Quantum and classical algorithms for nonlinear unitary dynamics

Noah Brüstle[1] and Nathan Wiebe[1,2,3]

[1]Department of Computer Science, University of Toronto, Toronto, Canada
[2]Pacific Northwest National Laboratory, Richland, USA
[3]Canadian Institute for Advanced Research, Toronto, Canada

Quantum algorithms for Hamiltonian simulation and linear differential equations more generally have provided promising exponential speed-ups over classical computers on a set of problems with high real-world interest. However, extending this to a nonlinear problem has proven challenging, with exponential lower bounds having been demonstrated for the time scaling. We provide a quantum algorithm matching these bounds. Specifically, we find that for a non-linear differential equation of the form $\frac{d|u\rangle}{dt} = A|u\rangle + B|u\rangle^{\otimes 2}$ for evolution of time T, error tolerance ϵ and c dependent on the strength of the nonlinearity, the number of queries to the differential operators that approaches the scaling of the quantum lower bound of $e^{o(T\|B\|)}$ queries in the limit of strong non-linearity. Finally, we introduce a classical algorithm based on the Euler method allowing comparably scaling to the quantum algorithm in a restricted case, as well as a randomized classical algorithm based on path integration that acts as a true analogue to the quantum algorithm in that it scales comparably to the quantum algorithm in tailored cases where sign problems are absent.

1 Introduction

The Hamiltonian Simulation Problem seeks to simulate quantum interactions with the use of a quantum computer; as was the first purpose proposed for the device when conceived by Richard Feynman, in 1982 [11]). This is achieved by solving the following differential equation:

$$i\frac{d|u\rangle}{dt} = H|u\rangle\,, \tag{1}$$

as defined by the Schrodinger equation describing the evolution of a quantum system; where $|u\rangle$ is a quantum state and H is a Hermitian matrix. This may be analytically resolved to $|u(t)\rangle = e^{-iHt}|u\rangle$. The numerical approximation of such $|u(t)\rangle$ is however believed to be exponentially hard for any classical algorithm in its generality [1]. The problem is, in fact, BQP-complete; meaning that any problem that a quantum computer can solve, with probability at least $\frac{2}{3}$, in time polynomial in the number of bits of input may be reduced to the Hamiltonian Simulation Problem. The existence of polynomial-time quantum algorithms for the Hamiltonian Simulation Problem is therefore of particular interest; further optimizations and improvements of the algorithm remain an active field of research [4] [25] [24] .

It is then natural to ask whether more general differential equations permit a quantum algorithm with an exponential asymptotic advantage over an adequately comparable classical algorithm. The problem of linear differential equations, removing the condition of our evolution remaining unitary, a.k.a:

$$\frac{d|u\rangle}{dt} = A|u\rangle\,, \tag{2}$$

for some complex matrix A, has been studied extensively [7][30]. In a paper by Berry [2], the problem was reduced to the quantum algorithm for solving linear systems of equations by Harrow,

Hassidim and Lloyd [15], providing us with the first algorithm with exponential improvement over known classical algorithms. Subsequent works expanded the context for which quantum differential equations solvers could be applied; [19] removed the restriction on the diagonalizability of the matrix, while [3] provided an algorithm capable of solving time-dependent equations.

Removing the condition of linearity has, however, proven to be more difficult. Previous papers [17] [10] provided us with heuristic quantum algorithms tackling the problem, but these papers fall short of providing a proper implementation or a rigorous analysis of the running time of the algorithms. A first successful attempt at providing a concrete algorithm to tackle the Quantum Nonlinear Differential Equations problem in a restricted case was made by Liu et al. in 2021 [22], using the Carleman Linearization technique that will be employed in this paper. However, the restrictions provided by the paper are bound to dissipative systems (a.k.a. with a loss of energy); this would exclude the initial Hamiltonian Simulation problem from our formalism. A subset of non-linear equations with no dissipative factor is further studied in [29], equally using Carleman Linearization. A different linearization technique, the Koopman-von Neumann linearization, was employed in [28] to solve a restricted class of norm-preserving nonlinear equations.

