Quantum algorithm and circuit design solving the Poisson equation



2013 New J. Phys. 15 013021

(http://iopscience.iop.org/1367-2630/15/1/013021)

New Journal of Physics
The open access journal for physics

# Quantum algorithm and circuit design solving the Poisson equation

**Yudong Cao[1], Anargyros Papageorgiou[2], Iasonas Petras[2], Joseph Traub[2] and Sabre Kais[3,4]**

[1] Department of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA
[2] Department of Computer Science, Columbia University, New York, NY 10027, USA
[3] Department of Chemistry, Physics, and Computer Science, Birck Nanotechnology Center, Purdue University, West Lafayette, IN 47907, USA
E-mail: kais@purdue.edu

*New Journal of Physics* **15** (2013) 013021 (29pp)
Received 10 July 2012
Published 11 January 2013
Online at http://www.njp.org/
doi:10.1088/1367-2630/15/1/013021

**Abstract.** The Poisson equation occurs in many areas of science and engineering. Here we focus on its numerical solution for an equation in $d$ dimensions. In particular we present a quantum algorithm and a scalable quantum circuit design which approximates the solution of the Poisson equation on a grid with error $\varepsilon$. We assume we are given a superposition of function evaluations of the right-hand side of the Poisson equation. The algorithm produces a quantum state encoding the solution. The number of quantum operations and the number of qubits used by the circuit is almost linear in $d$ and polylog in $\varepsilon^{-1}$. We present quantum circuit modules together with performance guarantees which can also be used for other problems.

[4] Author to whom any correspondence should be addressed.

## Contents



## 1. Introduction

Quantum computers take advantage of quantum mechanics to solve certain computational problems faster than classical computers. Indeed in some cases the quantum algorithm is exponentially faster than the best classical algorithm known [1–12].

In this paper we present a quantum algorithm and circuit solving the Poisson equation. The Poisson equation plays a fundamental role in numerous areas of science and engineering, such as computational fluid dynamics [13, 14], quantum mechanical continuum solvation [15], electrostatics [16], the theory of Markov chains [17–19] and is important for density functional theory and electronic structure calculations [20].

Any classical numerical algorithm solving the Poisson equation with error $\varepsilon$ has cost bounded from below by a function that grows as $\varepsilon^{-\alpha d}$, where $d$ denotes the dimension or the number of variables, and $\alpha > 0$ is a smoothness constant [21, 22]. Therefore the cost grows exponentially in $d$ and the problem suffers from the curse of dimensionality.

We show that the Poisson equation can be solved with error $\varepsilon$ using a quantum algorithm with a number of quantum operations which is almost linear in $d$ and polylog in $\varepsilon^{-1}$. A number of repetitions proportional to $\varepsilon^{-4\alpha}$ guarantees that this algorithm succeeds with probability arbitrarily close to 1. Hence the quantum algorithm breaks the curse of dimensionality and, with respect to the dimension of the problem $d$, enjoys exponential speedup relative to classical algorithms.

On the other hand, we point out that the output of the algorithm is a quantum state that encodes the solution on a regular grid rather than a bit string that represents the solution. It can

be useful if one is interested in computing a function of the solution rather than the solution itself. In general, the quantum circuit implementing the algorithm can be used as a module in other quantum algorithms that need the solution of the Poisson equation to achieve their main task.

In terms of the input of the algorithm, we assume that a quantum state encoding a superposition of function evaluations of the right-hand side of the Poisson equation is available to us, and we do not account for the cost of preparing this superposition. In general, preparing arbitrary quantum states is a very hard problem. Nevertheless, in certain cases one can efficiently prepare superpositions of function evaluations using the techniques in [23, 24]. We do not deal with the implementation of such superpositions in this paper.

There are many ways to solve the Poisson equation. We choose to discretize it on a regular grid in Cartesian coordinates and then solve the resulting system of linear equations. For this we use the quantum algorithm of [25] for solving linear equation systems. The solution of differential and partial differential equations (PDEs) is a natural candidate for applying that algorithm, as already stated in [25]. It has been applied to the solution of differential equations in [26, 27]. In the case of the Poisson equation that we consider in this paper there is no need, however, to assume that the matrix is given by an oracle. Indeed, a significant part of our work deals with the Hamiltonian simulation of the matrix of the Poisson equation. Moreover, it is an open problem to determine when it is possible to simulate a Hamiltonian with cost polynomial in the logarithm of the matrix size and the logarithm of $\varepsilon^{-1}$ [28]. Our results show that in the case of the Hamiltonian for the Poisson equation the answer is positive.

Our analysis of the implementation includes all the numerical details and will be helpful to researchers working on other problems. All calculations are carried out in fixed precision arithmetic and we provide accuracy and cost guarantees. We account for the qubits, including ancilla qubits, needed for the different operations. We provide quantum circuit modules for the approximation of trigonometric functions, which are needed in the Hamiltonian simulation of the matrix of the Poisson equation. We show how to obtain a quantum circuit computing the reciprocal of the eigenvalues using Newton iteration and modular addition and multiplication. We show how to implement quantum mechanically the inverse trigonometric function needed for controlled rotations. As we indicated, our results are not limited to the solution of the Poisson equation but can be used in other quantum algorithms. Our simulation module can be combined with splitting methods to simulate the Hamiltonian $-\Delta + V$, where $\Delta$ is the Laplacian and $V$ is a potential function. The trigonometric approximations can be used by algorithms dealing with quantum walks. The reciprocal of a real number and a controlled rotation by an angle obtained by an inverse trigonometric approximation are needed for implementing the linear system's algorithm [25] regardless of the matrix involved.

## 2. Overview

We consider the $d$-dimensional Poisson equation with Dirichlet boundary conditions.

**Definition 1.**

$$\begin{aligned} -\Delta u(x) &= f(x), \quad x \in I_d := (0,1)^d, \\ u(x) &= 0, \quad x \in \partial I_d, \end{aligned} \tag{1}$$

*where* $f : I_d \to \mathbb{R}$ *is a sufficiently smooth function; e.g. see [21, 29, 30] for details.*

For simplicity we study this equation over the unit cube but a similar analysis applies to more general domains in $\mathbb{R}^d$. Often one solves this equation by discretizing it and solving the resulting linear system. A finite difference discretization of the Poisson equation on a grid with mesh size $h$, using a $(2d+1)$ stencil for the Laplacian, yields the linear system

$$-\Delta_h \vec{v} = \vec{f}_h, \tag{2}$$

where $f_h$ is the vector obtained by sampling the function $f$ on the interior grid points [30–32]. The resulting matrix is symmetric positive definite.

To solve the Poisson equation with error $O(\varepsilon)$ both the discretization error and the error on the solution of the system should be $O(\varepsilon)$. This implies that $\Delta_h$ is a matrix of size proportional to $\varepsilon^{-\alpha d} \times \varepsilon^{-\alpha d}$, where $\alpha > 0$ is a constant that depends on the smoothness of the solution which, in turn, depends on the smoothness of $f$ [21, 30, 33]. For example, when $f$ has uniformly bounded partial derivatives up to order four then $\alpha = 1/2$.

There are different ways for solving this system using classical algorithms. Demmel [31, table 6.1] lists a number of possibilities. The conjugate gradient algorithm [34] is an example. Its cost for solving this system with error $\varepsilon$ is proportional to

$$\varepsilon^{-\alpha d}\sqrt{\kappa}\log\varepsilon^{-1},$$

where $\kappa$ denotes the condition number of $\Delta_h$. We know $\kappa = \varepsilon^{-2\alpha}$, independently of $d$. The resulting cost is proportional to $\varepsilon^{-\alpha d-\alpha}\log\varepsilon^{-1}$. For details about the solution of large linear systems see [35]. Observe that the factor $\varepsilon^{-\alpha d}$ in the cost is the matrix size and its contribution cannot be overcome. Any direct or iterative classical algorithm solving this system has a cost of at least $\varepsilon^{-\alpha d}$, since the algorithm must determine all unknowns. So any algorithm solving the system has a cost exponential in $d$. In fact a much stronger result holds, namely, the cost of any classical algorithm solving the Poisson equation in the worst case must be exponential in $d$ [21].

We present a scalable quantum circuit for the solution of (2) and thereby for the solution of the Poisson equation with error $O(\varepsilon)$ that uses a number of qubits proportional to $\max\{d, \log_2\varepsilon^{-1}\}(\log_2 d + \log_2\varepsilon^{-1})^2$ and a number of quantum operations proportional to $\max\{d, \log_2\varepsilon^{-1}\}(\log_2 d + \log_2\varepsilon^{-1})^3$. It can be shown that $\log_2 d = O(\log_2\varepsilon^{-1})$ and the above expressions are simplified to $\max\{d, \log_2\varepsilon^{-1}\}(\log_2\varepsilon^{-1})^2$ qubits and $\max\{d, \log_2\varepsilon^{-1}\}(\log_2\varepsilon^{-1})^3$ quantum operations. A measurement outcome at the final state determines whether the algorithm has succeeded or not. A number of repetitions proportional to the square of the condition number yields a success probability arbitrarily close to one.

In section 3 we deal with the discretization of the Poisson equation showing the resulting matrix. We also describe how the matrix in the multidimensional case can be expressed in terms of the one-dimensional matrix using Kronecker products. This, as we will see, is important in the simulation of the Poisson matrix. In section 4 we show the quantum circuit solving the Poisson equation. We perform the error analysis and show the quantum circuit modules computing the reciprocal of the eigenvalues and from those the controlled rotation needed at the end of the linear system's algorithm [25]. In section 5 we deal with the Hamiltonian simulation of the matrix of the Poisson equation. The exponential of the multidimensional Hamiltonian is the $d$-fold tensor product of the exponential of the one-dimensional Hamiltonian. It is possible to diagonalize the one-dimensional Hamiltonian using the quantum Fourier transform. Thus it suffices to approximate the eigenvalues in a way leading to the desired accuracy in the result. We show the quantum circuit modules performing the eigenvalue approximation and derive the overall simulation cost. In section 6 we derive the total cost for solving the Poisson equation. Section 7 is the conclusion. In appendix A we list a number of elementary quantum gates

and in appendix B we present a series of results concerning the accuracy and the cost of the approximations we use throughout the paper.

## 3. Discretization

### 3.1. One dimension

We start with the one-dimensional case to introduce the matrix $L_h$ that we will use later in expressing the $d$-dimensional discretization of the Laplacian, using Kronecker products. We have

$$\begin{aligned} -\frac{\mathrm{d}^2u(x)}{\mathrm{d}x^2} &= f(x), \quad x \in (0,1), \\ u(0) &= u(1) = 0, \end{aligned} \tag{3}$$

where $f$ is a given smooth function and $u$ is the solution we want to compute. We discretize the problem with mesh size $h = 1/M$ and we compute an approximate solution $v$ at $M+1$ grid points $x_i = ih$, $i = 0, \ldots, M$. Let $u_i = u(x_i)$ and $f_i = f(x_i)$, $i = 0, \ldots, M$.

Using finite differences at the grid points to approximate the second derivative (3) becomes

$$-\left.\frac{\mathrm{d}^2u(x)}{\mathrm{d}x^2}\right|_{x=x_i} = \frac{2u_i - u_{i-1} - u_{i+1}}{h^2} - \xi_i, \tag{4}$$

where $\xi_i$ is the truncation error and can be shown to be $O(h^2\|\frac{d^4u}{\mathrm{d}x^4}\|_\infty)$ if $f$ has the fourth derivative uniformly bounded by a constant [31].

Ignoring the truncation error, we solve

$$h^{-2}(-v_{i-1} + 2v_i - v_{i+1}) = f_i, \quad 0 < i < M. \tag{5}$$

With boundary conditions $v_0 = 0$ and $v_M = 0$, we have $M-1$ equations and $M-1$ unknowns $v_1, \ldots, v_{M-1}$:

$$h^{-2}\, L_h \begin{pmatrix} v_1 \\ \vdots \\ \vdots \\ v_{M-1} \end{pmatrix} := h^{-2} \begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ \vdots \\ v_{M-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ \vdots \\ f_{M-1} \end{pmatrix}, \tag{6}$$

where $L_h$ is the tridiagonal $(M-1) \times (M-1)$ matrix above; for the properties of this matrix, including its eigenvalues and eigenvectors see [31, section 6.3].

