log\left(\binom{\eta}{2}\right)\rceil = O(2\log(\eta))$ ancilla registers, and apply the $\text{SEL}_k^i(D)$ and $\text{SEL}_k^{ij}(V)$ controlled on the ancilla register marking the particles. Therefore, the total gate cost can be estimated as $O(\eta^2)$ times the cost to block encode

C^{ij}, for a total of $\widetilde{O}(\eta^2 d^2 p^2 N \log^2(N))$ Toffoli or simpler gates. Employing a total of $\lceil \log \binom{\eta}{2} \rceil + \lceil \log(n+1) \rceil + 2\lceil \log(d) \rceil + \lceil \log(p) \rceil + 6$ ancilla qubits and implementing the block encoding with a subnormalization factor $O(\eta^2 d N^{12})$. Now, if we express N in terms of ϵ as $N \sim \epsilon^{-1/p}$, we obtain

$$\widetilde{O}\left(\eta^2 d^2 \epsilon^{-1/p} \log^2(1/\epsilon)\right) \tag{103}$$

Toffoli or simpler gates needed to implement the desired block encoding. □

5 Discussion and Outlook

The simulation of physical systems is one of the most promising use cases of future quantum computers. In this work, we have addressed the block encoding of a class of elliptic operators that arise in such simulations, when discretized on a Cartesian grid of N grid points per dimension in d spatial dimensions.We provide explicit block encodings for the standard Laplacian operator in d dimensions with Dirichlet, Neumann, and Robin boundary conditions that require only $O(d \log(N))$ Toffoli gates. However, the error convergence is restricted to low order, i.e., $\epsilon \sim N^{-2}$(Dirichlet/Periodic) or N^{-1}(Neumann/Robin), using standard approaches. We provide these block encodings as linear combinations of unitaries, where the unitaries are simply the modular inc(dec)rement operators and reflections and their controlled versions, and can be implemented with $O(\log(N))$ Toffolis. For the Dirichlet and Neumann boundary value problems, we show how to obtain high-order convergence rates through the use of the periodic extension, so that for any $p > 3$ we can bound the global truncation error by $\epsilon \sim O(N^{-p+1})$ for these boundary value problems. We also address the block encoding of operators with boundary conditions on an irregular geometry (e.g., not given as the simple Cartesian product of closed intervals), assuming that we can efficiently compute a membership oracle marking to which part of the computational domain a specific grid point belongs providing the same asymptotic complexity for generalized rectangular domains in $\mathbb{R}^d$.

We then showed how to block encode a many-body differential operator by first block encoding a multi-particle potential V given as the sum of pair-wise interactions. We focus on the evaluation of the Lennard-Jones potential, which involves calculating terms of the form $V(r) = \frac{1}{r^{12}} - \frac{1}{r^6}$, where r is the interparticle distance. This choice of potential is ubiquitous in applications in molecular dynamics, and more generally interacting kinetic theory, as described by the Fokker-Planck and Backwards Kolmogorov equations. We provide efficient quantum circuits to block encode a radially symmetric "discrete position operator" in 4.1, which computes the computes the quadratic potential between two particles in d-dimensions. We address the singular behavior of the potential near $r = 0$ through the use of a cutoff parameter δ and polynomial approximation over the whole domain. We show an alternative technique using a piecewise polynomial approximation in conjunction with an inequality test, which may improve the convergence rate of polynomial approximations in practice. Finally, we show how to block encode the convective operator given as $\nabla V \cdot \nabla$, which describes particles interacting with a force field ∇V where ∇ is the ηd-dimensional gradient operator. Overall, we find that the number of Toffoli gates needed to implement these block encodings scale as $\widetilde{O}((\eta d)^2 \delta^{-1})$. Compared to the $O(N^{\eta d})$ operations needed to realize these matrices on a classical computer, this construction provides an exponential reduction in the number of quantum operations needed to realize the same operation on a quantum device.

With the inclusion of an interaction potential, the underlying PDE is no longer separable over the individual particles. As a result, it is unlikely that classical methods can

generically and/or significantly reduce the dimensionality of the system. Thus, in the many particle setting, the potential for exponential quantum advantage is much more significant. For example, in the ubiquitous problem of solving the encoded linear system, the minimum number of classical operations generically scales with the dimensionality of linear system as $\Omega(N^{\eta d})$. On the other hand, using the constructions we provide in this work, the minimum number of non-clifford quantum operations needed to solve the same system must can be exponentially smaller, i.e. $\Omega((\eta d)^2 N \log(N))$. Of course, this discussion neglects considerations of the conditioning of the linear system, which is unlikely to scale exponentially with η. Nevertheless, this factor is present in both quantum and classical linear solvers.

The block encodings we construct in this work are provided as linear combinations of unitaries (LCUs), additional work is still needed to compile these circuits into 1- and 2-qubit gates. In particular, we do not specify the construction of circuits that prepare the LCU coefficients, nor the construction of the controlled applications of the unitaries. More work is needed to address the constant factors in the complexity resulting from the controlled applications of the 1 sparse unitaries we use in the construction. We note that generation of software to automatically compile and optimize the quantum circuit implementations of these block encodings would be a useful development. More generally, it may also be worthwhile to explore Cartesian embedding methods [58, 61] in connection with the methods presented in Sec. 3.4, to construct efficient circuits for representing differential operators on curved domains. However, we point out that if one wishes to obtain generic high-order accuracy on irregular geometries, the *hp*-FEM approach is a much more robust framework than embedding methods. Block encoding the stiffness matrix from FEM is much more involved, and is the subject of future work.

