Quantum algorithms for solving linear differential equations

Dominic W. Berry[1]

[1]*Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*

Linear differential equations are ubiquitous in science and engineering. Quantum computers can simulate quantum systems, which are described by homogeneous linear differential equations that produce only oscillating terms. Here we extend quantum simulation algorithms to general inhomogeneous linear differential equations, which can include exponential terms as well as oscillating terms in their solution. As with other algorithms of this type, the solution is encoded in amplitudes of the quantum state. The algorithm does not give the explicit solution, but it is possible to extract global features of the solution.

I. INTRODUCTION

Differential equations are used for an enormous variety of applications, including industrial design and weather prediction. In fact, many of the main applications of supercomputers are in the form of large systems of differential equations [1]. Therefore quantum algorithms for solving differential equations would be extraordinarily valuable. A quantum algorithm for differential equations was proposed in Ref. [2], but that algorithm had very poor scaling in the time. The complexity of the simulation scaled exponentially in the number of time-steps over which to perform the simulation.

The algorithm in Ref. [2] may have been overly ambitious, because it aimed to solve nonlinear differential equations. A more natural application for quantum computers is *linear* differential equations. This is because quantum mechanics is described by linear differential equations. We find that, when we restrict to linear differential equations, it is possible to obtain an algorithm that is far more efficient than that proposed in Ref. [2].

We consider first-order linear differential equations. Using standard techniques, any linear differential equation with higher-order derivatives can be converted to a first-order linear differential equation with larger dimension. A first-order ordinary differential equation may be written as

$$\dot{x}(t) = A(t)x(t) + b(t), \tag{1}$$

where x and b are N_x-component vectors, and A is an $N_x \times N_x$ matrix. Classically, the complexity of solving the differential equation must be at least linear in N_x. The goal of the quantum algorithm is to solve the differential equation in time $O(\text{poly}\log N_x)$.

Quantum mechanics is described by differential equations of this form, except they are homogeneous ($b(t) = 0$), and $A(t) = iH(t)$, where $H(t)$ is Hermitian. This means that the solutions in quantum mechanics only include oscillating terms, whereas more general differential equations have solutions that may grow or decay exponentially. Quantum algorithms for simulating quantum mechanical systems have been extensively studied [3–9].

Classical physics is described by more general differential equations. Large systems of ordinary differential equations are produced by discretisation of partial differential equations. Many equations in physics are linear partial differential equations, where the time derivative depends linearly on spatial derivatives and the value of a quantity at some point in physical space. Examples include Stokes equations (for creeping fluid flow), the heat equation, and Maxwell's equations. Discretisation of the partial differential equation on a mesh of points results in an ordinary differential equation with a very large value of N_x.

In the case where A and b are time independent, then one can find the equilibrium solution of the differential equation by solving

$$Ax = -b. \tag{2}$$

A quantum algorithm for this problem was given by Harrow, Hassadim and Lloyd [10], with runtime that is polynomial in $\log(N_x)$ and the condition number of A. Ambainis has reported development of an improved algorithm [11], though this algorithm has not yet been released. We consider the more difficult case of solving the time evolution under linear differential equations, rather than just the equilibrium solution. We find that this case can also be solved using a modification of the method of Harrow, Hassadim and Lloyd.

II. TROTTER FORMULA APPROACH

Before explaining that approach, we first describe an approach using Trotter formulae, and the drawback to that approach. This will not be described rigorously, because it is not our main proposal for solving differential equations.

The homogeneous case, where $b = 0$, is analogous to Hamiltonian evolution. If A is antiHermitian, then we can take $A = iH$, where H is a Hermitian Hamiltonian. Evolution under this Hamiltonian can be solved by methods considered in previous work [6, 8]. Another case that can be considered is where A is Hermitian. In this case, the eigenvalues of A are real, and A can be diagonalised in the form $A = VDV^{-1}$, where D is a real diagonal matrix and V is unitary. The formal solution is then, for A independent of time, $x(t) = Ve^{D(t-t_0)}V^{-1}x(t_0)$.

The differential equation can be solved using a similar method to that used in Ref. [10]. The value of x is encoded in a quantum state as

$$|x\rangle = \mathcal{N}_x \sum_{j=1}^{N_x} x^{[j]}|j\rangle, \tag{3}$$

where $|j\rangle$ are computational basis states of the quantum computer, $x^{[j]}$ are the components of the vector x, and $\mathcal{N}_x$ is a normalisation constant. The state can be written in a basis corresponding to the eigenvectors of A:

$$|x\rangle = \sum_j \lambda_j |\lambda_j\rangle. \tag{4}$$

Using methods for Hamiltonian simulation, iA can be simulated. By using phase estimation, if the state is an eigenstate $|\lambda_j\rangle$, then the eigenvalue λ_j can be determined. Given maximum eigenvalue $\lambda_{\max}$, we would change the amplitude by a factor of $e^{(t-t_0)(\lambda_j-\lambda_{\max})}$. See Ref. [10] for the method of changing the amplitude. If this is done coherently, then the final state will encode $x(t)$.

For more general differential equations, A will be neither Hermitian nor antiHermitian. In this case, one can break A up into Hermitian (A_H) and antiHermitian (A_{aH}) components. The evolution under each of these components can be simulated individually, and the overall evolution simulated by combining these evolutions via the Trotter formula. The drawback to this approach is that it appears to give a complexity that increases exponentially with the time interval $\Delta t = t - t_0$ (though the complexity is still greatly improved over Ref. [2]).

If A were just Hermitian, then the eigenvector (or eigenspace) corresponding to the largest eigenvalue would not decay, and the system would end up in that state. Therefore the amplitude would not drop below the amplitude on the eigenspace corresponding to the largest eigenvalue. That is not the case when A is a more general matrix, because usually the maximum real part of an eigenvalue of A will be strictly less than the maximum eigenvalue of A_H. The amplitude must therefore decay exponentially, because we must use the maximum eigenvalue of A_H in simulating evolution under A_H.

The result of this is that the complexity of the simulation will scale exponentially in the time that the differential equation needs to be simulated over, Δt. The scaling will be considerably improved over that in Ref. [2], but it is desirable to obtain scaling that is polynomial in Δt. Another drawback is that this approach does not enable simulation of inhomogeneous differential equations.