### 3.2. Two dimensions

In two dimensions the Poisson equation is

$$\begin{aligned} -\frac{\partial^2u(x,y)}{\partial x^2} - \frac{\partial^2u(x,y)}{\partial y^2} &= f(x,y), \quad (x,y) \in (0,1)^2, \\ u(x,0) = u(0,y) = u(x,1) = u(1,y) &= 0, \quad x, y \in [0,1]. \end{aligned} \tag{7}$$

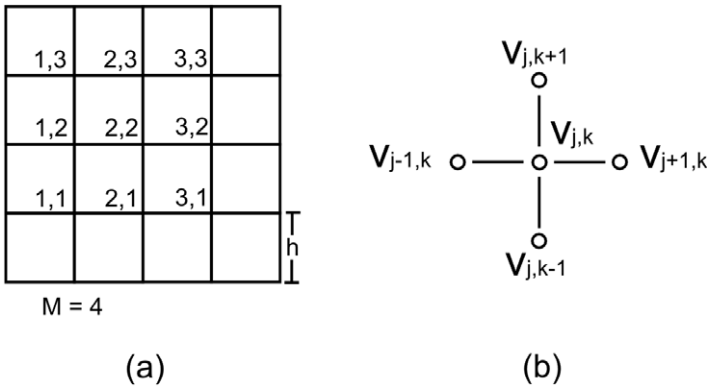


**Figure 1.** Discretization of the square domain and notation for indexing the nodes.

We discretize this equation using a grid with mesh size $h = 1/M$; see figure 1. Each node is indexed $u_{j,k}$, $j, k \in \{1, 2, \ldots, M\}$ (figures 1(a) and (b)). We approximate the second derivatives using

$$\frac{\partial^2 u}{\partial x^2}(x, y) \approx \frac{u(x-h, y) - 2u(x, y) + u(x+h, y)}{h^2},$$

$$\frac{\partial^2 u}{\partial y^2}(x, y) \approx \frac{u(x, y-h) - 2u(x, y) + u(x, y+h)}{h^2}.$$

Omitting the truncation error, and denoting the discretized Laplacian by $-\Delta_h$ we are led to solve

$$h^{-2}\left((-v_{j-1,k} + 2v_{j,k} - v_{j+1,k}) + (-v_{j,k-1} + 2v_{j,k} - v_{j,k+1})\right) = f_{j,k}, \tag{8}$$

where $f_{j,k} = f(jh, kh)$, $j, k = 1, 2, \ldots, M-1$ and $v_{j,k} = 0$ if $j$ or $k \in \{0, M\}$ i.e. when we have a point that belongs to the boundary.

Using the fact that the solution is zero at the boundary, we reindex (8) to obtain

$$h^{-2}(4v_i - v_{i-1} - v_{i+1} - v_{i-M+1} - v_{i+M-1}) = f_i, \quad i = 1, 2, \ldots, (M-1)^2. \tag{9}$$

Equivalently, we denote this system by

$$-\Delta_h \vec{v} = \vec{f}_h,$$

where $\Delta_h$ is the discretized Laplacian.

For example, when $M = 4$, as in figure 1, we have that $\vec{v} = [v_1, \ldots, v_9]^{\mathrm{T}}$. Furthermore (9) becomes

$$h^{-2}A\begin{pmatrix} v_1 \\ \vdots \\ v_9 \end{pmatrix} := h^{-2}\begin{pmatrix} B & -I & \\ -I & B & -I \\ & -I & B \end{pmatrix}\begin{pmatrix} v_1 \\ \vdots \\ v_9 \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_9 \end{pmatrix}, \tag{10}$$

where $I$ is the $3 \times 3$ identity matrix, $B$ is

$$\begin{pmatrix} 4 & -1 & \\ -1 & 4 & -1 \\ & -1 & 4 \end{pmatrix}.$$

$A$ is a Hermitian matrix with a particular block structure that is independent of $M$.

In particular, on a square grid with mesh size $h = 1/M$ we have

$$-\Delta_h = h^{-2}A \tag{11}$$

and $A$ can be expressed in terms of $L_h$ as follows:

$$A = \begin{pmatrix} L_h + 2I & -I & 0 & \cdots & \cdots & 0 \\ -I & L_h + 2I & -I & 0 & \cdots & 0 \\ 0 & -I & \ddots & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & -I & 0 \\ \vdots & \vdots & 0 & -I & L_h + 2I & -I \\ 0 & 0 & \cdots & 0 & -I & L_h + 2I \end{pmatrix} \tag{12}$$

and its size is $(M-1)^2 \times (M-1)^2$ [31].

Recall that $L_h$ is the $(M-1) \times (M-1)$ matrix shown in (6) and $I$ is the $(M-1) \times (M-1)$ identity matrix. Moreover, $A$ can be expressed using Kronecker products as follows:

$$A = L_h \otimes I + I \otimes L_h. \tag{13}$$

### 3.3. *d* dimensions

We now consider the problem in $d$ dimensions. Consider the Laplacian

$$\Delta = \sum_{k=1}^{d} \frac{\partial^2}{\partial x_k^2}.$$

We discretize $\Delta$ on a grid with mesh size $h = 1/M$ using divided differences.

As before, this leads to a system of linear equations

$$-\Delta_h \vec{v} = \vec{f}_h. \tag{14}$$

Note that $-\Delta_h = h^{-2}A$ is a symmetric positive definite matrix and $A$ is given by

$$A = \underbrace{L_h \otimes I \otimes \cdots \otimes I}_{d \text{ matrices}} + I \otimes L_h \otimes I \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes L_h,$$

and has size $(M-1)^d \times (M-1)^d$. $L_h$ is the $(M-1) \times (M-1)$ matrix shown in (6) and $I$ is the $(M-1) \times (M-1)$ identity matrix. See [31] for details.

Observe that the matrix exponential has the form

$$\mathrm{e}^{\mathrm{i}A\gamma} = \underbrace{\mathrm{e}^{\mathrm{i}L_h\gamma} \otimes \cdots \otimes \mathrm{e}^{\mathrm{i}L_h\gamma}}_{d \text{ matrices}}, \tag{15}$$

for all $\gamma \in \mathbb{R}$, where $\mathrm{i} = \sqrt{-1}$. We will use this fact later to derive the quantum circuit solving the linear system.

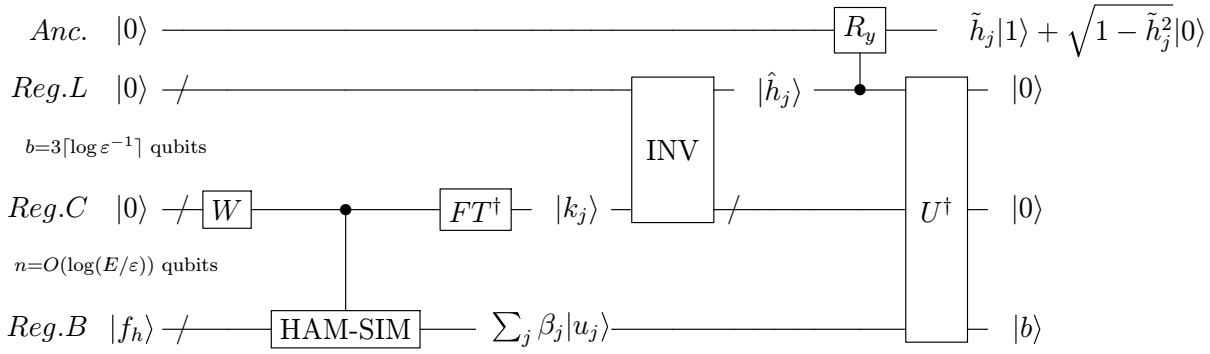


**Figure 2.** Overview of the circuit for solving the Poisson equation. Wires with '/' represent registers or groups of qubits. $W$ denotes the Walsh–Hadamard transform which applies a Hadamard gate on every qubit of the register. FT represents the quantum Fourier transform. 'HAM-SIM' is the Hamiltonian simulation subroutine that implements the operation $e^{-2\pi i\Delta_h/E}$. 'INV' is the subroutine that computes $\lambda^{-1}$. $U^\dagger$ represent uncomputation, which is the adjoint of all the operations before the controlled $R_y$ rotation.

## 4. Quantum circuit

We derive a quantum circuit solving the system $-\Delta_h\vec{v} = \vec{f}_h$, where $h = 1/M$ and without loss of generality we assume that $M$ is a power of two. We obtain a solution of the system with error $O(\varepsilon)$. The steps below are similar to those in [25].

(i) As in [25] assume the right-hand side vector $\vec{f}_h$ has been prepared quantum mechanically as a quantum state $|f_h\rangle$ and stored in the quantum register $B$. Note $|f_h\rangle = \sum_{j=0}^{(M-1)^d-1} \beta_j|u_j\rangle$, where $|u_j\rangle$ denote the eigenstates of $-\Delta_h$ and $\beta_j$ are the coefficients.

(ii) Perform phase estimation using the state $|f_h\rangle$ in the bottom register and the unitary matrix $e^{-2\pi i\Delta_h/E}$, where $\log_2 E = \lceil \log d \rceil + \log(4M^2)$. The number of qubits in the top register of phase estimation is $n = O(\log(E/\varepsilon))$.

(iii) Compute an approximation of the inverse of the eigenvalues $\lambda_j$. Store the result on a register $L$ composed of $b = 3\lceil \log \varepsilon^{-1} \rceil$ qubits (figure 2). The approximation error of the reciprocals is at most $\varepsilon$.

(iv) Introduce an ancilla qubit to the system. Apply a controlled rotation on the ancilla qubit. The rotation operation is controlled be the register $L$ which stores the reciprocals of the eigenvalues of $-\Delta_h$ (figure 2). The controlled rotation results in $\sqrt{1-(C_d/\lambda_j)^2}|0\rangle + (C_d/\lambda_j)|1\rangle$, where $C_d$ is a constant.

(v) Uncompute all other qubits on the system except the qubit introduced on the previous item.

(vi) Measure the ancilla qubit. If the outcome is 1, the bottom register of phase estimation collapses to the state $\sum_{j=0}^{(M-1)^d-1} \beta_j {\lambda_j}^{-1}|u_j\rangle$ up to a normalization factor, where $|u_j\rangle$ denote the eigenstates of $-\Delta_h$. This is equal to the normalized solution of the system. If the outcome is 0, the algorithm has failed and we have to repeat it. An alternative would be to include amplitude amplification to boost the success probability. Amplitude amplification has been considered in the literature extensively and we do not deal with it here.

### 4.1. Error analysis

We carry out the error analysis to obtain the implementation details. For $d = 1$ the eigenvalues of the second derivative are

$$4M^2 \sin^2(j\pi/(2M)), \quad j = 1, \ldots, M-1.$$

For $d > 1$, the eigenvalues of $-\Delta_h$ are given by sums of the one-dimensional eigenvalues, i.e.

$$\sum_{k=1}^{d} \left[4M^2 \sin^2(j_k\pi/(2M))\right], \quad j_k = 1, \ldots, M-1, \quad k = 1, \ldots, d.$$

We consider them in non-decreasing order and denote them by $\lambda_j$, $j = 1, \ldots, (M-1)^d$. Then $\lambda_1 = 4dM^2\sin^2(\pi/(2M))$ is the minimum eigenvalue and $\lambda_{(M-1)^d} = 4dM^2\sin^2(\pi(M-1)/(2M)) \leqslant 4dM^2$ is the maximum eigenvalue.

Define $E$ by

$$\log_2 E = \lceil \log_2 d \rceil + \log_2(4M^2). \tag{16}$$

Then the eigenvalues are bounded from above by $E$. Recall that we have already assumed that $M$ is a power of two. Then $E = 2^{\lceil \log_2 d \rceil} 4M^2 \in \mathbb{N}$.

Note that the implementation accuracy of the eigenvalues determines the accuracy of the system solution.

Our algorithm uses approximations $\hat{\lambda}_j$, such that $|\lambda_j - \hat{\lambda}_j| \leqslant \frac{17E}{2^\nu} \leqslant \varepsilon$; see theorem B.2 in appendix B. We use $n = \log_2 E + \nu$ bits to represent each eigenvalue, of which the most significant $\log_2 E$ bits hold each integer part and the remaining bits hold each fractional part. Without loss of generality, we can assume that $2^\nu \gg E$. More precisely, we consider an approximation $\hat{\Delta}_h$ of matrix $\Delta_h$ such that the two matrices have the same eigenvectors while their eigenvalues differ by at most $\varepsilon$.