This work represents one step toward the goal of developing efficient quantum subroutines for realistic simulation tasks involving key differential operators on future quantum computers. The resulting circuits are conceptually simple and rely on the use of well-known primitives such as shift and reflection gates to block encode the Laplacian matrix, as well as on more modern methods based on inequality testing and quantum signal processing to efficiently block encode a more general elliptic operator such as $\nabla V(\mathbf{x}) \cdot \nabla$, with V a scalar multi-particle potential. This work thus provides a blueprint for the block encoding of many-body differential equations, which is likely to be a core subroutine in future fault-tolerant applications involving the numerical solution of high-dimensional partial differential equations. However, more work is needed to address the question of the efficiency and applicability quantum algorithms for solving PDEs beyond the Schrödinger equation in an end-to-end fashion. With this goal in mind, we expect these constructions will be useful for analyzing the quantum resources needed to simulate the high-dimensional, many-body, systems of interest that arise in the physical and applied sciences.

Acknowledgments

T.D.K., A.M.A., K.K.M., and K.B.W. were supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Numbers DE-SC0023273 and DE-SC0025526. T.D.K was also supported by a Siemens FutureMakers Fellowship. J.-P.L. was supported by the National Science Foundation (PHY-1818914, CCF-1729369), the NSF Quantum Leap Challenge Institute (QLCI) program (OMA-2016245, OMA-2120757), and a Simons Foundation award (No. 825053). T.D.K would also like to thank Ian Convy, Di Fang, Cory Hargus, and Torin Stetina for insightful discussions in the early portions of this work. We also thank Philipp Schleich

and Torin Stetina for helpful comments on an early version of the manuscript.

References

[1] L.C. Evans. *Partial differential equations*. Graduate studies in mathematics. American Mathematical Society, 2010. ISBN 978-0-8218-4974-3. URL https://books.google.com/books?id=Xnu0o_EJrCQC.

[2] G. Teschl. *Mathematical methods in quantum mechanics: With applications to Schrödinger operators*. Graduate studies in mathematics. American Mathematical Society, 2009. ISBN 978-0-8218-4660-5. URL https://books.google.com/books?id=bYqaAwAAQBAJ.

[3] Sarah K. Leyton and Tobias J. Osborne. A quantum algorithm to solve nonlinear differential equations, 2008. URL https://arxiv.org/abs/0812.4423.

[4] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, February 2014. ISSN 1751-8121. DOI: 10.1088/1751-8113/47/10/105301. URL http://dx.doi.org/10.1088/1751-8113/47/10/105301.

[5] Dominic W. Berry, Andrew M. Childs, Aaron Ostrander, and Guoming Wang. Quantum algorithm for linear differential equations with exponentially improved dependence on precision. *Communications in Mathematical Physics*, 356(3):1057–1081, October 2017. ISSN 1432-0916. DOI: 10.1007/s00220-017-3002-y. URL http://dx.doi.org/10.1007/s00220-017-3002-y.

[6] Andrew M. Childs and Jin-Peng Liu. Quantum spectral methods for differential equations. *Communications in Mathematical Physics*, 375(2):1427–1457, February 2020. ISSN 1432-0916. DOI: 10.1007/s00220-020-03699-z. URL http://dx.doi.org/10.1007/S00220-020-03699-Z.

[7] Hari Krovi. Improved quantum algorithms for linear and nonlinear differential equations. *Quantum*, 7:913, February 2023. ISSN 2521-327X. DOI: 10.22331/q-2023-02-02-913. URL http://dx.doi.org/10.22331/q-2023-02-02-913.

[8] Dong An, Jin-Peng Liu, Daochen Wang, and Qi Zhao. A theory of quantum differential equation solvers: limitations and fast-forwarding, 2022. URL https://arxiv.org/abs/2211.05246.

[9] Pedro C. S. Costa, Philipp Schleich, Mauro E. S. Morales, and Dominic W. Berry. Further improving quantum algorithms for nonlinear differential equations via higher-order methods and rescaling, 2023. URL https://arxiv.org/abs/2312.09518.

[10] Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, and Sabre Kais. Quantum algorithm and circuit design solving the poisson equation. *New Journal of Physics*, 15(1):013021, January 2013. ISSN 1367-2630. DOI: 10.1088/1367-2630/15/1/013021. URL http://dx.doi.org/10.1088/1367-2630/15/1/013021.

[11] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. Preconditioned quantum linear system algorithm. *Physical Review Letters*, 110(25), June 2013. ISSN 1079-7114. DOI: 10.1103/physrevlett.110.250504. URL http://dx.doi.org/10.1103/physrevlett.110.250504.

[12] Ashley Montanaro and Sam Pallister. Quantum algorithms and the finite element method. *Physical Review A*, 93(3), March 2016. ISSN 2469-9934. DOI: 10.1103/physreva.93.032324. URL http://dx.doi.org/10.1103/physreva.93.032324.

[13] Pedro C. S. Costa, Stephen Jordan, and Aaron Ostrander. Quantum algorithm for simulating the wave equation. *Physical Review A*, 99(1), January 2019. ISSN

2469-9934. DOI: 10.1103/physreva.99.012323. URL http://dx.doi.org/10.1103/physreva.99.012323.

[14] Noah Linden, Ashley Montanaro, and Changpeng Shao. Quantum vs. classical algorithms for solving the heat equation. *Communications in Mathematical Physics*, 395 (2):601–641, August 2022. ISSN 1432-0916. DOI: 10.1007/s00220-022-04442-6. URL http://dx.doi.org/10.1007/s00220-022-04442-6.

[15] Andrew M. Childs, Jin-Peng Liu, and Aaron Ostrander. High-precision quantum algorithms for partial differential equations. *Quantum*, 5:574, November 2021. ISSN 2521-327X. DOI: 10.22331/q-2021-11-10-574. URL http://dx.doi.org/10.22331/q-2021-11-10-574.

[16] Shi Jin, Nana Liu, and Yue Yu. Quantum simulation of partial differential equations: Applications and detailed analysis. *Physical Review A*, 108(3), September 2023. ISSN 2469-9934. DOI: 10.1103/physreva.108.032603. URL http://dx.doi.org/10.1103/physreva.108.032603.