III. LINEAR SYSTEMS APPROACH

To avoid this problem we propose an approach based on the algorithm for solving linear systems from Ref. [10]. The trick is to encode the solution of the differential equation at different times using the one state. That is, we wish to obtain the final state proportional to

$$|\psi\rangle := \sum_{j=0}^{N_t} |t_j\rangle|x_j\rangle. \tag{5}$$

The number N_t is the number of time steps, t_j is the time $t_0 + jh$, where h is the time interval in the discretisation of the differential equation, x_j is the approximation of the value of x at time t_j, and Δt is the total time interval over which the differential equation is to be solved. We use the subscript j to index the vectors, and superscript for components of these vectors.

Once this state has been created, the state encoding the solution at the final time $t_0 + \Delta t$ can be approximated by measuring the register encoding the time and getting that time. Just using this method, the probability of obtaining the final time is small ($1/(N_t + 1)$). To obtain a significant probability of success, one can add times beyond $t_0 + \Delta t$ where x is constant. We take x to be constant for $t_0 + \Delta t$ to $t_0 + 2\Delta t$, so $N_t = 2\Delta t/h$. Then any measurement result for the time in this interval will give the state corresponding to the solution. By this method, the probability of success can be boosted significantly, without changing the scaling for N_t.

To numerically solve differential equations, the simplest method is the Euler method, which discretises the differential equation as

$$\frac{x_{j+1} - x_j}{h} = A(t_j)x_j + b(t_j). \tag{6}$$

For times after $t_0 + \Delta t$, we set $x_{j+1} = x_j$ to ensure that x is constant. The Euler method yields an error that scales as $O(h^2)$ for a single time step. Therefore, we expect that the error in the total simulation is $O(N_t h^2) = O(\Delta t^2/N_t)$. To achieve error bounded by ϵ, we can take $N_t = O(\Delta t^2/\epsilon)$. To show these scalings rigorously requires additional constraints on the problem.

In particular, to rigorously bound the error it is necessary that the eigenvalues of $A(t_j)$ have no positive real part. Otherwise the error can grow exponentially. In cases where $A(t_j)$ does have an eigenvalue with positive real part, one can simply subtract a multiple of the identity, and rescale the solution. Note that ϵ is the error in the solution of the differential equation, and is distinct from error in the solution of linear systems.

More generally, linear multistep methods have the form [12, 13]

$$\sum_{\ell=0}^{k} \alpha_\ell x_{j+\ell} = h \sum_{\ell=0}^{k} \beta_\ell [A(t_{j+\ell})x_{j+\ell} + b(t_{j+\ell})]. \tag{7}$$

Multistep methods can be chosen such that the error is of higher order in h, but there is the problem that the method may not be stable. That is, even if the exact solution of the differential equation is bounded, the solution of the difference equation may be unbounded.

To examine the stability, one defines the generating polynomials

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j, \qquad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j. \tag{8}$$

The stability can be examined via the roots of the equation

$$\rho(\zeta) - \mu\sigma(\zeta) = 0. \tag{9}$$

One defines the set S by

$$S := \left\{ \mu \in \mathbb{C};\ \begin{array}{l} \text{all roots } \zeta_j(\mu) \text{ of (9) satisfy } |\zeta_j(\mu)| \le 1 \\ \text{multiple roots satisfy } |\zeta_j(\mu)| < 1 \end{array} \right\}. \tag{10}$$

S is called the stability domain or stability region of the multistep method. In addition, if the roots of $\sigma(\zeta)$ all satisfy $|\zeta| \le 1$, and repeated roots satisfy $|\zeta| < 1$, then the method is said to be stable at infinity.

A linear multistep method is said to be order p if it introduces local errors $O(h^{p+1})$. This means that, if it is applied with exact starting values to the problem $\dot{x} = t^q$ $(0 \le q \le p)$, it integrates the problem without error. A linear multistep method has order p if and only if [12]

$$\rho(e^h) - h\sigma(e^h) = O(h^{p+1}). \tag{11}$$

A useful property of linear multistep methods is for them to be A-stable [13, 14].

Definition 1. *A linear multistep method is called A-stable if $S \supset \mathbb{C}^-$, i.e., if*

$$\operatorname{Re}\lambda \le 0 \implies \textit{numerical solution for } \dot{x} = \lambda x \textit{ is bounded.} \tag{12}$$

This definition means that, if the solution of the differential equation is bounded, then the approximation given by the multistep method is bounded as well. For a scalar differential equation, the multistep method is bounded whenever λ is in the left half of the complex plane. The Euler method is A-stable, but it is not possible to construct arbitrary order A-stable multistep methods. The second Dahlquist barrier is that an A-stable multistep method must be of order $p \le 2$ [13, 14]. As we wish to consider higher-order multistep methods, we relax the condition and require that the linear multistep method is $A(\alpha)$-stable [13, 15].

Definition 2. *A linear multistep method is $A(\alpha)$-stable, $0 < \alpha < \pi/2$, if*

$$S \supset S_\alpha = \{\mu; |\arg(-\mu)| < \alpha, \mu \neq 0\}. \tag{13}$$

This definition means that, in the case of a scalar differential equation, the multistep method is bounded whenever λ is within a wedge in the left half of the complex plane. For a vector differential equation, the eigenvalues of A should be within this wedge. It is known that, for any $\alpha < \pi/2$ and $k \in \mathbb{N}$, there is an $A(\alpha)$-stable linear k-step method of order $p = k$ [12, 16].

The error in the total solution of the differential equation will be $O(N_t(\Delta t)^{p+1})$. In order to obtain a rigorous result, we specialise to the case that A and b are independent of time. The relevant bound is given in Theorem 7.6 in Chapter V of Ref. [13].