We use phase estimation with the unitary matrix $e^{-i\hat{\Delta}_h t_0/E}$ whose eigenvalues are $e^{2\pi i\hat{\lambda}_j t_0/(E2\pi)}$. Setting $t_0 = 2\pi$ we obtain the phases $\phi_j = \hat{\lambda}_j/E \in [0, 1)$. The initial state of phase estimation is (figure 2)

$$|0\rangle^{\otimes n}|f_h\rangle = \sum_{j=1}^{(M-1)^d} \beta_j |0\rangle^{\otimes n}|u_j\rangle,$$

where $|u_j\rangle$ is the $j$th eigenvector of $-\Delta_h$ and $\beta_j = \langle u_j|f_h\rangle$, for $j = 1, 2, \ldots, (M-1)^d$. Since we are using finite bit approximations of the eigenvalues, we have

$$\phi_j = \frac{\hat{\lambda}_j}{E} = \frac{\hat{\lambda}_j 2^\nu}{2^n}.$$

Then $\phi_j\, 2^n$ is an integer and phase estimation succeeds with probability 1 (see [36, section 5.2, p 221] for details).

The state prior to the application of the inverse Fourier transform in the phase estimation is

$$\sum_{j=1}^{(M-1)^d} \beta_j \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i\phi_j k}|k\rangle|u_j\rangle. \tag{17}$$

After the application of the inverse Fourier transform to the first $n$ qubits we obtain

$$\sum_{j=1}^{(M-1)^d} \beta_j |k_j\rangle|u_j\rangle,$$

where

$$k_j = 2^n \phi_j = 2^n \hat{\lambda}_j / E = \hat{\lambda}_j 2^\nu \in \mathbb{N}. \tag{18}$$

Now we need to compute the reciprocals of the eigenvalues. Observe that

$$\begin{aligned}\lambda_1/d &= 4M^2 \sin^2(\pi/(2M)) = 4M^2(\pi/(2M) + O(M^{-3}))^2 \\ &= \pi^2 + O(M^{-2}) > 5,\end{aligned}$$

where the last inequality holds trivially for sufficiently large $M$. This implies $\hat{\lambda}_j/C_d \geqslant \hat{\lambda}_1/C_d \geqslant 4$, where $C_d = 2^{\lfloor \log_2 d \rfloor}$, for sufficiently large $M$. We obtain $k_j = 2^n \hat{\lambda}_j / E \geqslant \hat{\lambda}_1 \geqslant 4C_d$.

Append $b$ qubits initialized to $|0\rangle$ on the left (Reg. $L$ in figure 2), to obtain

$$\sum_{j=1}^{(M-1)^d} \beta_j |0\rangle^{\otimes b} |k_j\rangle |u_j\rangle.$$

Note that from (18) $k_j$, $\hat{\lambda}_j$ and $\hat{\lambda}_j/C_d$ have the same bit representation. The difference between the integer $k_j$ and the other two numbers is the location of the decimal point; it is located after the most significant $\log_2 E$ bit in $\hat{\lambda}_j$, and after the most significant $\log_2(E/C_d)$ bit in $\hat{\lambda}_j/C_d$. Therefore, we can use the labels $|k_j\rangle$, $|\hat{\lambda}_j\rangle$ and $|\hat{\lambda}_j/C_d\rangle$ interchangeably, and write the state above as

$$\sum_{j=1}^{(M-1)^d} \beta_j |0\rangle^{\otimes b} |\hat{\lambda}_j/C_d\rangle |u_j\rangle.$$

Now we need to compute $h_j := h(\hat{\lambda}_j/C_d) = C_d/\hat{\lambda}_j$. We do this using Newton iteration. We explain the details in section 4.2. We obtain an approximation $\hat{h}_j$ such that

$$\left|\hat{h}_j - h_j\right| \leqslant \varepsilon_0^2, \tag{19}$$

where $\varepsilon_0 = \min\{\varepsilon, E^{-1}\}$. We store this approximation in the register composed of the leftmost $b = 3\lceil \log_2 \varepsilon_0^{-1} \rceil$ qubits.

This leads to the state

$$\sum_{j=1}^{(M-1)^d} \beta_j |\hat{h}_j\rangle |\hat{\lambda}_j/C_d\rangle |u_j\rangle.$$

We append, on the left, a qubit initialized at $|0\rangle$ (Anc. in figure 2). We get

$$\sum_{j=1}^{(M-1)^d} \beta_j |0\rangle |\hat{h}_j\rangle |\hat{\lambda}_j/C_d\rangle |u_j\rangle.$$

We need to perform the conditional rotation

$$R|0\rangle|\omega\rangle = \left(\omega|1\rangle + \sqrt{1-\omega^2}|0\rangle\right)|\omega\rangle, \quad 0 < \omega < 1.$$

For this, we will approximate the first qubit by

$$\omega'|1\rangle + \sqrt{1-(\omega')^2}|0\rangle,$$

with $|\omega - \omega'| \leqslant \varepsilon_1^2$, $\varepsilon_1 = \min\{\varepsilon, 1/(4M^2)\}$. We discuss the cost of implementing this approximation in section 4.3.

The result of approximating the conditional rotation is to obtain $|\tilde{h}_j\rangle$, where $\tilde{h}_j$ is a $q = \Theta(\log_2 \varepsilon_1^{-1})$ bit number less than 1 satisfying $|\tilde{h}_j - \hat{h}_j| \leqslant \varepsilon_1^2$ and, therefore,

$$|\tilde{h}_j - h_j| \leqslant \varepsilon_0^2 + \varepsilon_1^2, \tag{20}$$

for each $j = 1, \ldots, (M-1)^d$.

Ignoring the ancilla qubits needed for implementing the approximation of the conditional rotation, we have the state

$$\sum_{j=1}^{(M-1)^d} \beta_j \left(\tilde{h}_j|1\rangle + \sqrt{1-\tilde{h}_j^2}|0\rangle\right)|\hat{h}_j\rangle|\hat{\lambda}_j/C_d\rangle|u_j\rangle.$$

Uncomputing all the qubits except the leftmost gives the state

$$|\psi\rangle := \sum_{j=1}^{(M-1)^d} \beta_j \left(\tilde{h}_j|1\rangle + \sqrt{1-\tilde{h}_j^2}|0\rangle\right)|0\rangle^{\otimes b}|0\rangle^{\otimes n}|u_j\rangle.$$

Let $P_1 = |1\rangle\langle 1| \otimes I$ be the projection acting non-trivially on the first qubit. The system $-\Delta_h \vec{v} = \vec{f}_h$ has solution $\sum_{j=1}^{(M-1)^d} \beta_j \frac{1}{\lambda_j}|u_j\rangle$. We derive the error as follows:

$$\begin{aligned}
&C_d^{-1}\left\|\sum_{j=1}^{(M-1)^d} b_j \frac{C_d}{\lambda_j}|1\rangle|0\rangle^{\otimes(b+n)}|u_j\rangle - P_1|\psi\rangle\right\| \\
&= C_d^{-1}\left\|\sum_{j=1}^{(M-1)^d} \beta_j \frac{C_d}{\lambda_j}|u_j\rangle - \sum_{j=1}^{(M-1)^d} \beta_j \tilde{h}_j|u_j\rangle\right\| \\
&= C_d^{-1}\left\|\sum_{j=1}^{(M-1)^d} \beta_j \frac{C_d}{\lambda_j}|u_j\rangle - \sum_{j=1}^{(M-1)^d} \beta_j (\tilde{h}_j - h_j + h_j)|u_j\rangle\right\| \\
&\leqslant \left\|\sum_{j=1}^{(M-1)^d} \beta_j \left(\frac{1}{\lambda_j} - \frac{1}{\hat{\lambda}_j}\right)|u_j\rangle\right\| + \varepsilon_0^2 + \varepsilon_1^2 \leqslant \frac{17E}{2^v} + \varepsilon_0^2 + \varepsilon_1^2,
\end{aligned} \tag{21}$$

where the second from last inequality is obtained using (20) and the last inequality is due to the fact that

$$\left|\frac{1}{\lambda} - \frac{1}{\hat{\lambda}}\right| \leqslant |\lambda - \hat{\lambda}|, \quad \lambda, \hat{\lambda} > 1.$$

Setting $v = \lceil \log_2(17E/\varepsilon) \rceil$ gives error $\varepsilon(1 + o(1))$ and the number of matrix exponentials used by the algorithm is $O(\log_2(E/\varepsilon))$. Therefore, if we measure the first qubit of the state $|\psi\rangle$ and the outcome is 1 the state collapses to a normalized solution of the linear system.

### *4.2. Computation of* $\lambda^{-1}$

In this part we deal with the computation of the reciprocals of the eigenvalues, which is marked as the 'INV' module in figure 2. For this we use Newton iteration to approximate $v^{-1}$, $v > 1$. We perform $s$ iterative steps and obtain the approximation $\hat{x}_s$. The input and the output of each iterative step are $b$ bit numbers. All of the calculations in each step are performed in fixed

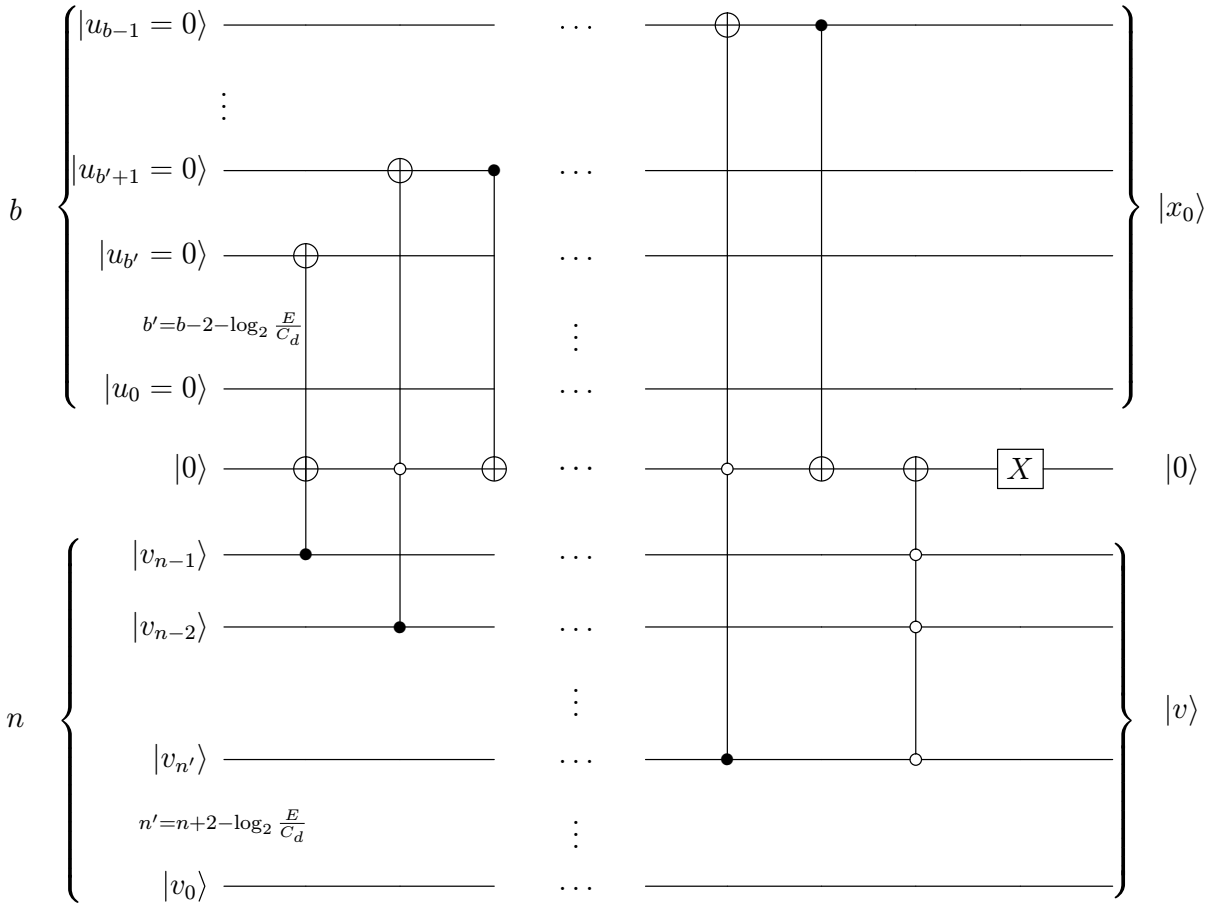


**Figure 3.** The quantum circuit computing the initial approximation $\hat{x}_0 = 2^{-p}$ of Newton iteration for approximating $v^{-1}$, $2^{p-1} \leqslant v \leqslant 2^p$. See appendix A for definitions of the basic gates.

precision arithmetic. The initial approximation is $\hat{x}_0 = 2^{-p}$, $2^{p-1} < v \leqslant 2^p$. (We use the notation $\hat{x}_i$ to emphasize that these values have been obtained by truncating a quantity $x_i$ to $b$ bits of accuracy, $i = 0, \ldots, s$.)