[17] Shi Jin, Nana Liu, and Yue Yu. Quantum simulation of the Fokker-Planck equation via Schrodingerization, 2024. URL https://arxiv.org/abs/2404.13585.

[18] Shi Jin, Nana Liu, and Chuwen Ma. Quantum simulation of Maxwell's equations via Schrödingersation, 2023. URL https://arxiv.org/abs/2308.08408.

[19] Shi Jin, Xiantao Li, Nana Liu, and Yue Yu. Quantum simulation for partial differential equations with physical boundary or interface conditions, 2023. URL https://arxiv.org/abs/2305.02710.

[20] Shi Jin, Nana Liu, and Chuwen Ma. Schrödingerisation based computationally stable algorithms for ill-posed problems in partial differential equations, 2024. URL https://arxiv.org/abs/2403.19123.

[21] Dong An, Jin-Peng Liu, and Lin Lin. Linear combination of Hamiltonian simulation for nonunitary dynamics with optimal state preparation cost. *Physical Review Letters*, 131(15), October 2023. ISSN 1079-7114. DOI: 10.1103/physrevlett.131.150603. URL http://dx.doi.org/10.1103/physrevlett.131.150603.

[22] Dong An, Andrew M. Childs, and Lin Lin. Quantum algorithm for linear non-unitary dynamics with near-optimal dependence on all parameters, 2023. URL https://arxiv.org/abs/2312.03916.

[23] Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information and Computation*, 12(11 & 12):901–924, November 2012. ISSN 1533-7146. DOI: 10.26421/qic12.11-12-1. URL http://dx.doi.org/10.26421/QIC12.11-12-1.

[24] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, December 2006. ISSN 1432-0916. DOI: 10.1007/s00220-006-0150-x. URL http://dx.doi.org/10.1007/s00220-006-0150-x.

[25] Lin Lin. Lecture notes on quantum algorithms for scientific computation, 2022. URL https://arxiv.org/abs/2201.08309.

[26] Daan Camps, Lin Lin, Roel Van Beeumen, and Chao Yang. Explicit quantum circuits for block encodings of certain sparse matrices, 2022. URL https://arxiv.org/abs/2203.10236.

[27] Guang Hao Low and Isaac L. Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1), January 2017. ISSN 1079-7114. DOI: 10.1103/physrevlett.118.010501. URL http://dx.doi.org/10.1103/physrevlett.118.010501.

[28] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, July 2019. ISSN 2521-327X. DOI: 10.22331/q-2019-07-12-163. URL http://dx.doi.org/10.22331/q-2019-07-12-163.

[29] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC '19, page 193–204. ACM, June 2019. DOI: 10.1145/3313276.3316366. URL http://dx.doi.org/10.1145/3313276.3316366.

[30] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, November 2020. ISSN 2521-327X. DOI: 10.22331/q-2020-11-11-361. URL http://dx.doi.org/10.22331/q-2020-11-11-361.

[31] Jin-Peng Liu, Herman Øie Kolden, Hari K. Krovi, Nuno F. Loureiro, Konstantina Trivisa, and Andrew M. Childs. Efficient quantum algorithm for dissipative nonlinear differential equations. *Proceedings of the National Academy of Sciences*, 118(35), August 2021. ISSN 1091-6490. DOI: 10.1073/pnas.2026805118. URL http://dx.doi.org/10.1073/pnas.2026805118.

[32] Dong An, Di Fang, Stephen Jordan, Jin-Peng Liu, Guang Hao Low, and Jiasu Wang. Efficient quantum algorithm for nonlinear reaction-diffusion equations and energy estimation, 2022. URL https://arxiv.org/abs/2205.01141.

[33] Junyu Liu, Minzhao Liu, Jin-Peng Liu, Ziyu Ye, Yunfei Wang, Yuri Alexeev, Jens Eisert, and Liang Jiang. Towards provably efficient quantum algorithms for large-scale machine-learning models. *Nature Communications*, 15(1), January 2024. ISSN 2041-1723. DOI: 10.1038/s41467-023-43957-x. URL http://dx.doi.org/10.1038/s41467-023-43957-x.

[34] Marcelo Forets and Amaury Pouly. Explicit error bounds for carleman linearization, 2017. URL https://arxiv.org/abs/1711.02552.

[35] Xiangyu Li, Xiaolong Yin, Nathan Wiebe, Jaehun Chun, Gregory K. Schenter, Margaret S. Cheung, and Johannes Mülmenstädt. Potential quantum advantage for simulation of fluid dynamics. *Physical Review Research*, 7(1), January 2025. ISSN 2643-1564. DOI: 10.1103/physrevresearch.7.013036. URL http://dx.doi.org/10.1103/PhysRevResearch.7.013036.

[36] Di Fang, Lin Lin, and Yu Tong. Time-marching based quantum solvers for time-dependent linear differential equations. *Quantum*, 7:955, March 2023. ISSN 2521-327X. DOI: 10.22331/q-2023-03-20-955. URL http://dx.doi.org/10.22331/q-2023-03-20-955.

[37] Shi Jin, Nana Liu, and Yue Yu. Time complexity analysis of quantum difference methods for linear high dimensional and multiscale partial differential equations. *Journal of Computational Physics*, 471:111641, December 2022. ISSN 0021-9991. DOI: 10.1016/j.jcp.2022.111641. URL http://dx.doi.org/10.1016/j.jcp.2022.111641.

[38] Pedro C. S. Costa, Stephen Jordan, and Aaron Ostrander. Quantum algorithm for simulating the wave equation. *Physical Review A*, 99(1), January 2019. ISSN 2469-9934. DOI: 10.1103/physreva.99.012323. URL http://dx.doi.org/10.1103/PhysRevA.99.012323.