Theorem 3. *Suppose a linear multistep method is of order p, $A(\alpha)$-stable and stable at infinity. If the matrix A is diagonalisable (i.e. there exists a matrix V such that $V^{-1}AV = D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$) with eigenvalues satisfying*

$$|\arg(-\lambda_i)| \leq \alpha \qquad for\ i = 1, \ldots, N_x, \tag{14}$$

then there exists a constant M (depending only on the method) such that for all $h > 0$ the global error satisfies

$$\|x(t_m) - x_m\| \leq M\kappa_V \left(\max_{0 \leq j < k} \|x(t_j) - x_j\| + h^p \int_{t_0}^{t_m} \|x^{(p+1)}(\xi)\| d\xi \right), \tag{15}$$

where $\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V.

Here the superscript with round brackets denotes repeated derivative. We can use this result to show a lemma on the scaling of the error.

Lemma 4. *Suppose a linear multistep method is of order p, $A(\alpha)$-stable and stable at infinity. If the matrix A is diagonalisable (i.e. there exists a matrix V such that $V^{-1}AV = D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$) with eigenvalues satisfying*

$$|\arg(-\lambda_i)| \leq \alpha \qquad for\ i = 1, \ldots, N_x, \tag{16}$$

and b is constant, then the global error satisfies

$$\|x(t_m) - x_m\| = O\left(\kappa_V^2 (\|x_{\mathrm{init}}\| + \|b\|/\|A\|) \left[\kappa_V (h\|A\|)^2 + m(h\|A\|)^{p+1} \right] \right), \tag{17}$$

where $\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V.

Proof. The linear multistep method requires a starting method to obtain the values of x_j for $0 < j < k$. The term $\max_{0 \leq j < k} \|x(t_j) - x_j\|$ arises from the inaccuracy in this procedure. One can simply use the Euler method, in which case the error is $O(h^2)$. It is also possible to use higher-order starting methods, but there is not a convenient rigorous result that can be used. To determine the error in the Euler method, one can simply use Theorem 3. Because $k = 1$ and $p = 1$ for the Euler method, and for the initial point there is zero error ($x(t_0) = x_0$), Theorem 3 gives

$$\|x(t_j) - x_j\| \leq M_E \kappa_V h \int_{t_0}^{t_j} \|x^{(2)}(\xi)\| d\xi, \tag{18}$$

where M_E is the constant for the Euler method. We consider this expression for $0 \leq j < k$, and obtain

$$\max_{0 \leq j < k} \|x(t_j) - x_j\| \leq M_E \kappa_V h^2 (k-1) \max_{\xi \in [t_0, t_0 + (k-1)h]} \|x^{(2)}(\xi)\|. \tag{19}$$

In using these results, it is necessary to place upper bounds on the values of $\|x^{(p+1)}(\xi)\|$ and $\|x^{(2)}(\xi)\|$. In general these will depend on the value of $b(t)$, and its time-dependence. It is well-behaved if b is a constant, in which case the exact solution is

$$x(t) = e^{A(t-t_0)}(x_{\mathrm{init}} + A^{-1}b) - A^{-1}b. \tag{20}$$

Then

$$x^{(\ell)}(t) = e^{A(t-t_0)}(A^\ell x_{\mathrm{init}} + A^{\ell-1}b), \tag{21}$$

so

$$\begin{aligned} \|x^{(\ell)}(t)\| &= \|V e^{D(t-t_0)} V^{-1}(A^\ell x_{\mathrm{init}} + A^{\ell-1}b)\| \\ &\leq \kappa_V (\|A\|^\ell \|x_{\mathrm{init}}\| + \|A\|^{\ell-1}\|b\|) \end{aligned} \tag{22}$$

In the first line we have used the diagonalisation of A, and in the second line we have used the condition that $|\arg(-\lambda_i)| \leq \alpha$. Using Theorem 3, the error is bounded as

$$\begin{aligned}\|x(t_m) - x_m\| &\leq M\kappa_V \left(\max_{0\leq j<k} \|x(t_j) - x_j\| + h^p \int_{t_0}^{t_m} \|x^{(p+1)}(\xi)\| d\xi \right) \\ &\leq M\kappa_V \left[M_E \kappa_V^2 h^2 k (\|x_{\text{init}}\| + \|b\|/\|A\|)\|A\|^2 + h^p (t_m - t_0)\kappa_V (\|x_{\text{init}}\| + \|b\|/\|A\|)\|A\|^{p+1} \right] \\ &= O\left(\kappa_V^2 (\|x_{\text{init}}\| + \|b\|/\|A\|) \left[\kappa_V (h\|A\|)^2 + m(h\|A\|)^{p+1} \right] \right). \end{aligned} \tag{23}$$

□

This result means that, disregarding the dependence on many of the quantities, and omitting the error due to the starting method, the error scales as $O((\|A\|h)^p \|A\| \Delta t)$ for total time Δt. To achieve error bounded by ϵ, we then use

$$N_t = O\left(\frac{(\|A\|\Delta t)^{1+1/p}}{\epsilon^{1/p}} \right). \tag{24}$$

That is, the number of time steps required is close to linear in the time.

Given a linear multistep method, it is straightforward to encode this method as a linear system

$$\mathcal{A}\vec{x} = \vec{b}. \tag{25}$$

Here $\vec{x}$ is the vector of blocks x_j, $\vec{b}$ is a vector of the blocks b, and $\mathcal{A}$ is a matrix describing the initial condition and discretised differential equation. As an example, the equations for the Euler method and A and b independent of time may be expressed as

$$\begin{bmatrix} \mathbb{1} & 0 & 0 & 0 & 0 \\ -(\mathbb{1} + Ah) & \mathbb{1} & 0 & 0 & 0 \\ 0 & -(\mathbb{1} + Ah) & \mathbb{1} & 0 & 0 \\ 0 & 0 & -\mathbb{1} & \mathbb{1} & 0 \\ 0 & 0 & 0 & -\mathbb{1} & \mathbb{1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_{\text{in}} \\ bh \\ bh \\ 0 \\ 0 \end{bmatrix}. \tag{26}$$