Theorem B.1 of appendix B gives the error of Newton iteration which is

$$|\hat{x}_s - v^{-1}| \leqslant \varepsilon_0^2 \leqslant \varepsilon,$$

where we have $\varepsilon_0 = \min\{\varepsilon, E^{-1}\}$, $s = \lceil \log_2 \log_2(2/\varepsilon_0^2) \rceil$ and the number of bits satisfies $b \geqslant 2\lceil \log_2 \varepsilon_0^{-1} \rceil + O(\log_2 \log_2 \log_2 \varepsilon_0^{-1})$.

Therefore, it suffices that the module of the quantum circuit that computes $1/\lambda_j$ carries each iterative step with $3\lceil \log_2 \varepsilon_0^{-1} \rceil$ qubits of accuracy.

The quantum circuit computing the initial approximation $\hat{x}_0$, of the Newton iteration is given in figure 3. The second register holds $|v\rangle$ and is $n$ qubits long, of which the first $\log_2(E/C_d)$ qubits represent the integer part of $v$ and the remaining ones its fractional part. The first register is $b$ qubits long. Recall that $\hat{\lambda}_j/C_d \geqslant 4$. So input values below 4 do not correspond to meaningful eigenvalue estimates and we do not need to compute their reciprocals altogether; they can be ignored. Hence the circuit implements the unitary transformation

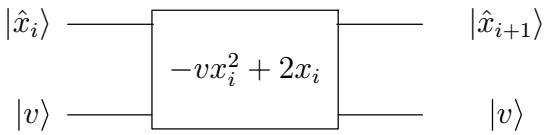


**Figure 4.** Circuit implementing each iterative step of the Newton method.

$|0\rangle^{\otimes b}|v\rangle \to |0\rangle^{\otimes b}|v\rangle$, if the first $\log_2(E/C_d) - 2$ bits of $v$ are all zero. Otherwise, it implements the initial approximation $\hat{x}_0$ through the transformation $|0\rangle^{\otimes b}|v\rangle \to |\hat{x}_0\rangle|v\rangle$.

Each iteration step $x_{i+1} = -vx_i^2 + 2x_i$ is implemented using a quantum circuit of the form shown in figure 4 that computes $|\hat{x}_i\rangle|v\rangle \to |\hat{x}_{i+1}\rangle|v\rangle$. This involves quantum circuits for addition and multiplication which have been studied in the literature [37].

The register holding $|v\rangle$ is $n$ qubits long and the register holding the $|\hat{x}_i\rangle$ and $|\hat{x}_{i+1}\rangle$ is $b$ qubits long. Note that internally the modules performing the iteration steps may use more than $b$ qubits, say, double precision, so that the addition and multiplication operations required in the iteration are carried out exactly and then return the most significant $b$ qubits of the result. The total number of qubits required for the implementation of each of these modules is $O(\log \varepsilon_0^{-1})$ and the total number of gates is a low degree polynomial in $\log \varepsilon_0^{-1}$.

### *4.3. Controlled rotation*

We now consider the implementation of the controlled rotation

$$R|0\rangle|\omega\rangle = \left(\omega|1\rangle + \sqrt{1-\omega^2}|0\rangle\right)|\omega\rangle, \quad 0 < \omega < 1.$$

Assume for a moment that we have obtained $|\theta\rangle$, a $q$ qubit state, corresponding to an angle $\theta$ such that $\sin\theta$ approximates $\omega$. Then we can use controlled rotations $R_y$ about the $y$-axis to implement $R$. We consider the binary representation of $\theta$ and have

$$\theta = \theta_1 \ldots \theta_q = \sum_{j=1}^{q} \theta_j 2^{-j}, \quad \theta_j \in \{0, 1\}.$$

Then

$$R_y(2\theta) = \mathrm{e}^{-\mathrm{i}\theta Y} = \begin{pmatrix} \sqrt{1-\sin^2\theta} & -\sin\theta \\ \sin\theta & \sqrt{1-\sin^2\theta} \end{pmatrix}$$
$$= \prod_{j=1}^{q} \mathrm{e}^{-\mathrm{i}Y\theta_j/2^j} = \prod_{j=1}^{q} R_y^{\theta_j}\left(2^{1-j}\right),$$

where $Y$ is the Pauli $Y$ operator and $\theta \in [0, \pi/2]$. The detailed circuit is shown in figure 5.

We now turn to the algorithm that calculates $|\theta\rangle$ from $|\omega\rangle$. Since $\omega$ corresponds to the reciprocal of an approximate eigenvalue of the discretized Laplacian, we know that $\sin^{-1}(\omega)$ belongs to the first quadrant and $\sin^{-1}(\omega) = \Omega(1/M^2)$. Therefore, we can find an angle $\theta$ such that $|\sin(\theta) - \omega| \leqslant \varepsilon_1^2$, $\varepsilon_1 = \min\{1/(4M^2), \varepsilon\}$, using bisection and an approximation of the sine function.

In appendix B we show the error in approximating the sine function using fixed precision arithmetic. In section 5 we show the details of the resulting quantum algorithm computing the

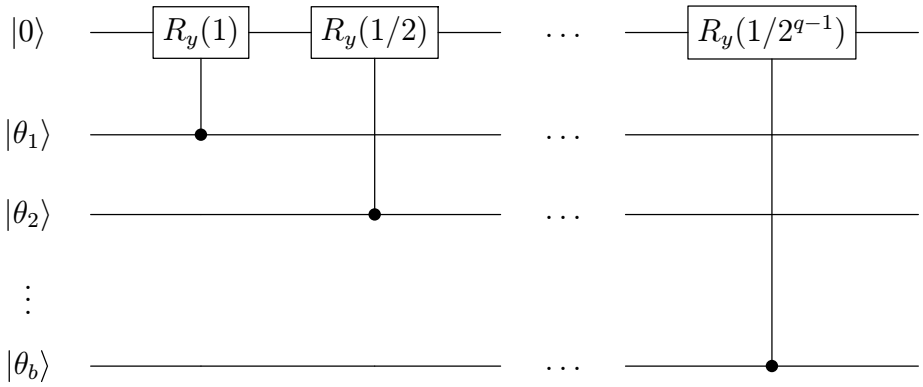


**Figure 5.** Circuit for executing the controlled $R_y$ rotation. See appendix A for definitions of basic gates.

approximation to the sine function. These results, with a minor adjustment in the number of bits needed can be used here. We will not deal with the details of the quantum algorithm for the sine function in this section since we present them in section 5 that deals with the simulation of Poisson's matrix. We will only describe the steps of the algorithm and its cost.

### Algorithm

(i) Take as an initial approximation of $\theta$ the value $\pi/4$.

(ii) Approximate the $\sin(\theta)$ with error $\varepsilon_1^2/2$ using our algorithm for the sine function (details in section 5 and appendix B). Let $s_\theta$ denote this approximation.

(iii) If $s_\theta < \omega - \varepsilon_1^2/2$, set $\theta$ to be the midpoint of the right subinterval.

(iv) If $s_\theta > \omega + \varepsilon_1^2/2$, set $\theta$ to be the midpoint of the left subinterval.

(v) Repeat the steps 2–4 $\lceil \log_2 \varepsilon_1^{-2} \rceil + 1$ times.

An evaluation at the midpoint of an interval yields a value that satisfies either the condition of step 3, or that of step 4, or $|s_\theta - \omega| \leqslant \varepsilon_1^2/2$. If at any time both the conditions of steps 3 and 4 are false then $\theta$ will not change its value until the end. Then, at the end, we have $|\sin(\theta) - \omega| \leqslant |\sin(\theta) - s_\theta| + |s_\theta - \omega| \leqslant \varepsilon_1^2$, since the error in computing the sine is $\varepsilon_1^2/2$. On the other hand, if $\theta$ is updated until the very end of the algorithm the final value of theta also satisfies $|\sin(\theta) - w| \leqslant \varepsilon_1^2$, because in the final interval we have $|\sin(\theta) - \omega| \leqslant |\theta - \sin^{-1}(\omega)| \leqslant \varepsilon_1^2$.

In a way similar to that of propositions 1 and 2 of appendix B we carry out the steps of the algorithm in $q$ bit fixed precision arithmetic, $q = \max\{2\nu + 9, 13 + \nu + 2\log_2 M\}$ and sufficiently large $\nu$ to satisfy the accuracy requirements. (The last expression for $q$ is slightly different form that in proposition 2 because it accounts for the fact that in the case we are dealing with here the angle is $\Omega(1/M^2)$.) This gives us an approximation to the sine with error $2^{-(\nu-1)}$. We set

$$\nu = \lceil \log_2 \varepsilon_1^{-2} \rceil + 1.$$

Thus $\nu$ and $q$ are both $\Theta(\log_2 \varepsilon_1)$.

The algorithm for the sine function is based on an approximation of the exponential function using repeated squaring. Each square requires $O(q^2)$ quantum operations and $O(q)$

qubits. This is repeated $\nu$ times before the approximation to the sine is obtained. Thus the cost of one bisection step requires $O(\nu q^2)$ quantum operations and $O(\nu q)$ qubits. So, in terms of $\varepsilon_1$, the total cost of bisection is proportional to $(\log_2 \varepsilon_1^{-1})^4$ quantum operations and $(\log_2 \varepsilon_1^{-1})^3$ qubits.

## 5. Hamiltonian simulation of the Poisson matrix

In this section we deal with the implementation of the 'HAM-SIM' module (figure 2) which effectively applies $e^{-i\hat{\Delta}_h t_0}$ onto register $B$. In our case the eigenvectors of the discretized Laplacian are known and we use approximations of the eigenvalues. From (11) and (15) we have

$$e^{-i\Delta_h \gamma} = \underbrace{e^{ih^{-2}L_h\gamma} \otimes \cdots \otimes e^{ih^{-2}L_h\gamma}}_{d \text{ matrices}}. \tag{22}$$

Thus it suffices to implement $e^{ih^{-2}L_h\gamma}$, for certain $\gamma \in \mathbb{R}$, $\gamma = 2\pi \cdot 2^t/E$, $t = 0, 1, \ldots, \log_2 E - 1$ that are required in phase estimation. This can be accomplished by considering the spectral decomposition $S\Lambda S$ of the matrix $L_h$, where $S$ is the matrix of the sine transform [31, 40]. Then $S$ can be implemented using the quantum Fourier transform. We will implement an approximation of $\Lambda$.

We remark that the quantum circuits presented here can be used in the simulation of the Hamiltonian $-\Delta + V$ using splitting formulas. For results concerning Hamiltonian simulation using splitting formulas see [9, 28, 41].

### 5.1. One-dimensional case

We start with the implementation of $e^{ih^{-2}L_h\gamma}$, $\gamma = 2\pi 2^t/E$, $E = 4M^2$ when $d = 1$ and $t = 0, 1, \ldots, n-1$, where $n$ is the number of qubits in register $C$; see (17). The form of $L_h$ is shown in (6) and is positive definite. It is a Toeplitz matrix and it is known that this type of matrix can be diagonalized via the sine transform $S$ [42]. We have $L_h = S\Lambda S$, where $\Lambda$ is an $(M-1)\times(M-1)$ diagonal matrix containing the eigenvalues $4\sin^2(j\pi/(2M))$, $j = 1, \ldots, M-1$, of $L_h$ and $S = \{S_{i,j}\}_{i,j=1,2,\ldots,M-1}$ is the sine transform where $S_{i,j} = \sqrt{\frac{2}{M}}\sin\left(\frac{\pi ij}{M}\right)$, $i, j$=1, …, $M-1$. Thus

$$e^{ih^{-2}L_h\gamma} = S\, e^{ih^{-2}\Lambda\gamma} S. \tag{23}$$

The relationship between the sine and cosine transforms and the Fourier transform can be found in [40, theorem 3.10].