[39] Quynh T. Nguyen, Bobak T. Kiani, and Seth Lloyd. Block-encoding dense and full-rank kernels using hierarchical matrices: applications in quantum numerical linear algebra. *Quantum*, 6:876, December 2022. ISSN 2521-327X. DOI: 10.22331/q-2022-12-13-876. URL http://dx.doi.org/10.22331/q-2022-12-13-876.

[40] Ian D Kivlichan, Nathan Wiebe, Ryan Babbush, and Alán Aspuru-Guzik. Bounding the costs of quantum simulation of many-body physics in real space. *Journal of Physics A: Mathematical and Theoretical*, 50(30):305301, June 2017. ISSN 1751-8121. DOI: 10.1088/1751-8121/aa77b8. URL http://dx.doi.org/10.1088/1751-8121/aa77b8.

[41] Andrew M. Childs, Jiaqi Leng, Tongyang Li, Jin-Peng Liu, and Chenyi Zhang. Quantum simulation of real-space dynamics. *Quantum*, 6:860, November 2022. ISSN 2521-327X. DOI: 10.22331/q-2022-11-17-860. URL http://dx.doi.org/10.22331/q-2022-11-17-860.

[42] Yuan Su, Dominic W. Berry, Nathan Wiebe, Nicholas Rubin, and Ryan Babbush. Fault-tolerant quantum simulations of chemistry in first quantization. *PRX Quantum*, 2(4), November 2021. ISSN 2691-3399. DOI: 10.1103/prxquantum.2.040332. URL http://dx.doi.org/10.1103/PRXQuantum.2.040332.

[43] Haoya Li, Hongkang Ni, and Lexing Ying. On efficient quantum block encoding of pseudo-differential operators. *Quantum*, 7:1031, June 2023. ISSN 2521-327X. DOI: 10.22331/q-2023-06-02-1031. URL http://dx.doi.org/10.22331/q-2023-06-02-1031.

[44] John P Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, 2001.

[45] Nikita Guseynov, Xiajie Huang, and Nana Liu. Efficient explicit gate construction of block-encoding for Hamiltonians needed for simulating partial differential equations, 2024. URL https://arxiv.org/abs/2405.12855.

[46] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, Guillermo Alonso-Linaje, B. AkashNarayanan, Ali Asadi, Juan Miguel Arrazola, Utkarsh Azad, Sam Banning, Carsten Blank, Thomas R Bromley, Benjamin A. Cordier, Jack Ceroni, Alain Delgado, Olivia Di Matteo, Amintor Dusko, Tanya Garg, Diego Guala, Anthony Hayes, Ryan Hill, Aroosa Ijaz, Theodor Isacsson, David Ittah, Soran Jahangiri, Prateek Jain, Edward Jiang, Ankit Khandelwal, Korbinian Kottmann, Robert A. Lang, Christina Lee, Thomas Loke, Angus Lowe, Keri McKiernan, Johannes Jakob Meyer, J. A. Montañez-Barrera, Romain Moyard, Zeyue Niu, Lee James O'Riordan, Steven Oud, Ashish Panigrahi, Chae-Yeun Park, Daniel Polatajko, Nicolás Quesada, Chase Roberts, Nahum Sá, Isidor Schoch, Borun Shi, Shuli Shu, Sukin Sim, Arshpreet Singh, Ingrid Strandberg, Jay Soni, Antal Száva, Slimane Thabet, Rodrigo A. Vargas-Hernández, Trevor Vincent, Nicola Vitucci, Maurice Weber, David Wierichs, Roeland Wiersema, Moritz Willmann, Vincent Wong, Shaoming Zhang, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2018. URL https://arxiv.org/abs/1811.04968.

[47] Tyler Kharazi. Block encoding demos. https://github.com/Kharazitd/BlockEncodingDemos, 2025.

[48] Wikipedia contributors. Finite difference coefficient, 2023. URL https://en.wikipedia.org/w/index.php?title=Finite_difference_coefficient&oldid=1171577519. Accessed: 2025-05-21.

[49] Jianping Li. General explicit difference formulas for numerical differentiation. *Journal of Computational and Applied Mathematics*, 183(1):29–52, November 2005. ISSN 0377-0427. DOI: 10.1016/j.cam.2004.12.026. URL http://dx.doi.org/10.1016/j.cam.2004.12.026.

[50] E. Rieffel and W. Polak. *Quantum Computing: A Gentle Introduction*. Scientific and engineering computation. MIT Press, 2014. URL https://books.google.com/books?id=BLjDswEACAAJ.

[51] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5): 3457–3467, November 1995. ISSN 1094-1622. DOI: 10.1103/physreva.52.3457. URL http://dx.doi.org/10.1103/PhysRevA.52.3457.

[52] Craig Gidney. Constructing Large Increment Gates, 2015. URL https://algassert.com/circuits/2015/06/12/Constructing-Large-Increment-Gates.html. Accessed: 2023-10-03.

[53] Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, June 2018. ISSN 2521-327X. DOI: 10.22331/q-2018-06-18-74. URL http://dx.doi.org/10.22331/q-2018-06-18-74.

[54] Danial Motlagh and Nathan Wiebe. Generalized quantum signal processing, 2023. URL https://arxiv.org/abs/2308.01501.

[55] Yulong Dong, Xiang Meng, K. Birgitta Whaley, and Lin Lin. Efficient phase-factor evaluation in quantum signal processing. *Physical Review A*, 103(4), April 2021. ISSN 2469-9934. DOI: 10.1103/physreva.103.042419. URL http://dx.doi.org/10.1103/PhysRevA.103.042419.

[56] R R Rosales. Robin boundary conditions and the method of images, 2013. URL https://math.mit.edu/classes/18.306/Notes/. MIT 18.306 Course Notes. [Online; accessed 2025-05-21].

[57] Yu Tong, Dong An, Nathan Wiebe, and Lin Lin. Fast inversion, preconditioned quantum linear system solvers, fast green's-function computation, and fast evaluation of matrix functions. *Physical Review A*, 104(3), September 2021. ISSN 2469-9934. DOI: 10.1103/physreva.104.032422. URL http://dx.doi.org/10.1103/PhysRevA.104.032422.