Each entry of $\mathcal{A}$ is a block of the dimension of A, and each entry of $\vec{x}$ and $\vec{b}$ is a block of the dimension of x. The first row sets the initial value, $x_0 = x_{\text{in}}$. The next rows give $x_{j+1} - (x_j + Ax_j h) = bh$, corresponding to the discretisation of the differential equation via the Euler method. The final rows indicate equations where $x_{j+1} - x_j = 0$. This is for the times where x is required to be constant. More generally, we use the Euler method as a starting method, then continue with a higher-order method, then for the final rows again have $x_{j+1} - x_j = 0$ to ensure that all the final values are equal. Therefore, we set the blocks of $\mathcal{A}$ as, for $N_t \geq 2k$,

$$\begin{aligned} &\mathcal{A}_{j,j} = \mathbb{1}, && 0 \leq j < k, \quad N_t/2 < j \leq N_t, \\ &\mathcal{A}_{j,j-1} = -(\mathbb{1} + Ah), && 1 \leq j < k, \\ &\mathcal{A}_{j,j-k+\ell} = \alpha_l \mathbb{1} - \beta_\ell Ah, && k \leq j \leq N_t/2, \quad 0 \leq \ell \leq k. \\ &\mathcal{A}_{j,j-1} = -\mathbb{1}, && N_t/2 < j \leq N_t. \end{aligned} \tag{27}$$

We will always require $N_t \geq 2k$ when using $\mathcal{A}$, because otherwise there are not enough time steps to start the linear multistep method. We also set the blocks of $\vec{b}$ as

$$\begin{aligned} &b_0 = x_{\text{in}}, \\ &b_j = bh, && 1 \leq j < k, \\ &b_j = \textstyle\sum_{\ell=0}^{k} \beta_\ell bh, && k \leq j \leq N_t/2, \\ &b_j = 0, && N_t/2 < j \leq N_t. \end{aligned} \tag{28}$$

We require A, b, and x_{in} to be sparse, with no more than s nonzero elements in any row or column. We assume that the oracles are of the same form as in Ref. [8]. That is, the oracle for A is a unitary operator acting as

$$O_A |j, \ell\rangle |z\rangle = |j, \ell\rangle |z \oplus A^{[j,\ell]}\rangle. \tag{29}$$

Note that we use superscript to denote indexing within the block A. We also require an oracle for the sparseness, that locates the nonzero elements. Given a function $f(j, \ell)$ that gives the row index of the ℓth nonzero element in column j, we require a unitary oracle

$$O_F |j, \ell\rangle = |j, f(j, \ell)\rangle. \tag{30}$$

Because A is not Hermitian, we require a similar oracle to give the positions of the nonzero elements in a given row. We also require oracles to give the values and locations of non-zero elements for b and $x_{\rm in}$. These oracles ensure that the initial state corresponding to $\vec{b}$ can be prepared efficiently. Alternatively, it is also possible to consider b and $x_{\rm in}$ such the efficient preparation procedure of Ref. [17] can be used.

A linear system of equations can be solved using the algorithm of Ref. [10] with complexity $\tilde{O}(\log(N)s^4\kappa^2/\epsilon_L)$, where κ is the condition number of the matrix $\mathcal{A}$, and ϵ_L is the allowable error. (Note that the power of s should be 4, not 2 as given in Ref. [10].) We use the symbol ϵ_L to indicate the allowable error for the solution of the linear systems, which is distinct from the allowable error for the solution of the differential equation. The scaling can be improved if the method reported in Ref. [11] is used (the method is not given there, but is reported as being in a manuscript in preparation). The scaling reported there is $O(\kappa^{1+o(1)}\log^c N)$, which does not include the scaling in s or ϵ_L.

IV. BOUNDING THE CONDITION NUMBER

To determine the complexity in either case it is necessary to determine the value of the condition number κ. To bound this condition number we first determine bounds on the norms of $\|\mathcal{A}\|$ and $\|\mathcal{A}^{-1}\|$.

Lemma 5. *The matrix $\mathcal{A}$, with blocks given by Eq. (27), satisfies $\|\mathcal{A}\| = O(1)$ provided $h = O(1/\|A\|)$.*

Proof. To determine the upper bound on $\|\mathcal{A}\|$, we express $\mathcal{A}$ as a sum of block-diagonal matrices, and use the triangle inequality. Let us define $\mathcal{A}^{\{\ell\}}$ to be the block diagonal matrix with all entries zero, except

$$\mathcal{A}^{\{\ell\}}_{j,j-\ell} = \mathcal{A}_{j,j-\ell}. \tag{31}$$

We then have

$$\mathcal{A} = \sum_{\ell=0}^{k} \mathcal{A}^{\{\ell\}}, \tag{32}$$

so, via the triangle inequality,

$$\|\mathcal{A}\| \leq \sum_{\ell=0}^{k} \|\mathcal{A}^{\{\ell\}}\|. \tag{33}$$

The norm of a block-diagonal matrix is just the maximum norm of the blocks, so we find

$$\begin{aligned} \|\mathcal{A}^{\{0\}}\| &\leq \max(1, |\alpha_k| + |\beta_k| h \|A\|), \\ \|\mathcal{A}^{\{1\}}\| &\leq \max(1 + h\|A\|, |\alpha_{k-1}| + |\beta_{k-1}| h \|A\|), \\ \|\mathcal{A}^{\{\ell\}}\| &\leq |\alpha_\ell| + |\beta_\ell| h \|A\|, \qquad 1 < \ell \leq k. \end{aligned} \tag{34}$$

Because we require that $h = O(1/\|A\|)$, each of these norms is $O(1)$, and hence the overall norm is $O(1)$. □

Lemma 6. *Suppose that the multistep method is $A(\alpha)$-stable, the matrix A may be diagonalised as $A = VDV^{-1}$, and the eigenvalues of A all satisfy $|\arg(-\lambda_i)| \leq \alpha$. Then the matrix $\mathcal{A}$, with blocks given by Eq. (27), satisfies $\|\mathcal{A}^{-1}\| = O(N_t \kappa_V)$, where κ_V is the condition number of V.*

Proof. To upper bound $\|\mathcal{A}^{-1}\|$, we use a method analogous to that used to bound the error in Ref. [13]. As in the condition for Theorem 3, we assume that A may be diagonalised as

$$A = VDV^{-1}. \tag{35}$$

Note that A need not be Hermitian, so V need not be unitary. If we define $\mathcal{V}$ to be to the block matrix with V on the diagonal, and $\mathcal{D}$ to be the matrix corresponding to $\mathcal{A}$ except with A replaced with D, then $\mathcal{A} = \mathcal{V}\mathcal{D}\mathcal{V}^{-1}$. We obtain

$$\|\mathcal{A}^{-1}\| \leq \|\mathcal{V}\| \cdot \|\mathcal{D}^{-1}\| \cdot \|\mathcal{V}^{-1}\| = \kappa_V \|\mathcal{D}^{-1}\|. \tag{36}$$

To bound $\|\mathcal{A}^{-1}\|$ we therefore just need to bound $\|\mathcal{D}^{-1}\|$.