In particular, using the notation in [40], we have

$$T_M^{\dagger} F_{2M} T_M = C_{M+1} \oplus (-iS_{M-1}) = \begin{pmatrix} C_{M+1} & 0 \\ 0 & -iS_{M-1} \end{pmatrix}, \tag{24}$$

where $C_{M+1}$, $S_{M-1}$ denote the cosine and sine transforms, and the subscripts $M-1$ and $M+1$ emphasize the size of the respective matrix. $F_{2M}$ is the $2M \times 2M$ matrix of the Fourier

transform. The matrix $T_M$ has size $2M \times 2M$ and is given by (25)

$$T_M = \begin{pmatrix} 1 & & & & & & \\ & \frac{1}{\sqrt{2}} & & & \frac{\mathrm{i}}{\sqrt{2}} & & \\ & & \ddots & & & \ddots & \\ & & & \frac{1}{\sqrt{2}} & & & \frac{\mathrm{i}}{\sqrt{2}} \\ & & & & 1 & & \\ & & & \frac{1}{\sqrt{2}} & & & -\frac{\mathrm{i}}{\sqrt{2}} \\ & & \vdots & & & \vdots & \\ & \frac{1}{\sqrt{2}} & & & -\frac{\mathrm{i}}{\sqrt{2}} & & \end{pmatrix}. \tag{25}$$

The quantum circuits for implementing the unitary transformation $T_M$ is discussed in [38]. The action of $T_M$ can be described by [38]

$$\begin{aligned} T_M|0x\rangle &= \frac{1}{\sqrt{2}}|0x\rangle + \frac{1}{\sqrt{2}}|1x'\rangle, \\ T_M|1x\rangle &= \frac{\mathrm{i}}{\sqrt{2}}|0x\rangle - \frac{\mathrm{i}}{\sqrt{2}}|1x'\rangle, \end{aligned} \tag{26}$$

where $\mathrm{i}^2 = -1$, $x$ is an $n$-bit number ranging $1 \leqslant x < 2^n$ and $x'$ denotes its complement of 2 i.e. $x' = 2^n - x$. The basic idea of implementing $T_M$ is to separate its operation into an operator $D$, which ignores the complement of 2 in $T_M$, and a controlled permutation $\pi$, which transforms the state $|bx\rangle$ to $|bx'\rangle$ only if $b$ is 1. Therefore the action of $D$ and $\pi$ can be written as

$$\begin{aligned} D|0x\rangle &= \frac{1}{\sqrt{2}}|0x\rangle + \frac{1}{\sqrt{2}}|1x\rangle, \\ D|1x\rangle &= \frac{\mathrm{i}}{\sqrt{2}}|0x\rangle - \frac{\mathrm{i}}{\sqrt{2}}|1x\rangle, \\ \pi|0x\rangle &= |0x\rangle, \\ \pi|1x\rangle &= |1x'\rangle. \end{aligned} \tag{27}$$

Clearly, $T_M = D\pi$ and the overall circuit for implementing operation $T_M$ is shown in figure 8.

By (24) the sine transform $S$ can be implemented by cascading the quantum circuits in figure 8 with the circuit for the Fourier transform [36]. An ancilla bit is added to register $b$. It is kept in the state $|1\rangle$ in order to select the lower-right block

$$\begin{pmatrix} a & 0 \\ 0 & -\mathrm{i}S_{M-1} \end{pmatrix} \tag{28}$$

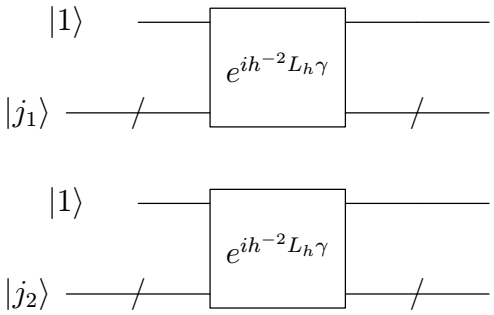


**Figure 6.** Quantum circuit for implementing $e^{-i\Delta_h\gamma}$, $\gamma \in \mathbb{R}$ for the two-dimensional discrete Poisson equation. The subroutine of $e^{ih^{-2}L_h\gamma}$ is shown in figure 7. The registers holding $|j_1\rangle$, $|j_2\rangle$ are $m$ qubits each.

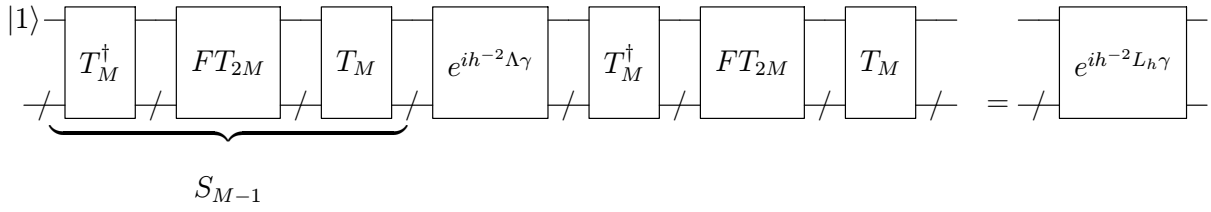


**Figure 7.** Quantum circuit for implementing $e^{ih^{-2}L_h\gamma}$, $\gamma \in \mathbb{R}$, where $L_h$ is the matrix in (6). $S_{M-1}$ represents the sine transform matrix of size $(M-1) \times (M-1)$, $M = 2^m$. This circuit acts on $m+1$ qubits.

from the unitary operation $T_M^\dagger F_{2M} T_M$ (24), $a \in \mathbb{C}$. Considering the state $|f_h\rangle$, that corresponds to the right-hand side of (6), and for $b_i = \langle i|f_h\rangle$ we have

$$(0, \underbrace{b_1, b_2, \ldots, b_{M-1}}_{\substack{\text{values on the} \\ (M-1) \text{ nodes}}}), \tag{29}$$

then the element $a$ in equation (28) has no effect, and the circuit in figure 6 is equivalent to applying $(S_{M-1}\, e^{2\pi i \Lambda 2^t/E} S_{M-1})$ onto the $(M-1)$ elements of $|f_h\rangle$. This is also equivalent to simulating the Hamiltonian $e^{2\pi i h^{-2}\Lambda 2^t/E}$ with the state $|f_h\rangle$ stored in register $b$.

We implement $e^{2\pi i h^{-2}\hat{\Lambda}2^t/E}$ where $\hat{\Lambda} = \{\hat{\lambda}_j\}_{j=1,\ldots,M-1}$ is a diagonal matrix approximating $\Lambda = \{\lambda_j\}_{j=1,\ldots,M-1}$.

We obtain each $\hat{\lambda}_j$, $j = 1, \ldots, M-1$ by the following algorithm. The general idea is to approximate $\sin x = \Im(e^{ix}) = \Im((e^{ix/r})^r)$ with $W^r$ where $W = 1 - ix/r + x^2/r^2$ is the Taylor expansion of $e^{ix/r}$ up to the second order term. $W^r$ is computed efficiently in fixed point arithmetic using repeated squaring. The detailed steps are the following.

**Eigenvalue simulation algorithm (ESA)**

(i) Let $r = 2^{\nu+7}$ where $\nu$ is positive integer which is related to the accuracy of the result. The inputs and the outputs of the modules below are $s = \max\{2\nu+9, 11+\nu+\log_2 M\}$ bit numbers. Internally the modules may carry out calculations in higher precision $O(s)$, but

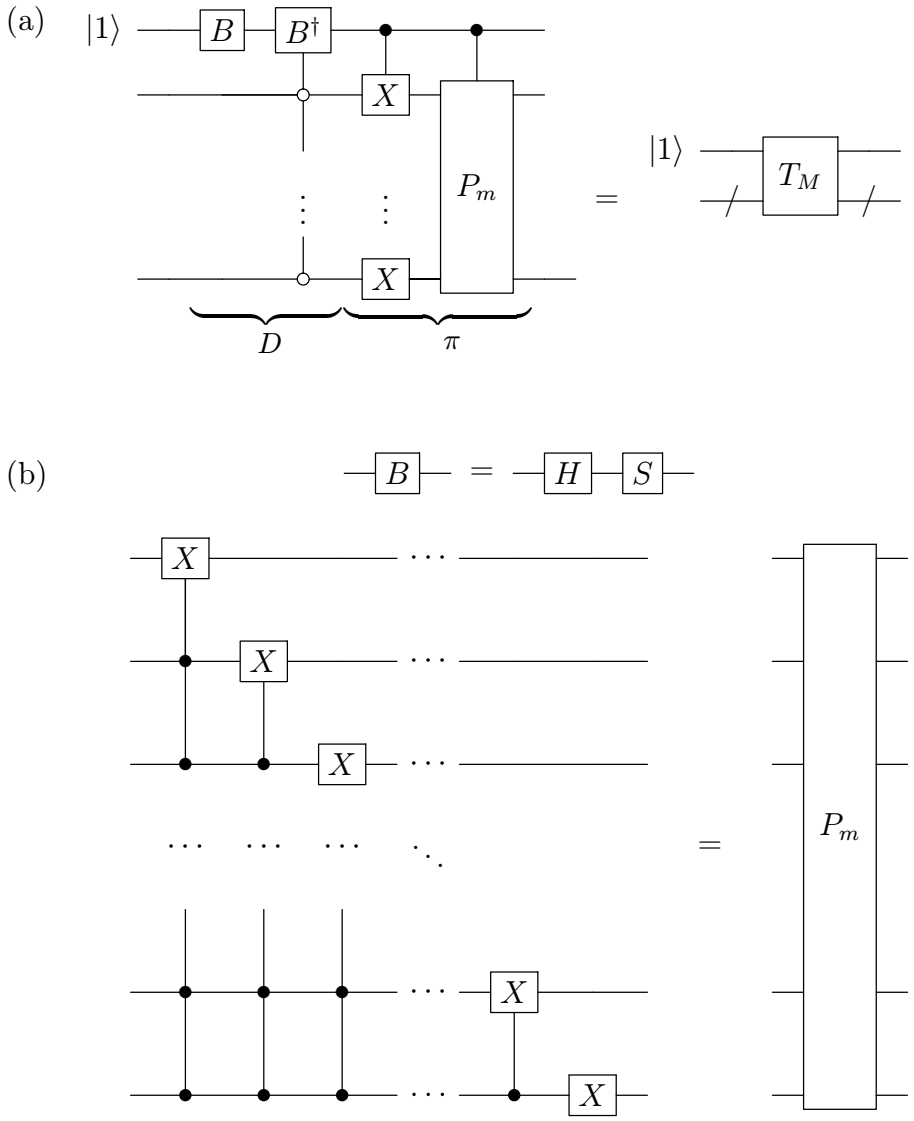


**Figure 8.** Quantum circuit for implementing $T_M$ in equations (24) and (25). In (b), $P_m$ denotes the map $|x\rangle \to |x+1 \bmod 2^n\rangle$ on $n$ qubits. Its implementation is described in [39]. See appendix A for the definitions of basic gates. (a) Generic circuit for $T_M = D\pi$, for details refer to [38]. (b) Implementation of $B$ and $P_m$ gates in (a).

the results are returned using $s$ bits. This value of $s$ follows from the error estimates in proposition B.2.

(ii) We perform the transformation

$$|j\rangle|0\rangle^{\otimes s} \to |j\rangle \underbrace{|\hat{y}_j = \hat{x}_j/r\rangle}_{s \text{ qubits}},$$

where $\hat{x}_j$ is the $s$ bit truncation of $x_j = \frac{\pi j}{2M}$. Note that $y_j = x_j/r \in (0, 1)$ and $\hat{y}_j$ is the $s$ bit truncation of $y_j$. Recall that $r \geqslant 2$ and $2M$ are powers of 2. Calculations are to be performed

in fixed precision arithmetic, so division does not actually need to be performed. All one needs to do is multiply $j$ by $\pi$ with $O(s)$ bits of accuracy, keeping in track the position of the decimal point and then take the most significant $s$ bits of the result.

(iii) We compute the real and imaginary parts of the complex number $\hat{W}_1$ by truncating, if necessary, the respective parts of $\hat{W}_0 = 1 - \hat{y}^2 + \mathrm{i}\hat{y}$ to $s$ bits of accuracy; see (B.7) in proposition B.1. This is expressed by the transformation

$$|\hat{y}_j\rangle|0\rangle^{\otimes s}|0\rangle^{\otimes s} \to |\hat{y}_j\rangle|\Re(\hat{W}_1)\rangle|\Im(\hat{W}_1)\rangle.$$

Note that since $|\hat{y}_j\rangle$ is $s$ qubits long, $\hat{W}_0$ can be computed exactly using double precision and ancilla qubits and the final result can be returned in $s$ qubits.

Complex numbers are implemented using two registers, holding the real and imaginary parts. Complex arithmetic is performed by computing the real and imaginary parts of the result.