A natural nonlinear extension of the Hamiltonian Simulation problem is the study of nonlinear differential equation with a unitary solution. Indeed, nonlinear differential equations with unitary solutions describe several physical phenomena [31]; the Gross-Pitaevskii equation, for example, is a nonlinear differential equation that may be used to model quantum systems of identical bosons. [13]. Nonetheless; it is not clear whether we may gain any substantial advantage by using quantum computing for these equations, relative to a classical model. Quantum computing is an inherently linear theory; the operations we may apply to our quantum states are linear maps. Deterministic classical numerical methods with better stability for nonlinear equations generally make use at multiple copies of the state at each iteration (example: higher-order Runge-Kutta method), which translates poorly to Quantum computing due to the no-cloning theorem.

We will first briefly state our results in Section 2. Then, Section 3 consists of a proof of our no-go result that shows that sub-exponential scaling with time would allow us to violate state discrimination lower bounds. In Section 4, we will discuss potential algorithmic solutions approaching these bounds. Subsection 4 describes a linearization of equation 4. Subsection 4.1will make use of our linearization to construct a quantum algorithm. Finally, in subsection 4.3 we will provide a set of potential avenues to dequantize the previous algorithm.

2 Results

The fact that quantum mechanics is a linear theory provides a major obstacle in the path of finding fast quantum algorithms for solving non-linear differential equations. The need to generalize of the equations can be seen from the results of Lloyd et al. In [23], the following set of equations is proposed,

$$\frac{d\,|u\rangle}{dt} = f(|u\rangle)\,|u\rangle \tag{3}$$

where $f(u)$ is a $d \times d$ anti-hermitian matrix that is an order m polynomial of $|u\rangle$ and $\langle u|$. While [23] claims that a quantum algorithm polynomial in the evolution time T is possible for this context - the arguments presented are not fully rigorous, and fail to abide by the lower bounds that may be established on the simulation of these equations.

Due to the fact that we require non-linear operations in the above differential equation, we utilize tensor operators $(\cdot)^{\otimes p} : \mathbb{C}^D \mapsto \mathbb{C}^{Dp}$. These allow us to copy a state multiple times and express the evolution as an arbitrary polynomial of the elements of our state $|u\rangle$. We use this notation to be able to express the differential equation in the form of rectangular matrices $R_1, R_2, \dots R_k$,

$$\frac{du}{dt} = R_1 u + R_2 u^{\otimes 2} + ... + R_k u^{\otimes k} \tag{4}$$

such that the transformation defined by the equation describes a unitary evolution on the vector u. We note here that certain operations on complex vectors (such as the absolute values used in the Gross-Pitaevskii equation) may be represented as polynomials after representing our complex

numbers as pairs of reals. For example, the action of i on the complex number $(a + ib)$ can be expressed as

$$\begin{bmatrix} a \\ b \end{bmatrix} \mapsto \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{5}$$

and similarly complex conjugation takes the form of a Pauli-Z operation.

Our first result, proved in Section 3, adapts the complexity-theoretic arguments from Childs and Young [9] to the context of the equations that we are studying to obtain a lower bound on the query complexity of solving non-linear differential equations.

Lemma 1. *Let $u : \mathbb{R} \mapsto \mathbb{C}^D$ for a positive integer D and let there exists a choice of matrices R_1 to R_k be rectangular matrices such that $R_j \in \mathbb{C}^{D \times D^j}$ and a d-sparse time-dependent Hamiltonian $H : \mathbb{R} \mapsto \mathbb{C}^{D \times D}$ such that:*

$$i\frac{du}{dt} = H(t)u + R_1 u + R_2 u^{\otimes 2} + ... + R_k u^{\otimes k}$$

describe a norm preserving evolution on u such that for every t there exists a unitary matrix V such that $u(t) = Vu(0)V^\dagger$. No algorithm can provide an approximate solution $\tilde{u}(t)$ such that $\|\tilde{u}(t)\tilde{u}^\dagger(t) - u(t)u(t)^\dagger\|_{\mathrm{Tr}} \leq 2\epsilon$ for any constant ϵ with fewer than $e^{\Omega(t \times \sum_{j=1}^{k} \|R_j\| 2^{-j/2} j)}$ queries to an oracle O_u on $u(0)$.