[58] Steven J. Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics*, 227(3): 1943–1961, January 2008. ISSN 0021-9991. DOI: 10.1016/j.jcp.2007.10.009. URL http://dx.doi.org/10.1016/j.jcp.2007.10.009.

[59] P Colella, D Graves, T Ligocki, D Trebotich, and B V Straalen. Embedded boundary algorithms and software for partial differential equations. *Journal of Physics: Conference Series*, 125:012084, July 2008. ISSN 1742-6596. DOI: 10.1088/1742-6596/125/1/012084. URL http://dx.doi.org/10.1088/1742-6596/125/1/012084.

[60] Dong Xu, Jianing Liu, Yunfeng Wu, and Chunning Ji. A high-efficiency discretized immersed boundary method for moving boundaries in incompressible flows. *Scientific Reports*, 13(1), January 2023. ISSN 2045-2322. DOI: 10.1038/s41598-023-28878-5. URL http://dx.doi.org/10.1038/s41598-023-28878-5.

[61] Armando Coco and Giovanni Russo. High order finite-difference ghost-point methods for elliptic problems in domains with curved boundaries, 2024. URL https://arxiv.org/abs/2405.13986.

[62] Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. Black-box quantum state preparation without arithmetic. *Physical Review Letters*, 122 (2), January 2019. ISSN 1079-7114. DOI: 10.1103/physrevlett.122.020502. URL http://dx.doi.org/10.1103/PhysRevLett.122.020502.

[63] Priyanka Mukhopadhyay, Torin F. Stetina, and Nathan Wiebe. Quantum simulation of the first-quantized Pauli-Fierz Hamiltonian. 2023. DOI: 10.48550/ARXIV.2306.11198. URL https://arxiv.org/abs/2306.11198.

[64] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton.

A new quantum ripple-carry addition circuit, 2004. URL https://arxiv.org/abs/quant-ph/0410184.

A Circuit Implementations

A.1 Modular increment and decrement

The modular left and right shift operators implement the following transformation on $N = 2^n$ states

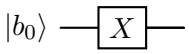
$$\begin{aligned} S^{-1}\,|l\rangle &= |(l-1) \mod N\rangle \\ S^{1}\,|l\rangle &= |(l+1) \mod N\rangle\,. \end{aligned} \tag{104}$$

In this section, we will construct the quantum circuit for the incrementer operation to provide some understanding of the underlying circuit structure.

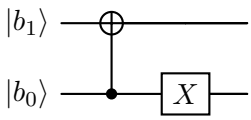
Let's first consider the simplest example, which is just a single qubit. In this case the shift-by-one operator is just the bitflip operator, or the Pauli-X gate

$$|b_0\rangle \mapsto |b_0 \oplus 1\rangle\,. \tag{105}$$

which is realized as the circuit



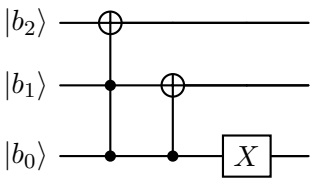
The two qubit case $|b_1, b_0\rangle \mapsto |b_1 \oplus b_0, b_0 \oplus 1\rangle$, which flips the state of qubit b_1 if the state of b_0 is 1 and flips b_0. Therefore $|00\rangle \mapsto |01\rangle \mapsto |10\rangle \mapsto |11\rangle \mapsto |00\rangle$. This circuit is realized via a controlled-X gate and an X gate:



The three qubit case is similar. This is given by the mapping

$$|b_2, b_1, b_0\rangle \mapsto |b_2 \oplus b_1 b_0, b_1 \oplus b_0, b_0 \oplus 1\rangle$$

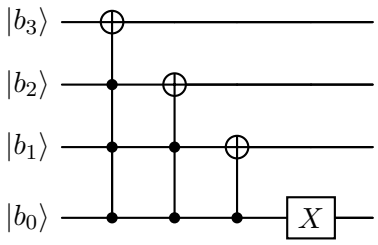
, which flips the 2nd qubit if the $b_0 = b_1 = 1$, flips the first qubit if $b_0 = 1$, and flips b_0. We can see that $|000\rangle \mapsto |001\rangle \mapsto |010\rangle \mapsto |011\rangle \mapsto |100\rangle \mapsto |101\rangle \mapsto |110\rangle \mapsto |111\rangle$ which is realized via the circuit:



The four qubit case is similar, the algebraic representation is

$$|b_3, b_2, b_1, b_0\rangle \mapsto |b_3 \oplus b_2 b_1 b_0, b_2 \oplus b_1 b_0, b_1 \oplus b_0, b_0 \oplus 1\rangle$$

and the circuit is



Carrying on in a similar fashion, for larger numbers of qubits will prepare the R operator for $n-$qubits. The implementation for the left-shift-by-one (i.e. decrement) operator is similar, and can be obtained by replacing the control on 1 with a control on 0 or equivalently conjugating the increment circuit by with Pauli X gates on all the wires. The total circuit depth is $\mathcal{O}(n)$. However, the multi-control Toffoli gates used in this scheme need to be compiled down to Toffoli gates, which can then be further compiled to one and two qubit gates. In [51], it is shown that the direct compilation of this circuit will require $O(n^2)$ Toffoli or simpler gates. However, we can use the scheme provided in [52] to implement this quantum circuit using $O(n)$ Toffoli gates, at the cost of an ancilla qubit.

Notice that when we desire to shift by any non-negative power of 2, say $0 \leq j \leq n$, the above circuit simplifies to only acting on the $n-j$ high bits of the register. Therefore, to implement the shift by constant circuit, it is sufficient to represent that constant as a signed combination (i.e. coefficients ± 1) of positive powers of 2 less than or equal to n. Therefore, combining the above increment-by-one circuit (which is just incrementing by 2^0), with a series of increments or decrements by powers of 2 to obtain the desired operation. Since in practice j is typically a small constant that does not scale with n, this procedure is quite efficient.