(iv) We compute $\hat{W}_r$ approximating $\hat{W}_1^r$ using repeated squaring. Each step of this procedure is accomplished by the transformation

$$|\Re(\hat{W}_{2^j})\rangle|\Im(\hat{W}_{2^j})\rangle|0\rangle^{\otimes s}|0\rangle^{\otimes s} \to |\Re(\hat{W}_{2^j})\rangle|\Im(\hat{W}_{2^j})\rangle|\Re(\hat{W}_{2^{j+1}})\rangle|\Im(\hat{W}_{2^{j+1}})\rangle,$$

which describes the steps in (B.7). The registers holding real and imaginary parts of the numbers are $s$ qubits long.

(v) $\Im(\hat{W}_r)$ approximates $\sin(\pi j/(2M))$ with error $2^{-(\nu-1)}$. Hence $\Im^2(\hat{W}_r)$ approximates the $\sin^2(\pi j/(2M))$. We compute the square of $\Im(\hat{W}_r)$ exactly and multiply it by $4M^2$ (this involves only shifting). We keep the most significant $\nu + \log_2(4M^2)$ bits of the result, which we denote by $\ell_j$. This means that the $\log_2(4M^2)$ bits of the binary string representing $\ell_j$ compose the integer part and the last $\nu$ bits compose the fractional parts of the approximation to $\lambda_j$. Then

$$|\lambda_j - \ell_j| \leqslant 17 \times 2^{-\nu} M^2.$$

For details of the error estimate see proposition B.2. When $d = 1$, $n$ (the number of qubits in register $C$) and $\nu$ are related by $n = \nu + \log_2(4M^2)$. Moreover, in the one-dimensional case $\hat{\lambda}_j = \ell_j$.

(vi) Let $k_j$ be the binary string representing $\ell_j$. For a fixed $t$, we implement the transformation

$$\underbrace{|k_j\rangle}_{n \text{ qubits}} |0\rangle^{\otimes n} \to |k_j\rangle \underbrace{|k_j 2^t\rangle}_{n \text{ qubits}}. \tag{30}$$

This is accomplished using CNOTs with the circuit shown in figure 9, since $t \leqslant n$ the total number of quantum operations and qubits required to implement the circuit for all the values of $t$ is $O(n^2)$.

(vii) Finally, we use phase kickback (see e.g. [43]) to obtain $\mathrm{e}^{2\pi\mathrm{i}\phi_j 2^t}$ from the $|k_j 2^t\rangle$ state where $\phi_j$ is the phase corresponding to the eigenvalue $\ell_j$ that approximates $\lambda_j$; see (18).

### 5.2. *Multidimensional case*

To implement $\mathrm{e}^{-\mathrm{i}\Delta_h\gamma}$, $\gamma = 2\pi 2^t/E$, $E$ defined in (16) and $t = 0, \ldots, n-1$ we use

$$\mathrm{e}^{-\mathrm{i}\Delta_h\gamma} = \underbrace{\mathrm{e}^{\mathrm{i}h^{-2}L_h\gamma} \otimes \cdots \otimes \mathrm{e}^{\mathrm{i}h^{-2}L_h\gamma}}_{d \text{ matrices}}. \tag{31}$$

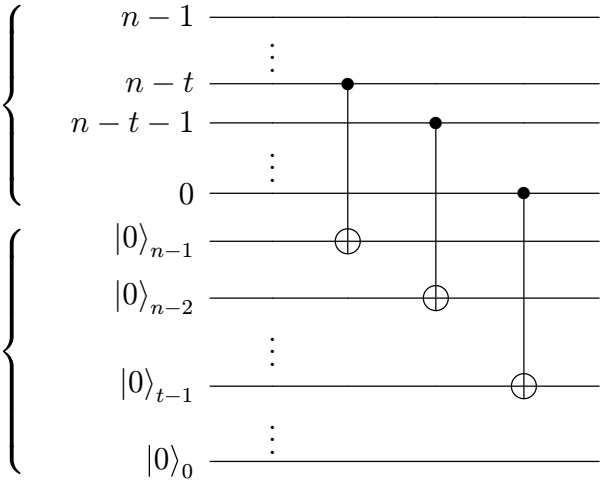


**Figure 9.** Quantum circuit for implementing the transformation in equation (30).

Therefore the quantum circuit implementing $e^{-i\Delta_h \gamma}$ in $d$ dimensions is obtained by the replication and parallel application of the circuit simulating $e^{ih^{-2}L_h\gamma}$. For example, when $d = 2$ we have the circuit in figure 6. The register $B$ of figure 2 contains $dm$ qubits, $m = \log_2 M$ and its initial state is assumed to have the form

$$\underbrace{(0,\ldots,0}_{M^d-(M-1)^d},\underbrace{b_1, b_2, \ldots, b_{(M-1)^d})}_{\substack{\text{values on the nodes of} \\ (M-1)^{(\times d)} \text{ grid}}}, \tag{32}$$

where $b_i = \langle i | f_h \rangle$. This way we select the $S_{M-1}$ block in $T_M^\dagger F_{2M} T_M$ in (24) in each circuit for $e^{ih^{-2}L_h\gamma}$. Recall that $|f_h\rangle$ corresponds to the right-hand side of (14).

The eigenvalues in the $d$-dimensional case are given as sums of the one-dimensional eigenvalues. We do not need to form the sums explicitly for the simulation of $-\Delta_h$; they are computed by the tensor products. The difference between the $d$-dimensional and one-dimensional cases is that the register $C$ in figure 2 has $\lceil \log_2 d \rceil$ additional qubits; i.e $n = \lceil \log_2 d \rceil + \log_2 4M^2 + \nu$. Accordingly, we generate the one-dimensional approximations to the eigenvalues using steps 1–5 of the eigenvalue estimation algorithm of the previous section. Then we append $\lceil \log_2 d \rceil$ qubits initialized to $|0\rangle^{\otimes \lceil \log_2 d \rceil}$ to the left of the register holding the $|\ell_j\rangle$ and carry out the remaining two steps, 6 and 7, with $n = \lceil \log_2 d \rceil + \log_2 4M^2 + \nu$. The error in the approximate eigenvalues is equal to $17M^2 d/2^\nu$; see theorem B.2.

### *5.3. Simulation cost*

Simulating the sine and cosine transforms (24) requires $O(m^2)$, $m = \log_2 M$ quantum operations and $O(m)$ qubits [38]. The diagonal eigenvalue matrix of the one-dimensional case (23) is simulated by ESA. Its steps 1–3 and 5 require $O(s^2)$ quantum operations and $O(s)$ qubits. In step 4 repeated squaring is performed $\nu + 7$ times. Each repetition or step of the procedure requires $O(s^2)$ quantum operations and $O(s)$ qubits. The total cost of step 4 is proportional to $\nu \cdot O(s^2)$ quantum operations and $\nu \cdot O(s)$ qubits, accounting for any ancilla qubits used in

repeated squaring. Step 6 requires $O(n+t)$ quantum operations and qubits for fixed $t$. Step 7 requires $O(n^2)$ quantum operations, due to the Fourier transform, and $O(n)$ qubits.

Using theorem B.2, and requiring error $\varepsilon$ in the approximation of the eigenvalues, we have

$$\frac{17E}{2^{\nu}} \leqslant \varepsilon,$$

$$\nu = \left\lceil \log_2 \frac{17E}{\varepsilon} \right\rceil,$$

i.e. $\nu = \Theta(\log_2 d + m + \log_2 \varepsilon^{-1})$. We also have $n = \Theta(\nu)$ and $s = \Theta(n)$.

We derive the simulation cost taking the following facts into account.

- Steps 1–5 deal with the approximation of the eigenvalues. These computations are not repeated for every $t = 0, \ldots, n-1$. The total cost of these steps is $O(n^3)$ quantum operations and $O(n^2)$ qubits.
- The total cost of step 6, resulting from all the values of $t$, is $O(n^2)$ quantum operations and qubits.
- The total cost of step 7, that applies phase kickback for all values of $t$, does not exceed $O(n^3)$ quantum operations and $O(n^2)$ qubits.

Therefore the total cost to simulate $\mathrm{e}^{\mathrm{i}h^{-2}L_h\gamma}$, $\gamma = 2\pi 2^t/E$, for all $t = 0, \ldots, n-1$, is $O(n^3)$ quantum operations and $O(n^2)$ qubits. From (22) we conclude that the cost to simulate Poisson's matrix for the $d$-dimensional problem is $d \cdot O(n^3)$ quantum operations and $d \cdot O(n^2)$ qubits.

Finally, we remark that the dominant component of the cost is the one resulting from the approximation of the eigenvalues (i.e. the cost of steps 1–5).

## 6. Total cost

We now consider the total cost for solving the Poisson equation (1). Discretizing the second derivative operator on a grid with mesh size $h = 1/M$ results in a system of linear equations, where the coefficient matrix is $(M-1)^d \times (M-1)^d$, i.e. exponential in the dimension $d \geqslant 1$. Solving this system using classical algorithms has a cost that grows at least as fast as the number of unknowns $(M-1)^d$. For the case $d = 2$, [31, table 6.1] summarizes the cost of direct and iterative classical algorithms solving this system.

For simulating Poisson's matrix we need $dO(n^3)$ quantum operations and $dO(n^2)$ qubits, where $n = O(\log_2 d + m + \log_2 \varepsilon^{-1})$ and $m = \log_2 M$. To this we add the cost for computing the reciprocal of the eigenvalues which is $O((\log_2 \varepsilon_0^{-1})^2 \log_2 \log_2 \varepsilon_0^{-1})$ quantum operations and $O((\log_2 \varepsilon_0^{-1}) \log_2 \log_2 \varepsilon_0^{-1})$ qubits, accounting for the $O(\log_2 \log_2 \varepsilon_0^{-1})$ Newton steps, $\varepsilon_0 = \min\{\varepsilon, E^{-1}\}$. Finally, we add the cost of the conditional rotation which is proportional to $(\log_2 \varepsilon_1^{-1})^4$ quantum operations and $(\log_2 \varepsilon_1^{-1})^3$ qubits, $\varepsilon_1 = \min\{1/(4M)^2, \varepsilon\}$.

From the above we conclude that the quantum circuit implementing the algorithm requires an order of $dO(n^3) + (\log_2 \varepsilon_1^{-1})^4$ quantum operations and $dO(n^2) + (\log_2 \varepsilon_1^{-1})^3$ qubits.

The relation between the matrix size and the accuracy is very important in assessing the performance of the quantum algorithm solving a linear system, since its cost depends

on both of these quantities [25]. In particular, for the Poisson equation we have ignored, so far, the effect of the discretization error of the Laplacian $\Delta$. If the grid is too coarse the discretization error will exceed the desired accuracy. If the grid is too fine, the matrix will be unnecessarily large. Thus the mesh size and, therefore, the matrix size should depend on $\varepsilon$, i.e. $M = M(\varepsilon)$. This dependence is determined by the smoothness of the solution $u$, which, in turn, depends on the smoothness of the right-hand side function $f$. For example, if $f$ has uniformly bounded partial derivatives up to order four, then the discretization error is $O(h^2)$ and we set $M = \varepsilon^{-1/2}$; see [30, 31] for details. In general, we have $M = \varepsilon^{-\alpha}$, where $\alpha > 0$ is a parameter depending on the smoothness of the solution. This yields $n = O(\log_2 d + \log_2 \varepsilon^{-1})$, since $m = \log_2 M = \alpha \log_2 \varepsilon^{-1}$. The resulting number of the quantum operations for the circuit is proportional to

$$\max\{d, \log_2 \varepsilon^{-1}\}(\log_2 d + \log_2 \varepsilon^{-1})^3,$$

and the number of qubits is proportional to

$$\max\{d, \log_2 \varepsilon^{-1}\}(\log_2 d + \log_2 \varepsilon^{-1})^2.$$

It can be shown that $\log_2 d = O(\log_2 \varepsilon^{-1})$ and the number of quantum operations and qubits become proportional to

$$\max\{d, \log_2 \varepsilon^{-1}\}(\log_2 \varepsilon^{-1})^3$$

and

$$\max\{d, \log_2 \varepsilon^{-1}\}(\log_2 \varepsilon^{-1})^2,$$

respectively.

Observe that the condition number of the matrix is proportional to $\varepsilon^{-2\alpha}$ and is independent of $d$. Therefore a number of repetitions proportional to $\varepsilon^{-4\alpha}$ leads to a success probability arbitrarily close to one, regardless of the value of $d$. This follows because repeating an algorithm many times increases its probability of succeeding at least according to the Chernoff bounds [36, box 3.4, p 154]. In contrast to this, the cost of any deterministic classical algorithm solving the Poisson equation is exponential in $d$. Indeed, for error $\varepsilon$ the cost is bounded from below by a quantity proportional to $\varepsilon^{-d/r}$ where $r$ is a smoothness parameter [21].