The matrix $\mathcal{D}$ corresponds to the linear multistep solution of decoupled scalar differential equations. That is, taking $z = V^{-1}x$, the differential equation becomes N_x decoupled differential equations

$$\dot{z}^{[j]}(t) = \lambda_j z^{[j]}(t) + [V^{-1}b]^{[j]}. \tag{37}$$

The matrix $\mathcal{D}$ gives decoupled linear multistep solutions of each of these differential equations. It may therefore be written in block-diagonal form, with each block corresponding to solution of each of these decoupled equations. The value of $\|\mathcal{D}^{-1}\|$ can therefore be bounded by the maximum of the norm of the inverse of each of these blocks.

To bound the norm of the inverse, we can take $\mathcal{D}\vec{z} = \vec{y}$, and determine a bound on the norm of $\vec{z}$ for a given norm of $\vec{y}$. We can determine this by separately examining the uncoupled blocks in $\mathcal{D}$. For each of these blocks (labelled by j) we have the linear multistep equation, for $m = 0, \ldots, N_t/2 - k$,

$$\sum_{i=0}^{k} (\alpha_i - h\lambda_j\beta_i) z^{[j]}_{m+i} = y^{[j]}_{m+k}. \tag{38}$$

We also have, for the initial condition, $z^{[j]}_0 = y^{[j]}_0$, and for the Euler method as the starting method with $0 \le m < k-1$,

$$z^{[j]}_{m+1} - (1 + h\lambda_j) z^{[j]}_m = y^{[j]}_{m+1}. \tag{39}$$

For the end of the simulation, we have for $N_t/2 \le m < N_t$,

$$z^{[j]}_{m+1} - z^{[j]}_m = y^{[j]}_{m+1}. \tag{40}$$

We can see that Eq. (38) is equivalent to Eq. (7.11) in the method used to bound error in Ref. [13]. We identify $z^{[j]}_{m+i}$ as equivalent to e_{m+i} in Ref. [13], and $y^{[j]}_{m+k}$ as equivalent to $\delta_h(x_m)$ in Ref. [13]. As in that method, we can define

$$\begin{aligned} E_m &:= (z^{[j]}_{m+k-1}, \ldots, z^{[j]}_{m+1}, z^{[j]}_m)^T, \\ \Delta_m &:= (y^{[j]}_{m+k}/(\alpha_k - h\lambda_j\beta_k), 0, \ldots, 0)^T. \end{aligned} \tag{41}$$

As the problem is equivalent to that considered in Ref. [13], the result given in Eq. (7.24) of that reference holds:

$$\|E_{m+1}\| \le M \left(\|E_0\| + \sum_{\ell=0}^{m} \|\Delta_\ell\| \right), \tag{42}$$

where M is a constant depending only on the method. Using the definition of Δ_ℓ gives

$$\begin{aligned} \|E_{m+1}\| &\le M \left(\|E_0\| + \sum_{\ell=0}^{m} |y^{[j]}_{\ell+k}|/|\alpha_k - h\lambda_j\beta_k| \right) \\ &\le M \left(\|E_0\| + \sum_{\ell=0}^{m} |y^{[j]}_{\ell+k}|/|\alpha_k| \right). \end{aligned} \tag{43}$$

In the last line we have used the fact that the condition of $A(\alpha)$ stability means $\alpha_k \cdot \beta_k > 0$, so $|\alpha_k - h\lambda_j\beta_k|^{-1} \le |\alpha_k|^{-1}$.

For the starting method, we have used the Euler method, and the result is simpler. For the Euler method, E_m and Δ_m are scalars, and are just $z^{[j]}_m$ and $y^{[j]}_{m+1}$. The corresponding result is therefore, for $0 < m < k$,

$$\begin{aligned} |z^{[j]}_m| &\le M_E \left(|z^{[j]}_0| + \sum_{\ell=0}^{m-1} |y^{[j]}_{\ell+1}| \right) \\ &= M_E \sum_{\ell=0}^{m} |y^{[j]}_\ell|. \end{aligned} \tag{44}$$

Here M_E is the corresponding constant for the Euler method. For the end of the simulation, we have for $N_t/2 \le m < N_t$, $z^{[j]}_{m+1} - z^{[j]}_m = y^{[j]}_{m+1}$, so

$$|z^{[j]}_m| \le |z^{[j]}_{N_t/2}| + \sum_{\ell=N_t/2+1}^{m} |y^{[j]}_\ell|. \tag{45}$$

Now we can bound the norm of E_0 as

$$\begin{aligned}\|E_0\| &\leq \sum_{m=0}^{k-1} |z_m^{[j]}| \\ &\leq M_E k \sum_{\ell=0}^{k-1} |y_\ell^{[j]}|. \end{aligned} \tag{46}$$

We can use this result to bound $|z_m^{[j]}|$ as, for $k \leq m \leq N_t/2$,

$$\begin{aligned}|z_m^{[j]}| &\leq \|E_{m+1-k}\| \\ &\leq M \left(\|E_0\| + \sum_{\ell=0}^{m-k} |y_{\ell+k}^{[j]}|/|\alpha_k| \right) \\ &\leq M \left(M_E k \sum_{\ell=0}^{k-1} |y_\ell^{[j]}| + \sum_{\ell=k}^{m} |y_\ell^{[j]}|/|\alpha_k| \right). \end{aligned} \tag{47}$$