We note that the proof of this lower bound applies to the family of equations studied by [23]. Previous works such as [21] provide an algorithm with exponential time scaling respecting the bounds of Lemma 1; however, it is not shown that these methods saturate the bound. Furthermore, the algorithm also scales exponentially in $\frac{1}{\epsilon}$. Given that more recent Hamiltonian simulation results were able to provide logarithmic scaling with the inverse of the error [5] [25], this provides a clear avenue for improvement. Deciding whether it is possible to find an algorithm that has a query complexity that scales logarithmically with the error for the nonlinear context and also matches the scaling matching provided in Lemma 1 remained an open problem. Our paper provides an algorithm capable of doing so.

Theorem 2. *Suppose we have for bounded matrices A and B an equation of the form:*

$$\frac{d\,|u\rangle}{dt} = A\,|u\rangle + B\,|u\rangle^{\otimes 2}$$

Where A is $d_{\|A\|}-$sparse and B is $d_{\|B\|}-$sparse, and the dimension of u is bounded by: $\dim(|u\rangle) = 2^n$. Then there exists a quantum algorithm simulating $u(T)$ for a time $T > 0$ within error E with respect to the Euclidean norm for some arbitrarily small constant δ' with at most

$$O\left(\frac{e^{2Tc(1+\delta')}(d_A^2\|A\| + d_B^2\|B\|)\log(1/E)n}{\log\log(1/E)}\right)$$

oracle queries, where c is in $O(\|B\|)$.

While the differential equation in Theorem 2 is focused on quadratic non-linearities this can easily be generalized to any polynomial case [12].

The relation between the run times of our quantum algorithm, and established deterministic classical numerical methods such as the Runge-Kutta method, is not entirely clear. While it is known that we may replicate any deterministic classical algorithm with a quantum algorithm using polynomial resources [27], it may not be excluded that a polynomially optimal quantum algorithm using our input model could run exponentially worse than deterministic classical numerical methods that use different oracles for their inputs. This apparent contradiction stems from the fact that our quantum algorithms have access to a weaker oracular model; deterministic algorithms generally can directly access the values of the vector u; while the quantum algorithm can only obtain a quantum state representing the input vector, more akin to a probability distribution. Thus; to create a classical equivalent to this problem, we propose an input model of a probability vector. Oracle queries to the input vector for this model are then random samples from our probability vector. To perform our operations on this large random vector, we will also proceed by a random

sampling algorithm, making use of the well-known Path Integral Montecarlo method [14]. As is often the case for these types of algorithms, the runtime will depend on the variation in the output of our sampling - which may be very large, depending on the input to the problem. We will treat this variance as a separate variable - a more precise definition will be provided later.

Theorem 3. *Suppose we have for bounded matrices A and B an equation of the form:*

$$\frac{d\left|u\right\rangle}{dt} = A\left|u\right\rangle + B\left|u\right\rangle^{\otimes 2}$$

Where A and B are d_A and d_B-sparse respectively, $|u\rangle$ is a norm 1 vector in $\mathbb{C}^{2^n}$ and the assumption is made that the linear transformation on $|u\rangle$ remains unitary at all times t. Suppose that the path integral formulation for the Carleman block matrix of this problem has a variance of $\mathbb{V}(V)$. Then there exists a classical algorithm which outputs $v(j)$ such that:

$$\|v(j) - \langle u(0)|\, U(T)\, |u(0)\rangle\| \leq \epsilon$$

for a time $T > 0$ concerning the Euclidean norm, using at most

$$O\left(\frac{\mathbb{V}(V)(d_A^2 + d_B^2)e^{O(Tc(1+\delta))}}{\epsilon^2}\right)$$

queries to our oracles, with our constants defined as previously.

We note here that the constant c used is identical to the constant used in theorem 2. We note that if we allow linear time scaling in n, the logarithm of the dimension, time scaling matching the quantum case is demonstrated in the same section. This is of interest, as the lower bound arguments for Lemma 1 do not rely on dimension scaling. We may further note that the output produced by our quantum algorithm must also be sampled to obtain an estimate of the expectation calculated by our classical algorithm. This process loses the logarithmic error scaling, making it instead linear with the inverse of ϵ, as shown in Corollary 13. Thus the classical algorithm only has quadratically worse scaling with the error.