A.2 Reflection Operators

Lemma 9 (Cost to synthesize diagonal reflection unitary)**.** *Let $A \subset [N]$ mark states in the standard computational basis. Let R_A be reflection about the states in A, that is*

$$R_A = -2 \sum_{j \in A} |j\rangle \langle j| + I \tag{106}$$

which is just

$$R_A |i\rangle = \begin{cases} 1 & i \notin A \\ -1 & i \in A \end{cases} \tag{107}$$

Then the number of elementary gates to implement R_A is $O(|A|\mathcal{C}(C^n(Z)))$ where $\mathcal{C}(C^n(Z))$ is the cost to implement the n-qubit controlled z-gate.

Proof. Let $a \in A$ and $a \in \{0,1\}^n$ is an $n-$bit string so that we may write $a = \sum_{j=0}^{n-1} a_j 2^j \simeq \{a_0, a_1, \ldots, a_{n-1}\} \equiv b$. Let $S[a] = \{j; a_j = 0 \forall a_j \in a\}$ be the set of all unset bits of a. Then we let $X_{S[a]} = \bigotimes_{i \in [n]} X^{\neg a_i}$ be the operator that performs an $X-$gate on every qubit that corresponds to an unset bit in the bitstring a. Additionally, let $C^n(Z)$ be the n-fold controlled Z operation,

$$C^n(Z) = \sum_{i \in [N-1]} |i\rangle \langle i| - |N-1\rangle \langle N-1| . \tag{108}$$

Then $\langle d_0| \langle d_1| \cdots \langle d_{n-1}| X_{S[a]} C^n(Z) X_{S[a]} |d_0\rangle |d_1\rangle \cdots |d_{n-1}\rangle$ satisfies

$$\begin{aligned}
&\langle d_0| \langle d_1| \cdots \langle d_{n-1}| X_{S[a]} C^n(Z) X_{S[a]} |d_0\rangle |d_1\rangle \cdots |d_{n-1}\rangle \\
&= \langle d_0 \oplus \neg a_0| \langle d_1 \oplus \neg a_1| \cdots \langle d_{n-1} \oplus \neg a_{n-1}| C^n(Z) |d_0 \oplus \neg a_0\rangle |d_1 \oplus \neg a_1\rangle \cdots |d_{n-1} \oplus \neg a_{n-1}\rangle
\end{aligned}$$

But then, since $\langle i| C^n(z) |i\rangle = -1 \iff i = N-1$, implies that in order for the input state $|d\rangle = |d_0\rangle \cdots |d_{n-1}\rangle$ to be in the marked subspace, it must satisfy:

$$\begin{aligned}
d_0 \oplus \neg a_0 &= 1 \\
d_1 \oplus \neg a_1 &= 1 \\
&\vdots \\
d_{n-1} \oplus \neg a_{n-1} &= 1
\end{aligned} \tag{109}$$

which implies

$$\begin{aligned}
d_0 &= a_0 \\
d_1 &= a_1 \\
&\vdots \\
d_{n-1} &= a_{n-1}
\end{aligned} \tag{110}$$

therefore, the operations above prepared the reflection about the marked state a.

We can form a product of these reflections to get a reflection over all the elements in a subspace marked by elements of a set. Since the number of elements in the set is $|A|$, we expect this process needs to be performed for $O(|A|)$ times. Resultantly, the cost to implement this operation is $O(|A|\mathcal{C}(C^n(z)))$ where $\mathcal{C}(C^n(z)) = O(n)$ is the cost to implement the controlled-Z on n-qubits in terms of Toffoli and T gates. □

B Block Encoding Costs

Theorem 3 (Block encoding $d-$dimensional Laplacian with Dirichlet Boundary)**.** *Let L be the $d-$dimensional finite difference Laplacian with a $p = 2a+1$ point scheme on N grid points. And let $A = A_0 \cup A_1 \cdots \cup \cdots A_{d-1}$ be the set of all boundary points over the d-dimensional domain with each $A_i \subset [N]$. We additionally assume that each A_i is a connected set in the sense that $\forall x, y \in A_i$, $\exists i \leq |A_i|$ such that $x \pm i \equiv y \mod N$ and we also assume that each A_i can be specified without dependence on the coordinates in A_j for every $j \neq i$.*

We define the shifted set

$$A_{i+c} = \{j + c \mod N : j \in A_i\}. \tag{111}$$

Then the $1d$ Laplacian L_d with boundary nodes A_d can be specified via the LCU

$$\frac{1}{2} \sum_{j=-a}^{a} r_j \left(S^j + S^j R_{A_d - j} \right) + \frac{1}{2}(I + R_{A_d}). \tag{112}$$

And the d-dimensional Laplacian can be encoded using the direct sum

$$L = \bigoplus_{i=0}^{d-1} L_i, \tag{113}$$

where L_i is the Laplacian formed by separable connected Dirichlet boundary conditions in each direction i.

Proof. Our goal is to show that Eq. (112) results in a matrix acting as the identity on the boundary points and the Laplacian on the interior points. Then, we enforce the assumed condition that the boundary nodes can be specified in each dimensions independently to combine these one-dimensional operators into the appropriate $d-$dimensional operator.