For convenience we define the quantity

$$M_T := \max(M M_E k, M/|\alpha_k|, M_E, 1). \tag{48}$$

Then we find that, for all $0 \leq m \leq N_t$,

$$|z_m^{[j]}| \leq M_T \sum_{\ell=0}^{m} |y_\ell^{[j]}|. \tag{49}$$

Hence we can determine an overall upper bound on the norm of $z^{[j]}$ as

$$\begin{aligned}\|z^{[j]}\|^2 &\leq M_T^2 \sum_{m=0}^{N_t} \left(\sum_{\ell=0}^{m} |y_\ell^{[j]}| \right)^2 \\ &\leq M_T^2 N_t^2 \|y^{[j]}\|^2. \end{aligned} \tag{50}$$

Summing over j, this then gives

$$\|\vec{z}\| \leq M_T N_t \|\vec{y}\|. \tag{51}$$

This result means that $\|\mathcal{D}^{-1}\| \leq M_T N_t$, where M_T depends only on the method. This then bounds the norm of $\mathcal{A}^{-1}$ as

$$\|\mathcal{A}^{-1}\| = O(N_t \kappa_V). \tag{52}$$

□

We can now use these results to bound the condition number of $\mathcal{A}$.

Theorem 7. *Suppose that the multistep method is $A(\alpha)$-stable, the matrix A may be diagonalised as $A = VDV^{-1}$, the eigenvalues of A all satisfy $|\arg(-\lambda_i)| \leq \alpha$, and $h = O(1/\|A\|)$. Then the matrix $\mathcal{A}$, with blocks given by Eq. (27), has condition number $\kappa = O(N_t \kappa_V)$, where κ_V is the condition number of V.*

Proof. The condition number is given by the formula

$$\kappa = \left(\max_{\vec{x}} \frac{\|\mathcal{A}\vec{x}\|}{\|\vec{x}\|} \right) \left(\max_{\vec{x}} \frac{\|\vec{x}\|}{\|\mathcal{A}\vec{x}\|} \right) = \|\mathcal{A}\| \cdot \|\mathcal{A}^{-1}\|. \tag{53}$$

The conditions of this theorem ensure that the conditions of Lemmas 5 and 6 hold. Therefore we can use the bounds on $\|\mathcal{A}\|$ and $\|\mathcal{A}^{-1}\|$ from those lemmas to obtain $\kappa = O(N_t \kappa_V)$. □

This result for the condition number can be explained in a more intuitive way. Each value of $y_i^{[j]}$ is equivalent to an excitation of the differential equation at a single time. Therefore $\vec{z}$ is close to the solution of the differential equation with each of those individual excitations. An excitation can not cause a growing solution, because of the stability condition. This means that the worst that an excitation can do is cause a solution that is displaced by a proportional amount for the remaining time. Therefore the norm of $\vec{z}$ can not be more than a factor of N_t times the norm of $\vec{y}$.

V. ALGORITHM FOR SOLVING LINEAR SYSTEMS

We can use the bound on the condition number to estimate the complexity of the quantum algorithm. We will first explain the scaling in a simple way, then give a more rigorous result. Using Eq. (24) for the number of time steps, we have (ignoring dependence on many of the quantities)

$$\kappa = O\left(\frac{(\|A\|\Delta t)^{1+1/p}}{\epsilon^{1/p}}\right). \tag{54}$$

Using this expression in the result for the complexity of solving linear systems from Ref. [10] gives

$$\tilde{O}(\log(N)s^4(\|A\|\Delta t)^{2+2/p}/(\epsilon^{2/p}\epsilon_L)). \tag{55}$$

If the technique of Ref. [11] can be used, then the scaling can be improved to

$$O(\log^c(N)(\|A\|\Delta t)^{1+1/p}/\epsilon^{1/p}). \tag{56}$$

There is a question of whether the scaling in Ref. [10] can be improved because we consider a specific application of the solution of linear systems. The scaling obtained in Ref. [10] is based on a worst-case scenario, that $\|\vec{b}\|$ scales as $\Lambda_{\max}\|\vec{x}\|$. It is easily shown that $\|\vec{x}\|$ scales as $\sqrt{N_t}$, provided the magnitude of the solution of the differential equation does not vary greatly in time. In contrast, the magnitude of $\vec{b}$ is given by

$$\begin{aligned}\|\vec{b}\|^2 &= \|x_{\rm in}\|^2 + (k-1)\|b\|^2h^2 + (N_t/2-k+1)\|b\|^2h^2\left(\sum_{\ell=0}^{k}\beta_\ell\right)^2 \\ &\leq \|x_{\rm in}\|^2 + (k-1)\|b\|^2h^2 + h\Delta t\|b\|^2\left(\sum_{\ell=0}^{k}\beta_\ell\right)^2.\end{aligned} \tag{57}$$

In the case that we use the Euler method, then $h \propto 1/\Delta t$, so $\|\vec{b}\| = O(1)$. In that case it is possible to solve the linear systems more efficiently than the worst-case scaling given by Ref. [10]. However, here we are concerned with using higher-order linear multistep methods. For these methods the scaling of $h\Delta t$ is close to N_t. In that case $\|\vec{b}\|$ has similar scaling to $\Lambda_{\max}\|\vec{x}\|$, so there is not a significant advantage to using this approach to improve on the result in [10]. Therefore we will just use the result stated in Ref. [10].

Before obtaining the overall scaling, another factor that needs to be considered is the scaling for creating the state encoding $\vec{b}$. This is because the algorithm of Ref. [10] uses an amplitude amplification approach, which requires the preparation of this state at each step. Therefore, the overall complexity is multiplied by the complexity of performing this state preparation. We assume that we have oracles that give the elements of $x_{\rm in}$ and $\vec{b}$ in the computational basis, and that these vectors are s-sparse. We find the following result.

Lemma 8. *The state encoding $\vec{b}$,*

$$|\vec{b}\rangle \propto \sum_{j,\ell}\vec{b}_j^{[\ell]}|j,\ell\rangle, \tag{58}$$

can be prepared using $O(\sqrt{s}+\log(N_t))$ calls to the oracles for $x_{\rm in}$ and b, provided the normalisations of $x_{\rm in}$ and b are known.

Proof. For this preparation we can assume that the normalisations of $x_{\rm in}$ and b are known. That is because this normalisation can be determined with complexity $O(s)$, and need only be determined once. Because the overall complexity of the simulation is greater than this, it can just be assumed that the normalisation is known.