For a more restricted case of equations which we will name *Geometrically Local Equations* (note: these are strictly distinct from the quantum notion of a geometrically local Hamiltonian), an algorithm polynomially matching the quantum algorithm is achievable, as proposed in the following theorem,

Theorem 4. *Suppose we have for bounded matrices A and B an equation of the form:*

$$\frac{d\left|u\right\rangle}{dt} = A\left|u\right\rangle + B\left|u\right\rangle^{\otimes 2},$$

where $|u\rangle$ is a norm 1 vector in $\mathbb{C}^{2^n}$ and the assumption is made that $|u(t)\rangle$ remains a unit vector at all times t. Assume further that our equation is geometrically $k-$local in $\mathbb{Z}^D$. Then there exists a classical algorithm which outputs $v(j)$ in $\mathbb{C}^{2^n}$ such that:

$$\|v(j) - |u(T)\rangle\| \leq \epsilon$$

for a time $T > 0$ with respect to the Euclidean norm, using at most

$$O\left(\frac{(kT)^{D+1}e^{2(D+1)(|A|+2|B|)T}}{(2\epsilon)^{D+2}}\right)$$

queries to classical oracles O_A, O_B such that $O_A(i,j)$ yields the value of the i^{th} non-zero matrix element in row j and similarly $O_B(i,j)$ yields the corresponding matrix element in row j of B and oracles f_A, f_B such that $f_A(x) = y$ if and only if A_{xy} is the i^{th} non-zero matrix element in row x and f_B yields the corresponding locations of the non-zero matrix elements of B.

3 Quantum Lower Bound Argument

We will make use of the arguments from [9] to create a lower bound for the problem we are studying. Our lower bound builds upon the results from section 2 in [9], which we summarize in the following lemma. We will require equations of the form:

$$i\frac{d}{dt}|\psi\rangle = H(t)|\psi\rangle + K|\psi\rangle, \tag{6}$$

where K is a non-linear operator defined by:

$$\langle x|(K|\psi\rangle) = \kappa(|\langle x|\psi\rangle|)\langle x|\psi\rangle, \tag{7}$$

for any computational basis vector $|x\rangle$, and some function $\kappa : \mathbb{R}^+ \to \mathbb{R}^+$. Thus, K is a diagonal matrix with elements $\kappa(|\langle x|\psi\rangle|)$. $H(t)$ is a time-dependent hermitian operator chosen according to K. To avoid the quadratic term of $\kappa(|\langle x|\psi\rangle|)$ and allowing us to construct odd polynomials; we may also alternatively define, for $\langle x|\psi\rangle = a_x + b_x i$,

$$\langle x|(K|\psi\rangle) = \kappa(a+b)\langle x|\psi\rangle, \tag{8}$$

We note here that for the restricted domain $b_x = 0$, $a_x \geq 0$ in which the arguments of [9] are made, this alternative definition is equivalent. We will equally make use of the following definition from Equation 8 of [9]:

$$\bar{\kappa}(z) := \kappa\left(\left(\frac{1+z}{2}\right)^{1/2}\right) - \kappa\left(\left(\frac{1-z}{2}\right)^{1/2}\right). \tag{9}$$

Then; in the Bloch sphere representation, our state $|\phi\rangle = (x, y, z)$ under the action of only the nonlinearity will act as:

$$\frac{d}{dt}(x, y, z) = \bar{\kappa}(z)(-y, x, 0). \tag{10}$$

Then the following lemma applies:

Lemma 5 (Section 2.3 of [9])**.** *Suppose there are constants $g, \delta > 0$ such that $\bar{\kappa}(z) \geq gz$ for all $0 \leq z \leq \delta$. Then, evolving the equation 6 for time $O(\frac{1}{g}\log(\frac{1}{\epsilon}))$ will allow us to distinguish 2 chosen states $|u\rangle$ and $|v\rangle$ with overlap $|\langle u|v\rangle| \leq 1-\epsilon$ with success probability greater than $\frac{4}{5}$.*

First, we must define the oracles we will use throughout this paper.