## 7. Conclusion and future directions

We present a quantum algorithm and a circuit for approximating the solution of the Poisson equation in $d$ dimensions. The algorithm breaks the curse of dimensionality and in terms of $d$ yields an exponential speedup relative to classical algorithms. The quantum circuit is scalable and has been obtained by exploiting the structure of the Hamiltonian for the Poisson equation to diagonalize it efficiently. In addition, we provide quantum circuit modules for computing the reciprocal of eigenvalues and trigonometric approximations. These modules can be used in other problems as well.

The successful development of the quantum Poisson solver opens up entirely new horizons in solving structured systems on quantum computers, such as those involving Toeplitz matrices. Hamiltonian simulation techniques [9, 28, 41] can also be combined with our algorithm to extend its applicability to PDEs, signal processing, time series analysis and other areas.

## Acknowledgments

SK and YC would like to thank the NSF CCI Center, 'Quantum Information for Quantum Chemistry (QIQC)', award number CHE-1037992 and Army Research Office (ARO) for partial support. AP, IP and JFT thank the NSF for financial support.

## Appendix A

In this paper, $X$, $Y$ and $Z$ are Pauli matrices $\sigma_x$, $\sigma_y$ and $\sigma_z$. $I$ represents the identity matrix. $H$ is the Hadamard gate and $W$, in figure 2, represents $H^{\otimes n}$ where $n$ is the number of qubits in the register. The matrix representations of other quantum gates used are the following:

$$V^\dagger = \frac{1}{2}\begin{pmatrix} 1-\mathrm{i} & 1+\mathrm{i} \\ 1+\mathrm{i} & 1-\mathrm{i} \end{pmatrix}, \quad R_{zz}(\theta) = \mathrm{e}^{\mathrm{i}\theta}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{A.1}$$

$$R_x(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & \mathrm{i}\sin\left(\frac{\theta}{2}\right) \\ \mathrm{i}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{A.2}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{i} \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\frac{\pi}{4}} \end{pmatrix}, \quad R_z(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\theta} \end{pmatrix}. \tag{A.3}$$

## Appendix B

**Theorem B.1.** *Consider the approximation $\hat{x}_s$ to $v^{-1}$, $v > 1$, using $s$ steps of Newton iteration, with initial approximation $\hat{x}_0 = 2^{-p}$, $2^{p-1} < v \leqslant 2^p$. Assume that each step takes $b$ bit numbers as inputs and produces $b$ bit outputs and that all internal calculations are carried out in fixed precision arithmetic. Then the error is*

$$|\hat{x}_s - v^{-1}| \leqslant \varepsilon_N + s2^{-b},$$

*where $\varepsilon_N$ denotes the desired error of Newton iteration without considering the truncation error, $\varepsilon_N \geqslant 2^{-2^s}$. The truncation error is given by the second term and $s \geqslant \lceil \log_2 \log_2 \varepsilon_N^{-1} \rceil$, $b > p$.*

**Proof.** Consider the function $g(x) = 1/x - v$, $x > 0$, where $g(1/v) = 0$. The Newton iteration for approximating the zero of $g$ is given by

$$x_{s+1} = \varphi(x_s) = 2x_s - vx_s^2, \quad s = 0, 1, \ldots.$$

The error $e_s = |x_s - 1/v|$ satisfies $e_{s+1} = ve_s^2$. Unfolding the recurrence we get

$$e_s \leqslant (ve_0)^{2^s}.$$

Let $x_0 = 2^{-p}$. Now consider the lowest power of 2 that is greater than or equal to $v$, i.e. $2^{p-1} < v \leqslant 2^p$. Clearly $p > 1$ since $v > 1$ and $ve_0 < 1/2$. For error $\varepsilon_N$ we have $2^{-2^s} \leqslant \varepsilon_N$, which implies $s \geqslant \lceil \log_2 \log_2 \varepsilon_N^{-1} \rceil$.

The derivative of the iteration function is decreasing and we have $|\varphi'| \leqslant 2(1 - ax_0) \leqslant 1$. We will implement the iteration using fixed precision arithmetic. We first calculate the round off error. We have

$$\begin{aligned}
\hat{x}_0 &= x_0, \\
\hat{x}_1 &= \varphi(\hat{x}_0) + \xi_1, \\
\hat{x}_2 &= \varphi(\hat{x}_1) + \xi_2, \\
&\vdots \\
\hat{x}_s &= \varphi(\hat{x}_{s-1}) + \xi_s,
\end{aligned}$$

where the $\xi_i$ denotes truncation error at the respective steps. Thus

$$\hat{x}_s - x_s = \varphi(\hat{x}_{s-1}) + \xi_s - \varphi(x_{s-1}),$$

and using the fact $|\varphi'| \leqslant 1$ we obtain

$$|\hat{x}_s - x_s| \leqslant |\hat{x}_{s-1} - x_{s-1}| + |\xi_s| \leqslant \sum_{i=1}^{s} |\xi_i| \leqslant s2^{-b},$$

assuming that we truncate the intermediate results to $b$ bits of accuracy. □

**Lemma B.1.** *Let $x \in [\pi/(2M), \pi/2)$ and $W = 1 + \mathrm{i}\frac{x}{r} - \frac{x^2}{r^2}$. Then*

$$\left|\mathrm{e}^{\mathrm{i}x} - W^r\right| \leqslant 2^7/r.$$

**Proof.** $\mathrm{e}^{\mathrm{i}x} = (\mathrm{e}^{\mathrm{i}x/r})^r = (W + E(x/r))^r$, where for $y = x/r$, $E(y) = \sum_{k\geqslant 3} \frac{(\mathrm{i}y)^k}{k!}$ and

$$\left|\sum_{k\geqslant 3} \frac{(\mathrm{i}y)^k}{k!}\right| \leqslant \sum_{k\geqslant 3} \frac{|y|^k}{k!} = |y|^3 \sum_{k\geqslant 3} \frac{|y|^{k-3}}{k!} = |y|^3 \sum_{k\geqslant 0} \underbrace{\frac{k!}{(k+3)!}}_{\frac{1}{(k+1)(k+2)(k+3)} \leqslant \frac{1}{6}} \frac{|y|^k}{k!}$$

$$\leqslant \frac{|y|^3}{6}\mathrm{e}^{|y|} < |y|^3, \tag{B.1}$$

where the last inequality holds for $|y| = |\frac{x}{r}| < 1$, which is true due to our assumptions. Hence $\left|E(\frac{x}{r})\right| \leqslant |\frac{x}{r}|^3$ for $|x| < r$.

We then turn our attention to the powers of $W$,

$$|W| = \left|1 + \mathrm{i}\frac{x}{r} - \frac{x^2}{r^2}\right| \leqslant 1 + \frac{x}{r} + \frac{x^2}{r^2}. \tag{B.2}$$

For all $k \in \{1, 2, \ldots, r\}$ we have

$$|W|^k \leqslant \left(1 + \frac{x}{r} + \frac{x^2}{r^2}\right)^k \leqslant \mathrm{e}^{\left(\frac{x}{r} + \frac{x^2}{r^2}\right)k} = \mathrm{e}^{\frac{|x|}{r}k}\mathrm{e}^{\frac{|x|^2}{r^2}k} \leqslant \mathrm{e}^{|x|}\,\mathrm{e}^{\frac{|x|^2}{r}} \leqslant \mathrm{e}^{2x} \leqslant \mathrm{e}^{\pi} \tag{B.3}$$

where we have used the fact that $\frac{k}{r} < 1$. The second inequality is due to $(1+a)^k \leqslant \mathrm{e}^{ka}$, $a \in \mathbb{R}$, $k \in \mathbb{Z}^+$. Indeed,

$$(1+a)^k = \sum_{l=0}^{k} \binom{k}{l} a^{k-l} = \sum_{l=0}^{k} \frac{k!}{l!(k-l)!} a^{k-l} = \sum_{l=0}^{k} \frac{k!}{l!(k-l)!} \frac{(ka)^{k-l}}{k^{k-l}}$$
$$= \sum_{l=0}^{k} \underbrace{\frac{k(k-1)\cdots(l+1)}{k^{k-l}}}_{\leqslant 1} \underbrace{\frac{l\cdots 1}{l!}}_{\leqslant 1} \frac{(ka)^{k-l}}{(k-l)!} \leqslant \sum_{l=0}^{k} \frac{(ka)^{k-l}}{(k-l)!} = \sum_{l=0}^{k} \frac{(ka)^l}{l!} \leqslant \mathrm{e}^{ka}. \tag{B.4}$$

Finally we look at the approximation error. Note that

$$\mathrm{e}^{\mathrm{i}x} = \left(W + E\left(\frac{x}{r}\right)\right)^r = \sum_{k=0}^{r} \binom{r}{k} W^k \left[E\left(\frac{x}{r}\right)\right]^{r-k}$$
$$= W^r + \underbrace{\binom{r}{l} W^{r-1} E\left(\frac{x}{r}\right) + \cdots + \binom{r}{r} W^0 \left[E\left(\frac{x}{r}\right)\right]^r}_{\text{error in } r\text{-th power}}. \tag{B.5}$$

Consider the $k$th term in the error series. According to (B.1) we have

$$\binom{r}{k} |W|^{r-k} \left|E\left(\frac{x}{r}\right)\right|^k \leqslant C \binom{r}{k} \left|\frac{x}{r}\right|^{3k} = C \frac{r!}{k!(r-k)!} \frac{|x|^{3k}}{r^{3k}}$$
$$= C \frac{r(r-1)\cdots(r-k+1)}{k!} \frac{1}{r^k} \frac{|x|^{3k}}{r^{2k}}$$
$$\leqslant C \frac{|x|^k}{k!} \frac{|x|^{2k}}{r^{2k}} \leqslant \frac{\pi}{2} C \left(\frac{|x|}{r}\right)^{2k} \leqslant \frac{\pi}{2} \mathrm{e}^{\pi} \left(\frac{|x|}{r}\right)^{2k},$$

where $C = \mathrm{e}^{\pi}$ and we use Stirling's formula $k! = \sqrt{2\pi} k^{k+1/2} \exp\left(-k + \frac{\theta}{12k}\right)$, $\theta \in (0, 1)$, [44, p 257] to obtain $|x|^k/k! \leqslant 5^{-k} x^k \mathrm{e}^k \leqslant 1$ for $k \geqslant 5$, since $|x| \leqslant \frac{\pi}{2}$. So the total approximation error is bounded by

$$|\mathrm{e}^{\mathrm{i}x} - W^r| \leqslant \sum_{k=1}^{r} \binom{r}{k} |W|^{r-k} \left|\frac{x}{r}\right|^{3k} \leqslant \frac{\pi}{2} \mathrm{e}^{\pi} r \left(\frac{|x|}{r}\right)^2 \leqslant \mathrm{e}^{\pi} \left(\frac{\pi}{2}\right)^3 \frac{1}{r} \leqslant 2^7 \times \frac{1}{r}. \tag{B.6}$$

□

**Lemma B.2.** *Under the assumptions of lemma B.1*

$$|\sin x - \Im(W^r)| \leqslant 2^7/r$$

*and*

$$|\cos x - \Re(W^r)| \leqslant 2^7/r.$$

The proof is trivial and we omit it.

**Proposition B.1.** *Let $r = 2^{\nu+7}$ for $\nu \geqslant 1$ and consider the procedure computing $W^r$, as defined in lemma B.1 using repeated squaring. Assume each step computing a square carries out the*

*calculation using fixed precision arithmetic and that its inputs and outputs are s bit numbers. Let $\hat{W}_r$ be the final result. Then the error is*

$$\left|W^r - \hat{W}_r\right| \leqslant \frac{2^{\nu+9}}{2^s},$$

*for $s \geqslant 11 + \nu + \log_2 M$, where $1/M$ is the mesh size in the discretization of the Poisson equation.*

**Proof.** We are interested in estimating $\sin(j\pi/(2M))$, for $j = 1, 2, \ldots, M-1$. We consider $x \in [\pi/(2M), \pi/2)$. We approximate $e^{ix}$ and from this $\sin x$, which is the imaginary part of $e^{ix}$. Let $y = \frac{x}{r} \leqslant 2^{-7}$. We truncate it to $s$ bits of accuracy to obtain $\hat{y}$. Note that $W = 1 - y^2 + iy$ satisfies $|W|^2 = 1 - y^2 + y^4 < 1$. Let $\hat{W}_0 = 1 - \hat{y}^2 + i\hat{y}$, $y - \hat{y} \leqslant 2^{-s}$. Then $|\hat{W}_0|^2 \leqslant |W|^2 + 4y2^{-s} < 1$, for $s \geqslant 11 + \nu + \log_2 M$. This value of $s$ follows by solving

$$4y2^{-s} \leqslant y^2/2,$$

which ensures that $\hat{W}_0^2 \leqslant 1$. In addition,

$$\left|\Re\left(\hat{W}_0 - W\right)\right| \leqslant 2y2^{-s} + 2^{-2s}$$

and

$$\left|\Im\left(\hat{W}_0 - W\right)\right| \leqslant 2^{-s}.$$

Define the sequence of approximations

$$\begin{aligned}
\hat{W}_1 &= \hat{W}_0 + e_1, \\
\hat{W}_2 &= \hat{W}_1^2 + e_2, \\
&\vdots \\
\hat{W}_r &= \left(\hat{W}_{r/2}\right)^2 + e_r,
\end{aligned} \tag{B.7}$$

where $r = 2^{\nu+7}$ and the error terms $e_1, e_2, \ldots, e_r$ are complex numbers denoting that the real and imaginary parts of the results are truncated to $s$ bits of accuracy.