The finite-difference, periodic L operator on N nodes with is expressible as the sum

$$L = \sum_{i=-a}^{a} r_i S^i \tag{114}$$

where the coefficients r_i are given by Eq. (9) and S^i are the shift operators. Now, consider A_d a set of boundary points on the dth spatial coordinate. Then,

$$\begin{aligned}
&\sum_{j=-a}^{a} (r_j S^j + r_j S^j R_{A_d - j}) \\
&= \sum_{j=-a}^{a} (r_j S^j + r_j S^j (-2 \sum_{k \in A_d} |k-j\rangle \langle k-j| - I)) \\
&= \sum_{j=-a}^{a} r_j \sum_{i \in [N]} (|i+j\rangle \langle i| + |i+j\rangle \langle i| (-2 \sum_{k \in A_d} |k-j\rangle \langle k-j| + I))
\end{aligned}$$

$$\begin{aligned}
&\sum_{i \in [N]} (-2 \sum_{k \in A_d} |i+j\rangle \langle i| \, |k-j\rangle \langle k-j| + |i+j\rangle \langle i|) \\
&= -2 \sum_{i \in [N]} \sum_{k \in A_d} |i+j\rangle \, \delta_{i,k-j} \langle k-j| + \sum_{i \in [N]} |i+j\rangle \langle i| \\
&= -2 \sum_{k \in A_d} |k\rangle \langle k-j| + \sum_{i \in [N]} |i+j\rangle \langle i|
\end{aligned}$$

$$\begin{aligned}
&\sum_{i \in [N]} |i+j\rangle \langle i| + |i+j\rangle \langle i| (-2 \sum_{k \in A_d} |k\rangle \langle k-j| + I)) \\
&= \sum_{i \in [N]} |i+j\rangle \langle i| - 2 \sum_{k \in A_d} |k\rangle \langle k-j| + \sum_{i \in [N]} |i+j\rangle \langle i| \\
&= 2 \sum_{i \in [N]} |i+j\rangle \langle i| - 2 \sum_{k \in A_d} |k\rangle \langle k-j| \\
&\propto \sum_{i \in [N]} |i+j\rangle \langle i| - \sum_{k \in A_d} |k\rangle \langle k-j| \\
&= \begin{cases} 0 & \forall \; k \in A_d : \;\; |i+j\rangle \langle i| = |k\rangle \langle k-j| \\ 1 & \text{else} \end{cases}
\end{aligned}$$

Then,

$$\begin{aligned}
&\langle m| \sum_{i\in[N]} |i+j\rangle \langle i| - \sum_{k\in A_d} |k\rangle \langle k-j| \, |n\rangle \\
&= \sum_{i\in[N]} \delta_{i+j,m}\delta_{i,n} - \sum_{k\in A_d} \delta_{m,k}\delta_{n,k-j} \\
&= \delta_{n+j,m} - \delta_{m,A_d}\delta_{n,m-j} \\
&= \delta_{n+j,m} - \delta_{m,A_d}\delta_{n+j,m} \\
&= \begin{cases} 0 & m \in A_d \text{ and } n+j=m \\ 1 & m \notin A_d \text{ and } n+j=m \end{cases}
\end{aligned}$$

which is the desired behavior.

So, we may now write:

$$\begin{aligned}
&\sum_{j=-a}^{a} r_j \left(\sum_{n,m\in[N]} \delta_{n+j,m} - \sum_{n\in[N]} \sum_{m\in A_d} \delta_{n+j,m} \right) \\
&= \sum_{j=-a}^{a} r_j \left(S^j - \sum_{n\in[N]} \sum_{m\in A_d} \delta_{n+j,m} \right) \\
&= \sum_{j=-a}^{a} r_j \left(S^j - \sum_{n\in[N]} \sum_{m\in A_d} \delta_{n+j,m} \right)
\end{aligned}$$

which more plainly now, we can see that this serves to zero out the entries on the rows supported in A_d. Now we turn to the particular case of the diagonal:

$$r_0[I - \sum_{m\in A_d} \delta_{m,m}]_{jj} = \begin{cases} r_0 & m \notin A_d \\ 0 & m \in A_d. \end{cases}$$

Now, if we add another projector onto A_d we find,

$$[r_0(I - \sum_{m\in A_d} \delta_{m,m}) + \Pi_{A_d}]_{ii} = \begin{cases} r_0 & i \notin A_d \\ 1 & i \in A_d \end{cases}. \tag{115}$$

Then, we can decompose the projector $\Pi_{A_d} = I - \Pi_{A_d^c}$ as

$$\Pi_{A_d} = \frac{1}{2}\left(I - R_{A_d^c}\right) = \frac{1}{2}\left(I + R_{A_d}\right) \tag{116}$$

to obtain the final LCU

$$L_d = \frac{1}{2} \sum_{j=-a}^{a} r_j \left(S^j + S^j R_{A_d - j}\right) + \frac{1}{2}(I + R_{A_d}). \tag{117}$$

Resultantly, the final matrix has the form

$$L_d = \begin{pmatrix} L^{(l)} & & \\ & I_{|A_d|} & \\ & & L^{(r)} \end{pmatrix} \tag{118}$$

as desired. □

Theorem 4 (Cost to block encode the LCU given in Eq. (44))**.** *The number of elementary gates needed to implement the LCU given by Eq. (44) is* $O(p|A|\mathcal{C}(C^n(Z)))$ *Toffoli. Furthermore, the above implements an* $(O(1), \log(2p), 0)$ *block encoding of the desired matrix.*

Proof. The linear combination provided via Eq. (117) requires $2p$ shift matrices and p reflection matrices. Using the construction from [52] gates, the shift operators can be encoded using $O(n)$ Toffoli-or-smaller gates with an additional ancilla. We assume these to be the bottleneck for implementing these operations, so we will count the complexity in terms of the total number of Toffoli gates needed. Therefore, the cost to implement these shift gates is $O(np)$ Toffoli or simpler gates.

The reflection matrices are diagonal matrices with ± 1 on the diagonal, with -1's on rows who's indices correspond to states of the subspace being reflected over,

$$R_A |j\rangle = \begin{cases} |j\rangle & j \notin A \\ -|j\rangle & j \in A \end{cases} \tag{119}$$

as given in Lemma 9, the cost to implement the quantum circuit for the reflection is $O(|A|\mathcal{C}(C^n(Z))p)$. So that the total number of Toffoli or simpler operations needed is $O(p|A|\mathcal{C}(C^n(Z))) = O(np)$.