Definition 6 (Oracles O_A, O_B, O_u)**.** *We define quantum oracles that provide us with access to the initial state $u(0)$ as well as the matrices A and B in the following manner.*

- *Given a matrix $A \in \mathbb{C}^{2^n \times 2^n}$ of sparsity d_A, integers $i \leq 2^n$, $j \leq d_A$. We will define the oracle $O_A|i\rangle|j\rangle|0\rangle|0\rangle = O_A|i\rangle|j\rangle|p\rangle|q\rangle$ where p is an integer indicating the location of the $j-th$ non-zero column of row i of A, and q is a binary representation of $A[i,p]$.*
- *Given a matrix $B \in \mathbb{C}^{2^n \times 2^{2n}}$ of sparsity d_B, integers $i \leq 2^n$, $j \leq d_B$. We will define the oracle $O_B|i\rangle|j\rangle|0\rangle|0\rangle = O_B|i\rangle|j\rangle|p\rangle|q\rangle$ where p is an integer indicating the location of the $j-th$ non-zero column of row i of B, and q is a binary representation of $B[i,p]$.*
- *Given an initial state vector $|u(0)\rangle \in \mathbb{C}^{2^n}$, we define O_u as the unitary operator that prepares the initial state via $O_u|0\rangle^{\otimes n} = |u(0)\rangle$.*

We will use lemma 5 and the above definition of O_u to prove our main lemma of this section, which is a strengthening of the lower bounds contained in [9].

Lemma 1. *Let $u : \mathbb{R} \mapsto \mathbb{C}^D$ for a positive integer D and let there exists a choice of matrices R_1 to R_k be rectangular matrices such that $R_j \in \mathbb{C}^{D \times D^j}$ and a d-sparse time-dependent Hamiltonian $H : \mathbb{R} \mapsto \mathbb{C}^{D \times D}$ such that:*

$$i\frac{du}{dt} = H(t)u + R_1 u + R_2 u^{\otimes 2} + ... + R_k u^{\otimes k}$$

describe a norm preserving evolution on u such that for every t there exists a unitary matrix V such that $u(t) = Vu(0)V^\dagger$. No algorithm can provide an approximate solution $\tilde{u}(t)$ such that $\|\tilde{u}(t)\tilde{u}^\dagger(t) - u(t)u(t)^\dagger\|_{\mathrm{Tr}} \leq 2\epsilon$ for any constant ϵ with fewer than $e^{\Omega(t\times\sum_{j=1}^{k}\|R_j\|2^{-j/2}j)}$ queries to an oracle O_u on $u(0)$.

Proof. Let us make a choice of R_j such that for all $|x\rangle$ and $|u\rangle$

$$\langle x|\left(R_j|u\rangle^{\otimes j}\right) = a_j|\langle x|u\rangle|^j\langle x|u\rangle \tag{11}$$

Then we have that:

$$\lim_{z\to 0}\bar{\kappa}(z) = \lim_{z\to 0}\frac{a_1\left(\frac{1+z}{2}\right)^{1/2} + a_2\left(\frac{1+z}{2}\right)^{2/2} + ... + a_k\left(\frac{1+z}{2}\right)^{k/2}}{z}. \tag{12}$$

We make use of l'Hospital's rule to evaluate this limit, differentiating the nominator and denominator;

$$\lim_{z\to 0}\bar{\kappa}(z) = \lim_{z\to 0}\frac{\sum_{j=1}^{k}2^{-j/2-1}a_j(1+z)^{j/2-1} + 2^{-j/2-1}a_j(1-z)^{j/2-1}}{1} = \sum_{j=1}^{k}2^{-j/2}a_j. \tag{13}$$

Thus; as these are analytic functions, we may say that g as defined in Lemma 5 may take any value greater than $\sum_{j=1}^{k}2^{-j/2}a_j$; and thus, we may distinguish $|\psi\rangle, |\phi\rangle$ by simulating our differential equation for time $T = O\left(\frac{1}{\sum_{j=1}^{k}2^{-j/2}a_j}\log\left(\frac{1}{\epsilon}\right)\right)$. We note here that for such a κ, equation 6 may be written in the form:

$$\frac{du}{dt} = R_1 u + R_2 u^{\otimes 2} + ... + R_k u^{\otimes k}, \tag{14}$$

as required for Lemma 1, with $\|R_{j+1}\| = a_j$.