Observe that if $|\hat{W}_{2^{j-1}}| < 1$ then $|\hat{W}_{2^j}| < 1$, since $|\hat{W}_{2^{j-1}}|^2 < 1$ and truncation of real and imaginary parts does not increase the magnitude of a complex number. Since $|\hat{W}_0| < 1$, all the numbers in the sequence (B.7) belong to the unit disc $S$ in the complex plane.

Let $z = a + bi$. Then the function that computes $z^2$ can be understood as a vector valued function of two variables, $h : S \to S$, such that $h(a, b) = (a^2 - b^2, 2ab)$. The Jacobian of $h$ is

$$J = 2\begin{pmatrix} a & -b \\ b & a \end{pmatrix}, \quad (a, b) \in S$$

and its Euclidean norm satisfies $\|J\| \leqslant 2$, since $a^2 + b^2 \leqslant 1$. Using this bound we obtain

$$\begin{aligned}
|W^r - \hat{W}_r| &\leqslant |W^r - (\hat{W}_{r/2})^2| + |e_r| \\
&\leqslant 2\{2|W^{r/4} - \hat{W}_{r/4}| + |e_{r/4}|\} + |e_r| \\
&\leqslant 2^{\nu+7}|W - \hat{W}_1| + 2^{\nu+7-1}|e_2| + \cdots + 2^0|e_{2^{\nu+7}}| \\
&= 2^{\nu+7}\left|W - \hat{W}_0\right| + 2^{\nu+7}|e_1| + \cdots + |e_{2^{\nu+7}}|
\end{aligned}$$

$$\leqslant 2^{\nu+7}\left|W-\hat{W}_0\right|+\frac{\sqrt{2}}{2^s}\sum_{j=0}^{\nu+7}2^{\nu+7-j}$$
$$\leqslant 2^{\nu+7}\sqrt{\left(2y\frac{1}{2^s}+\frac{1}{2^{2s}}\right)^2+\frac{1}{2^{2s}}}+\frac{\sqrt{2}}{2^s}\left(2^{\nu+8}-1\right)$$
$$\leqslant 4\frac{2^{\nu+7}}{2^s}, \tag{B.8}$$

where the last inequality follows since $2y+2^{-s}\leqslant 2^{-6}+2^{-11}$. □

**Proposition B.2.** *Under the assumptions of proposition B.1 we approximate* $\sin x$ *by* $\Im(\hat{W}_r)$, $x\in[\pi/(2M),\pi/2)$, *with* $s=\max\{2\nu+9, 11+\nu+\log_2 M\}$ *bits and* $r=2^{\nu+7}$. *Then the error is*

$$|\sin x-\Im(\hat{W}_r)|\leqslant 2^{-(\nu-1)}.$$

*Moreover, we note the following.*

- *Denoting by* $\hat{W}_{r,j}$ *the approximations to* $\sin(\pi j/(2M))$, $j=1,2,\ldots,M-1$, *we have the following error bound:*

$$\left|4M^2\sin^2(j\pi/(2M))-4M^2\left(\Im(\hat{W}_{r,j})\right)^2\right|\leqslant 2^{-(\nu-4)}M^2,$$

  $j=1,2,\ldots,M-1$, *for the eigenvalues of the matrix* $h^{-2}L_h$ *that approximates the second derivative operator, using mesh size* $h=1/M$.
- *Letting* $\ell_j$ *be the truncation of* $4M^2(\Im(\hat{W}_{r,j}))^2$ *to* $\nu$ *bits after the decimal point (the length of* $\ell_j$ *is* $\nu+\log_2(4M^2)$ *bits, and* $\nu$ *is sufficiently large to satisfy the accuracy requirements) we have*

$$\left|4M^2\sin^2(j\pi/(2M))-\ell_j\right|\leqslant 17\times 2^{-\nu}M^2,$$

  *for* $j=1,2,\ldots,M-1$.

**Proof.** We have

$$\left|e^{ix}-\hat{W}_r\right|\leqslant\left|e^{ix}-W^r\right|+\left|W^r-\hat{W}_r\right|$$
$$\leqslant\frac{2^7}{2^{\nu+7}}+\frac{2^{\nu+9}}{2^s}$$
$$=2^{-\nu}+\frac{2^{\nu+9}}{2^s}=\frac{1}{2^{\nu-1}}, \tag{B.9}$$

for $s=\max\{2\nu+9, 11+\nu+\log_2 M\}$, which completes the proof of the first part. The proof of the second and third parts follows immediately. □

**Theorem B.2.** *Consider the eigenvalues*

$$\lambda_{j_1,\ldots,j_d}=4M^2\prod_{k=1}^{d}\sin^2\left(\frac{j_k\pi}{2M}\right),$$

*$j_k = 1, 2, \ldots, M-1$, $k = 1, 2, \ldots, d$ of $-\Delta_h$, $h = 1/M$. Let*

$$\hat{\lambda}_{j_1,\ldots,j_d} = \sum_{k=1}^{d} \ell_{j_k},$$

*where $\ell_{j_k}$ are defined in proposition B.2, $j_k = 1, 2, \ldots, M-1$, $k = 1, 2, \ldots, d$. Then*

$$|\lambda_{j_1,\ldots j_d} - \hat{\lambda}_{j_1,\ldots,j_d}| \leqslant \frac{17M^2 d}{2^v}.$$

The proof follows from proposition B.2 and the fact that the $d$-dimensional eigenvalues are sums of the one-dimensional eigenvalues.

## References


[1] Abrams D S and Lloyd S 1997 Simulation of many-body Fermi systems on a quantum computer *Phys. Rev. Lett.* **79** 2586–9

[2] Abrams D S and Lloyd S 1999 Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors *Phys. Rev. Lett.* **83** 5162–5

[3] Aspuru-Guzik A, Dutoi A D, Love P J and Head-Gordon M 2005 Simulated quantum computation of molecular energies *Science* **379** 1704–7

[4] Dowling J P 2006 To compute or not to compute *Nature* **439** 919

[5] Lidar D and Wang H 1999 Calculating the thermal rate constant with exponential speed-up on a quantum computer *Phys. Rev.* E **59** 2429–38

[6] Lloyd S 1996 Universal quantum simulators *Science* **273** 1073–8

[7] Papageorgiou A, Petras I, Traub J F and Zhang C A fast algorithm for approximating the ground state energy on a quantum computer *Math. Comput.* to appear

[8] Shor P W 1994 Algorithms for quantum computation: discrete logarithm and factoring *Proc. 35th Annual Symp. Foundations of Computer Science* ed S Goldwasser (New York: IEEE Computer Society Press) pp 124–34

[9] Papageorgiou A and Zhang C 2012 On the efficiency of quantum algorithms for hamiltonian simulation *Quantum Inform. Process.* **11** 541–61

[10] Wang H, Ashhab S and Nori F 2012 Quantum algorithm for obtaining the energy spectrum of a physical system *Phys. Rev.* A **85** 062304

[11] Wang H, Kais S, Aspuru-Guzik A and Hoffmann M R 2008 Quantum algorithm for obtaining the energy spectrum of molecular systems *Phys. Chem. Chem. Phys.* **10** 5388–93

[12] You J Q and Nori F 2011 Atomic physics and quantum optics using superconducting circuits *Nature* **474** 589

[13] Batchelor G K 2000 *An Introduction to Fluid Dynamics* (Cambridge: Cambridge University Press)

[14] Fletcher C A J 1991 *Computational Techniques for Fluid Dynamics* vol 1, 2nd edn (Berlin: Springer)

[15] Tomasi J, Mennucci B and Cammi R 2005 Quantum mechanical continuum solvation models *Chem. Rev.* **105** 2999–3094

[16] Griffiths D J 1999 *Introduction to Electrodynamics* (Upper Saddle River, NJ: Prentice-Hall)

[17] Meyn S P 2007 *Control Techniques for Complex Networks* (Cambridge: Cambridge University Press)

[18] Meyn S P and Tweedie R L 2009 *Markov Chains and Stochastic Stability* (Cambridge: Cambridge University Press)

[19] Asmussen S and Glynn P W 2007 *Stochastic Simulation: Algorithms and Analysis (Stochastic Modelling and Applied Probability* vol 57*)* (Berlin: Springer) pp 103–5

[20] Engel E and Dreizler R M 2011 *Density Functional Theory: An Advanced Course* (New York: Springer)

[21] Werschulz A G 1991 *The Computational Complexity of Differential and Integral Equations: An Information-Based Approach* (New York: Oxford University Press)

[22] Ritter K and Wasilkowski G W 1996 On the average case complexity of solving poisson equations (*Lectures in Applied Mathematics* vol 32) ed J Renegar, M Shub and S Smale (University Park, PA: The Pennsylvania State University) pp 677–87
[23] Grover L and Rudolph T 2002 Creating superpositions that correspond to efficiently integrable probability distributions arXiv:quant-ph/0208112v1
[24] Soklakov A N and Schack R 2006 Efficient state preparation for a register of quantum bits *Phys. Rev.* A **73** 012307
[25] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **15** 150502
[26] Berry D W 2010 Quantum algorithms for solving linear differential equations arXiv:1010.2745v1 [quant-ph]
[27] Leyton S K and Osborne T J 2008 A quantum algorithm to solve nonlinear differential equations arXiv:0812.4423 [quant-ph]
[28] Childs A M and Wiebe N 2012 Hamiltonian simulation using linear combinations of unitary operations arXiv:1202.5822 [quant-ph]
[29] Evans L C 1998 *Partial Differential Equations* (Providence, RI: American Mathematical Society)
[30] Forsythe G E and Wasow W R 2004 *Finite-Difference Methods for Partial Differential Equations* (New York: Dover)
[31] Demmel J W 1997 *Applied Numerical Linear Algebra* (Philadelphia, PA: SIAM)
[32] LeVeque R J 2007 *Finite Difference Methods for Ordinary and Partial Differential Equations* (Philadelphia, PA: SIAM)
[33] Bramble J H and Hubbard B E 1962 On the formulation of finite difference analogues of the Dirichlet problem for Poisson's equation *Numer. Math.* **4** 313–27
[34] Saad Y 2003 *Iterative Methods for Sparse Linear Systems* (Philadelphia, PA: SIAM)
[35] Traub J F and Woźniakowski H 1984 On the optimal solution of large linear systems *J. ACM* **31** 545–59
[36] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)
[37] Vedral V, Barenco A and Ekert A 1996 Quantum networks for elementary arithmetic operations *Phys. Rev.* A **54** 147–53
[38] Klappenecker A and Roetteler M 2001 Discrete cosine transforms on quantum computers arXiv:quant-ph/0111038
[39] Pueschel M, Roetteler M and Beth T 1998 Fast quantum fourier transforms for a class of non-Abelian groups arXiv:quant-ph/9807064v1
[40] Wickerhauser M V 1994 *Adapted Wavelet Analysis from Theory to Software* (Wellesley, MA: A K Peters)
[41] Berry D W, Ahokas G, Cleve R and Sanders B S 2007 Efficient quantum algorithms for simulating sparse Hamiltonians *Commun. Math. Phys.* **270** 359–71
[42] di Benedetto F 1997 Preconditioning of block Toeplitz matrices by sine transforms *SIAM J. Sci. Comput.* **18** 499–515
[43] Jordan S P 2005 Fast quantum algorithm for numerical gradient estimation *Phys. Rev. Lett.* **95** 050501
[44] Abramowitz M and Stegun I A 1972 *Handbook of Mathematical Functions* (New York: Dover)