A state of dimension s can be prepared with complexity $O(\sqrt{s})$ using the method of Ref. [18]. As discussed above, we assume an oracle of the form used in Ref. [8] for the sparseness. This means that it only requires one oracle call to prepare an s-sparse state from a dimension s state with the same coefficients [8]. Therefore the complexity of preparing an s-sparse state is also $O(\sqrt{s})$.

Let us encode the state in three registers (which may themselves be composed of multiple qubits). The first is one qubit encoding $|0\rangle$ for the times $t_1,\ldots,t_{N_t/2}$, and $|1\rangle$ for the times $t_0, t_{N_t/2+1},\ldots,t_{N_t}$. The second register provides the remainder of the encoding of the time. The third register is of dimension N_x, and encodes b or $x_{\rm in}$.

By performing a rotation on the first qubit, based on the normalisations of x_{in} and b, we obtain the correct relative weighting of the two time ranges. Then, conditional on the qubit being in the state $|0\rangle$, we can prepare a superposition of times $t_1, \ldots, t_{N_t/2}$ in the second register, as well as a state encoding b in the third register. Conditional on the qubit being in the state $|1\rangle$, we can prepare the time t_0 in the second register, and x_{in} in the third register. The complexity of these controlled state preparations will be $O(\sqrt{s})$. The complexity of preparing the superposition of times can be made $O(\log(N_t))$ simply by choosing N_t to be a power of two (which does not change the scaling). □

We now translate the complexity of solving linear systems into the complexity of obtaining a state corresponding to the solution of the differential equation.

Theorem 9. *Suppose that the multistep method is order p and $A(\alpha)$-stable, the matrix A may be diagonalised as $A = VDV^{-1}$, the eigenvalues of A all satisfy $|\arg(-\lambda_i)| \leq \alpha$, and*

$$\max_{t\in[t_0,t_0+\Delta t]} \|x(t)\| = O(\|x(t_0+\Delta t)\|), \tag{59}$$

$$\epsilon = o(\|x_{\text{in}}\|). \tag{60}$$

Then a state encoding the solution of the differential equation at time $t_0 + \Delta t$ may be obtained to within trace distance ϵ using

$$\tilde{O}\left(\log(N_x) s^{9/2} (\|A\|\Delta t)^{5/2} \kappa_V^{23/4} (\|x_{\text{in}}\| + \|b\|/\|A\|)^{5/4} \|x(t_0+\Delta t\|/\epsilon^{9/4}\right) \tag{61}$$

calls to the oracles for A, b, and x_{in}.

Proof. There are two main issues that we need to take into account in determining the complexity of obtaining the solution of the differential equation. The first is the probability of obtaining the correct time, and the second is the relation of the allowable error for solving the differential equation to the allowable error for the solution of the linear systems.

To obtain the correct final state, one would need to measure the time in the ancilla register, and obtain a value in the range $t_0 + \Delta t$ to $t_0 + 2\Delta t$. This probability can, in principle, be small or even zero. However, the conditions of the theorem ensure that it is not. The probability of success is given by

$$p_{\text{time}} = \frac{\sum_{j=N_t/2}^{N_t} \|x_j\|^2}{\langle\psi|\psi\rangle}. \tag{62}$$

The normalisation of the state is given by

$$\begin{aligned}
\langle\psi|\psi\rangle &= \sum_{j=0}^{N_t} \|x_j\|^2 \\
&= \sum_{j=0}^{N_t} [|x(t_0+hj)\| + O(\epsilon)]^2 \\
&= O\left(N_t \max_{t\in[t_0,t_0+\Delta t]} \|x(t)\|^2\right) \\
&= O\left(N_t \|x(t_0+\Delta t)\|^2\right).
\end{aligned} \tag{63}$$

Here we have bounded the error in the state at all times by ϵ. This is because we choose parameters that bound the error at time $t_0 + \Delta t$ by ϵ. The bound on the error increases monotonically with time, so the error at earlier times will also be bounded by ϵ. We have also used the conditions (59) and (60) to obtain that $\epsilon = o(\|x(t_0 + \Delta t)\|)$. Using this result for $\langle\psi|\psi\rangle$ we obtain

$$\begin{aligned}
p_{\text{time}} &= \Omega\left(\frac{\sum_{j=N_t/2}^{N_t} [\|x(t_0+\Delta t)\| + O(\epsilon)]^2}{N_t \|x(t_0+\Delta t)\|^2}\right) \\
&= \Omega(1).
\end{aligned} \tag{64}$$

Therefore the probability of obtaining the correct time does not change the scaling of the complexity.

The other main issue that needs to be addressed to obtain the overall scaling is the scaling in the error. The ϵ_L used in the expression for the solution of the linear systems is the error in the state encoding all times, not the error in

the estimate of $x(t_0 + \Delta t)$ obtained (for which we use ϵ). To determine the relationship between ϵ and ϵ_L, we use the bound on the normalisation of the state. In solving for the state $|\psi\rangle$, we obtain the state coefficients approximating

$$x(t_0 + \Delta t)/\sqrt{\langle\psi|\psi\rangle}. \tag{65}$$

Using the bound on the normalisation in Eq. (63), if the error in the coefficients is no more than ϵ_L, the error in the solution of the differential equation will be $O(\epsilon_L\sqrt{N_t}\|x(t_0 + \Delta t\|)$. Therefore, the error in the solution of the differential equation will be bounded by ϵ if we take

$$\epsilon_L = \Theta\left(\epsilon/[\sqrt{N_t}\|x(t_0 + \Delta t)\|]\right). \tag{66}$$

Using this value of ϵ_L in the scaling from Ref. [10], together with $\kappa = O(N_t\kappa_V)$ from Theorem 7, and the complexity of state preparation from Lemma 8, the number of oracle calls is

$$\tilde{O}(\log(N_x)s^{9/2}N_t^{5/2}\kappa_V^2\|x(t_0 + \Delta t\|/\epsilon). \tag{67}$$