Suppose we have access to an oracle to some unknown state, and are promised that the oracle either yields a state operator $|\psi\rangle\langle\psi|$ or $|\phi\rangle\langle\phi|$ such that $\||\psi\rangle\langle\psi| - |\phi\rangle\langle\phi|\|_{\mathrm{Tr}} \geq 2\Delta$ for $\Delta > 0$. By the Helstrom bound, $N = \theta\left(\frac{1}{\Delta}\right)$ copies of our state (or oracle queries to O_u) are necessary to determine with probability $\frac{2}{3}$ which state the oracle generates. (This is an older result, but is stated in a similar form in Section 4 of [9]). The success rate of our algorithm for Lemma 5 depends on the distance between our states at the end of our evolution; we may permit an error ϵ sufficiently small to maintain a success rate greater than 2/3 with triangle inequality.

Suppose, now, that the differential equation can be simulated for time t and error ϵ with $e^{o(t\times\sum_{j=2}^{k}a_j2^{-j/2}j)}$ queries to our oracle on our state $|\psi\rangle$ or $|\phi\rangle$. Then using our algorithm in Lemma 5, the number N of oracle queries required to distinguish $|\psi\rangle$from $|\phi\rangle$ with success probability $\geq 2/3$ will be:

$$N = e^{o(T\times\sum_{j=2}^{k}a_j2^{-j/2}j)} = e^{o\left(\left(\frac{1}{\sum_{j=1}^{k}2^{-j/2}a_j}\log\left(\frac{1}{\epsilon}\right)\right)\times\sum_{j=2}^{k}a_j2^{-j/2}j\right)}, \tag{15}$$

$$N = e^{o\left(\log\left(\frac{1}{\epsilon}\right)\right)} = o\left(\frac{1}{\epsilon}\right). \tag{16}$$

This violates the Helstrom bound, and thus, we conclude that no such algorithm for solving our differential equation exists. This concludes our proof of Lemma 1. □

We will also further note that a lower bound of $d\|A\|t$ will be inherited from the regular Hamiltonian Simulation problem; which is shown, for example, in [6].

4 Quantum Algorithms for Non-Linear Differential Equations

We will now attempt to find algorithms that approach our exponential lower bound from the previous section.

As in [22] [29], we will start by making use of the Carleman Linearization process to obtain a higher-dimensional linear approximation of our equation. As a first step; we reduce our polynomial differential equation into the quadratic case. That this is permitted is a result taken directly from [12].

Lemma 7 (Proposition 3.4 of [12])**.** *Let $u : \mathbb{R} \mapsto \mathbb{C}^{D\times D}$ and let $R_j \in \mathbb{C}^{D\times D^j}$. The k-th order system ($k \geq 2$):*

$$\frac{du}{dt} = R_1 u + R_2 u^{\otimes 2} + ... + R_k u^{\otimes k}$$

may be reduced to a quadratic system in $\tilde{u}$, that is there exist matrices A and B such that

$$\frac{d\tilde{u}}{dt} = A\tilde{u} + B\tilde{u}^{\otimes 2} \tag{17}$$

where $\tilde{u} := \{u, u^{\otimes 2}, u^{\otimes 3}, \ldots, u^{\otimes(k-1)}\}$. Moreover; A, B respect:

$$\|A\| \leq \max_{1\leq i\leq k-1}(k-i)\sum_{j=1}^{i}\|R_j\|$$

$$\|B\| \leq (k-1)\sum_{j=2}^{k}\|R_j\|$$

here $\|\cdot\|$ refers to the spectral norm.

We will continue using $\|\cdot\|$ as a reference to the spectral norm throughout the paper.