The subnormalization factor is the $1-$norm of the coefficients in the LCU.

$$\lambda = \sum_{j=-a}^{a} |r_j| + 1 \leq \frac{2\pi^2}{3} + 1 = O(1) \tag{120}$$

where the first inequality can be found in Lemma 6 of Ref [40]. Then, in d-dimensions, using $O(\log(p) + \log(d))$ ancilla qubits to form the linear combination

$$\frac{1}{d} \sum_{i \in [d]} U_{L_j} \tag{121}$$

where U_{L_j} is the block encoding of the direction j Laplacian.

So in total, this requires $O(dp+\log(d))$ ancilla, $\alpha = O(\frac{1}{d})$, a number of Toffoli or smaller gates that scales as $O(dnp\mathcal{C}(C^n(Z))|A|)$ and in time $O(np\mathcal{C}(C^n(Z)))$. And similarly for η particles another multiplicative factor of η; $\alpha = O(\frac{1}{\eta d}) < \frac{\pi^2}{\eta d}$, $T = O(\eta dnp\mathcal{C}(C^n(Z))|A|)$, but add no cost to overall simulation time as they can be performed in parallel. □

Theorem 5 (Proof of circuit in Fig. 14)**.** *There exists a set of phases* $\phi_i \in \mathbb{R}$ *so that the quantum circuit in Fig. 14 encodes a superposition of evaluations of the approximate Lennard-Jones potential.*

Proof. By lemma 3, for any polynomial P and Q satisfying the conditions of the lemma, we are promised the existence of phases ϕ_i that encode the polynomial the QSP ansatz. The existence and rate of convergence of the approximating polynomial satisfying these assumptions is given by lemma 5. Now, we verify that the quantum circuit in Fig. 14 produces the desired output. The proof is obtained from direct computation of the operations. After the first layer of gates, the input quantum state is transformed to,

$$|0\rangle_{dim} |0\rangle_{coeff} |1\rangle_{phase} |0\rangle_{anc} |b\rangle_{sys} \xrightarrow{D(d)\otimes \text{PREP}_D \otimes e^{i\phi_0 Z}} \frac{e^{i\phi_0}}{\sqrt{\beta d}} \sum_{k\in[d]} \sum_{1\leq|j|\leq a} \sqrt{c_j} |k\rangle_{dim} |j\rangle_{coeff} |1\rangle_{phase} |0\rangle_{anc} |b\rangle_{sys} .$$

Then, applying the controlled block-encoding $\left(R_{kl}^{ij}\right)^2$, we obtain

$$\frac{e^{i\phi_0}}{\sqrt{\beta d}} \sum_{k\in[d]} \sum_{1\leq|j|\leq a} \sqrt{c_j} \left(|k\rangle_{dim} |j\rangle_{coeff} |1\rangle_{phase} |0\rangle_{anc} \left(R_{kl}^{ij}\right)^2 |b\rangle_{sys} + |\perp\rangle \right).$$

Let us now consider the action of the rest of the circuit on single basis state,

$$e^{i\phi_0} \left(|k\rangle_{dim} |j\rangle_{coeff} |1\rangle_{phase} |0\rangle_{anc} \left(R_{kl}^{ij}\right)^2 |b\rangle_{sys} + |\perp\rangle \right).$$

Due to the controlled application of the block encoding, the *dim* and *coeff* registers are unchanged and can be ignored for this part. Therefore, this portion of the circuit corresponds to applying QSP to the block encoding of the shifted difference of positions operator

$$e^{i\phi_0} \left(|1\rangle_{phase} |0\rangle_{anc} \left(R_{kl}^{ij}\right)^2 |b\rangle_{sys} + |\perp\rangle \right) \xrightarrow{qsp} e^{i\phi_0} \left(|1\rangle_{phase} |0\rangle_{anc} \widetilde{V} \left(\left(R_{kl}^{ij}\right)^2 \right) |b\rangle_{sys} + |\perp\rangle \right).$$

Now, with the *coeff* and *dim* registers, we have

$$e^{i\phi_0} \left(\frac{1}{\sqrt{\beta d}} \sum_{k\in[d]} \sum_{1\leq|j|\leq a} |k\rangle_{dim} |j\rangle_{coeff} |1\rangle_{phase} |0\rangle_{anc} \sqrt{c_j} \widetilde{V} \left(\left(R_{kl}^{ij}\right)^2 \right) |b\rangle_{sys} + |\perp\rangle \right),$$

and performing $\textsc{prep}_D^\dagger$ on the *coeff* register and X on the *phase* register, we have

$$e^{i\phi_0} \left(\frac{1}{\beta\sqrt{d}} \sum_{k\in[d]} |k\rangle_{dim} |0\rangle_{coeff} |0\rangle_{phase} |0\rangle_{anc} \sum_{1\leq|j|\leq a} c_j \widetilde{V} \left(\left(R_{kl}^{ij}\right)^2 \right) |b\rangle_{sys} + |\perp\rangle \right),$$

which we can simplify as

$$e^{i\phi_0} \left(\frac{1}{\beta\sqrt{d}} \sum_{k\in[d]} |k\rangle_{dim} |0\rangle_{coeff} |0\rangle_{phase} |0\rangle_{anc} \partial_k^i \widetilde{V} \left(\left(R_{kl}^{ij}\right)^2 \right) |b\rangle_{sys} + |\perp\rangle \right),$$

, so that upon measuring the *dim coeff phase* and *anc* registers in the zero state we have

$$e^{i\phi_0} \left(\frac{1}{\beta\sqrt{d}} \sum_{k\in[d]} |k\rangle_{dim} \partial_k^i \widetilde{V} \left(\left(R_{kl}^{ij}\right)^2 \right) |b\rangle_{sys} + |\perp\rangle \right), \tag{122}$$

which produces the desired output, a finite difference approximation to the gradient. □