Note that we can omit $\log(N_t)$ from Lemma 8, because the $\tilde{O}$ notation omits logarithmic factors. For the same reason, we have replaced $N = N_xN_t$ with N_x. We use a value of N_t that is sufficient to ensure that the error is no greater than ϵ, and that $h = O(1/\|A\|)$, which is a condition needed to use Theorem 7. If we take

$$N_t = \Theta\left(\|A\|\Delta t\sqrt{\frac{\kappa_V^3(\|x_{\rm in}\| + \|b\|/\|A\|)}{\epsilon}} + (\|A\|\Delta t)^{1+1/p}\left(\frac{\kappa_V^2(\|x_{\rm in}\| + \|b\|/\|A\|)}{\epsilon}\right)^{1/p}\right), \tag{68}$$

then using Lemma 4, the error will be bounded by ϵ. In addition, because $\epsilon = o(\|x_{\rm in}\|)$, we obtain $N_t = \Omega(\|A\|\Delta t)$, which ensures that $h = O(1/\|A\|)$.

We can simplify the result by taking

$$N_t = \Theta\left((\|A\|\Delta t)^{1+1/p}\sqrt{\frac{\kappa_V^3(\|x_{\rm in}\| + \|b\|/\|A\|)}{\epsilon}}\right). \tag{69}$$

Then the overall scaling of the number of black-box calls is

$$\tilde{O}\left(\log(N_x)s^{9/2}(\|A\|\Delta t)^{(5/2)(1+1/p)}\kappa_V^{23/4}(\|x_{\rm in}\| + \|b\|/\|A\|)^{5/4}\|x(t_0 + \Delta t\|/\epsilon^{9/4}\right). \tag{70}$$

Because we are using the $\tilde{O}$ notation, which omits sublinear terms, we can omit $1/p$ in giving the result. □

This result is somewhat conservative, because we have included the term due to the error in starting the linear multistep method. If we assume that this error is negligible, then we obtain

$$\tilde{O}\left(\log(N_x)s^{9/2}(\|A\|\Delta t)^{(5/2)(1+1/p)}\kappa_V^{2+5/p}(\|x_{\rm in}\| + \|b\|/\|A\|)^{5/2p}\|x(t_0 + \Delta t\|/\epsilon^{1+5/2p}\right). \tag{71}$$

This has the same scaling in $\|A\|\Delta t$, but improved scaling in other quantities.

VI. CONCLUSIONS

A quantum computer may be used to solve sparse systems of linear differential equations, provided the result may be encoded in a quantum state, rather than given explicitly. By encoding the differential equation as a linear system, and using the algorithm of Ref. [10] for solving linear systems, the complexity is (including only scaling in $\|A\|$ and Δt),

$$\tilde{O}\left((\|A\|\Delta t)^{5/2}\right). \tag{72}$$

This improves upon previous results for nonlinear differential equations, which were strongly exponential in Δt [2]. This algorithm has an enormous range of possible applications, because large systems of differential equations are ubiquitous in science and engineering. In particular they arise from the discretisation of partial differential equations.

These results are for constant coefficients, because that enables an analytic error analysis. This approach can also be used to solve linear differential equations with time-dependent coefficients, though the error analysis will be more difficult.

An interesting question is how the complexity will scale if the method referred to in Ref. [11] can be used. That reference refers to work in preparation providing an improved scaling for solving linear systems. The scaling is close to linear in the condition number, which would improve the scaling in $\|A\|\Delta t$ for the solution of linear differential equations.

Another interesting direction for future research is the problem of nonlinear differential equations. It is likely that the exponential scaling obtained in Ref. [2] is fundamental, because quantum mechanics is linear. However, it may potentially be possible to improve the constant of the exponential scaling.

Acknowledgments

The author is grateful for enlightening discussions with Andrew Childs and Jon Tyson.

---

[1] D. Kothe *et al.*,"Science at the Petascale 2009: Pioneering Applications Point the Way", National Center for Computational Sciences, http://www.nccs.gov/wp-content/media/nccs_reports/PetaDocLowRes2-9-10.pdf (accessed 13 October, 2010).

[2] S. K. Leyton and T. J. Osborne, "A quantum algorithm to solve nonlinear differential equations", arXiv:0812.4423 (2008).

[3] S. Lloyd, "Universal quantum simulators", Science **273**, 1073 (1996).

[4] D. Aharonov and A. Ta-Shma, "Adiabatic quantum state generation and statistical zero knowledge", in *Proceedings of the 35th ACM Symposium on Theory of Computing, 2003* (ACM, New York, 2003), pp. 20-29; arXiv:quant-ph/0301023.

[5] A. M. Childs, "Quantum information processing in continuous time", Ph.D. thesis, Massachusetts Institute of Technology, 2004.

[6] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, "Efficient quantum algorithms for simulating sparse Hamiltonians", Commun. Math. Phys. **270**, 359 (2007).

[7] A. M. Childs, "On the relationship between continuous- and discrete-time quantum walk", Commun. Math. Phys. **294**, 581 (2009).

[8] D. W. Berry and A. M. Childs, "Black-box Hamiltonian simulation and unitary implementation", arXiv:0910.4157 (2009).

[9] N. Wiebe, D. W. Berry, P. Høyer, and B. C. Sanders, "Higher order decompositions of ordered operator exponentials", J. Phys. A: Math. Theor. **43**, 065203 (2010).

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

[11] A. Ambainis, "New Developments in Quantum Algorithms", arXiv:1006.4014 (2010).

[12] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations* (Wiley, Chichester England, 2008).

[13] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems* (Springer-Verlag, Berlin, 1991).

[14] G. Dahlquist, "A special stability problem for linear multistep methods", BIT **3**, 27 (1963).

[15] O. B. Widlund, "A note on unconditionally stable linear multistep methods", BIT **7**, 65 (1967).

[16] R. D. Grigoreff and J. Schroll, "Über $A(\alpha)$-stabile Verfahren hoher Konsistenzordnung", Computing **20**, 343 (1978).

[17] L. Grover and T. Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions", arXiv:quant-ph/0208112 (2002).

[18] L. K. Grover, "Synthesis of quantum superpositions by quantum computation", Phys. Rev. Lett. **85**, 1334 (2000).