To describe our matrices A and B, it will first be necessary to define the *transfer matrices*:

$$T^i_{i+j} := \sum_{j=0}^{m-1} I^{\otimes j} \otimes R_i \otimes I^{\otimes(k-j-1)} \tag{18}$$

Our matrices A and B will be described as:

$$A = \begin{bmatrix} T^1_1 & T^1_2 & T^1_3 & \ldots & T^1_{k-1} \\ 0 & T^2_2 & T^2_3 & \ldots & T^2_{k-1} \\ 0 & 0 & T^3_3 & \ldots & T^3_{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & T^{k-1}_{k-1} \end{bmatrix} \tag{19}$$

$$B = \begin{bmatrix} T^1_k & 0 & 0 & \ldots & 0 \\ T^2_k & T^2_{k+1} & 0 & \ldots & 0 \\ T^3_k & T^3_{k+1} & T^3_{k+2} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T^{k-1}_k & T^{k-1}_{k+1} & T^{k-1}_{k+2} & \ldots & T^{k-1}_{2(k-1)} \end{bmatrix} \otimes \langle k-1| \tag{20}$$

We note that $\|T^i_{i+j}\| \leq j\|T_j\|$ hold by the triangle inequality and the subadditivity of the operator norm; our bounds on the norms of A and B follow. A further important consequence of this is that the sparsity remains manageable; T^i_{i+j} has a sparsity of at most jd_{R_i}, where d_{R_i} is the sparsity of R_i. We may then say the following:

Lemma 8. *Given matrix A and B as defined in lemmma 7, the sparsities d_A and d_B may be bounded as:*

$$d_A \leq \max_{1\leq i\leq k-1}(k-i)\sum_{j=2}^{k}(j-i)d_{R_i}$$

$$d_B \leq \max_{1\leq i\leq k-1}\sum_{j=1}^{i}(k-j)d_{R_i}$$

The transformation of Lemma 7 is exact and will preserve unitarity in the higher dimensional space if A and B are anti-Hermitian. Let us then define:

$$u_m = \tilde{u}^{\otimes m} \tag{21}$$

$$A_m = \sum_{j=0}^{m-1} I^{\otimes j} \otimes A \otimes I^{\otimes(k-j-1)} \tag{22}$$

$$B_m = \sum_{j=0}^{m-1} I^{\otimes j} \otimes B \otimes I^{\otimes(k-j-1)} \tag{23}$$

We note that $\|A_m\| \leq m\|A\|$ and $\|B_m\| \leq m\|B\|$ hold by the triangle inequality and the subadditivity of the operator norm. The following lemma describes our Carleman linearization, as shown in [12],

Lemma 9 (Proposition 3.2 of [12])**.** *If $\tilde{u}$ solves equation 17, then*

$$\frac{du_m}{dt} = A_m u_m + B_m u_{m+1}.$$

The equation of Lemma 9 may be rewritten as the following infinite dimensional block matrix equation,

$$\begin{bmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \\ \frac{du_3}{dt} \\ \vdots \end{bmatrix} = \begin{bmatrix} A_1 & B_1 & 0 & 0 & \dots \\ 0 & A_2 & B_2 & 0 & \dots \\ 0 & 0 & A_3 & B_3 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \end{bmatrix}.$$

We define our solution vector to the above equation as u_∞ and the infinite-dimensional block matrix as G. Unless otherwise specified; we will here on out treat each block as a unit, rather than referring to specific coordinates of the matrix; thus, $G_{2,3} = B_2$, for example. As shown in [12], this solves for each u_m exactly. It remains to find an appropriate approximation of this method for u_1, which corresponds to the solution of our initial nonlinear equation. We proceed by a truncation,

$$\begin{bmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \\ \frac{du_3}{dt} \\ \vdots \\ \frac{du_{m-1}}{dt} \\ \frac{du_m}{dt} \end{bmatrix} \approx \begin{bmatrix} A_1 & B_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & A_2 & B_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & A_3 & B_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \\ 0 & 0 & 0 & 0 & \dots & A_{m-1} & B_{m-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & A_m \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{m-1} \\ u_m \end{bmatrix}.$$

We will use G_m to describe the above truncation of G. Theorem 4.2 in [12] provides us an estimate of our error for $0 < t < \frac{1}{\|B\|}$; however, their estimate diverges with m for $t > \frac{1}{\|B\|}$. As is common with approximation techniques, it is, therefore, necessary for us to divide our evolution time into multiple smaller time steps.

We note that this requires us to update not only u_1 at a given step; but also u_2 to u_m, since our linear system is dependent on these variables. To ensure that all variables are appropriately updated, a larger matrix is required at each additional time step we add. The starting m must therefore be sufficiently large to cover the required truncations at all steps. We will compute the required truncations for each step and the required starting m within this section.

For convenience, we enumerate our time steps as $j = 1$ to $j = J$, where $j = 1$ is the last time step. This is because the size of the matrix we consider is also shrinking as we approach the last time step. We define the following vector:

$$v(j, m) := \{v_1(t_j), v_2(t_j), \ldots, v_m(t_j))\} \tag{24}$$

as our approximation of

$$u(j, m) := \{u_1(t_j), u_2(t_j), \ldots, u_m(t_j)\} \tag{25}$$

Let

$$v(j) := v(j, m_j) \tag{26}$$

$$u(j) := u(j, m_j) \tag{27}$$

m_j corresponds to the value of the m used for our truncated block matrix G_m at step j. (The value of m_j will be chosen later.) The error ϵ_j on our vector $v(j)$ is defined as

$$\epsilon_j := |u(j) - v(j)|. \tag{28}$$

Define further our time interval leading to step j:

$$\hat{t}_j := t_{j-1} - t_j \tag{29}$$

Let $u[m]$ be the truncation of the block vector u to blocks 1 to m. Similarly, let $M[m_1, m_2]$ be the truncation of the block matrix M restricted to block rows 1 to m_2 and block columns 1 to m_2. To denote instead the exclusion of rows/columns 1 to m, write $\bar{m}$.

We may remark here that on the Lebesgue space with Euclidean norm L^2, for row-finite M, the initial value problem $\frac{dx}{dt} = Mx$, $x(0)$ has a unique solution $x(t) = e^{Mt}x(0) = \sum_{j\in\mathbb{N}} \frac{M^j t^j}{j!}x(0)$; further, in the case of matrices generated by Carleman Linearization, this solution will be finite everywhere [26] [18].

Thus, the exact solution at a given time step, $u(j)$ will be described as:

$$u(j) = \left(e^{\hat{t}_{j+1}G}u(j+1, \infty)\right)[m_j] = \left(e^{\hat{t}_{j+1}G}[m_j, \infty]u(j+1, \infty)\right) \tag{30}$$

Our approximation for the first m_j values will be

$$v(j) = \left(e^{\hat{t}_{j+1}G_{m_{j+1}}}v(j+1)\right)[m_j] = \left(e^{\hat{t}_{j+1}G_{m_{j+1}}}[m_j, m_{j+1}]v(j+1)\right) \tag{31}$$

In the following lemma; we will define the length of each time step and bound the error introduced at each time step by our truncation. The remainder of the section will serve to prove these bounds.

Lemma 10. *Let $u(j)$, $v(j)$, m_j be defined as in (30),(31) where G_{m_j} is the truncated Carleman matrix for step j. Let $m_j = \sum_{s=1}^{j-1} k_j + 1$ where $k_j = \log_{1+\delta}\left(\frac{1}{\epsilon}\right) + \log_{1+\delta}(j)$ and let $|v(j+1) - u(j+1)| \leq \epsilon_{j+1}$ for some $\epsilon > 0$, $\delta > 0$ and all $j \leq w$. Then there exists a constant c such that:*

1. *The truncation error at time step j may be bounded by:*

$$\left|e^{c\hat{t}_j G_{j+1}}v(j+1)|(m_j) - u(j)|\right| \leq \frac{\epsilon}{j\delta^2} + \epsilon_{j+1},$$

where

$$\hat{t}_j = \frac{1}{jc(1+\delta)}.$$

2. *Our evolution at time step t_j is $\frac{\epsilon}{j\delta^2}$ close to a unitary matrix with respect to the operator norm.*

3. *Given a time evolution with w timesteps, the total error of the truncation process $\epsilon_{tot} = |v(0) - u(0)|$ may then be bounded by :*

$$\epsilon_{tot} \leq \frac{\ln(w)\epsilon}{\delta^2}.$$

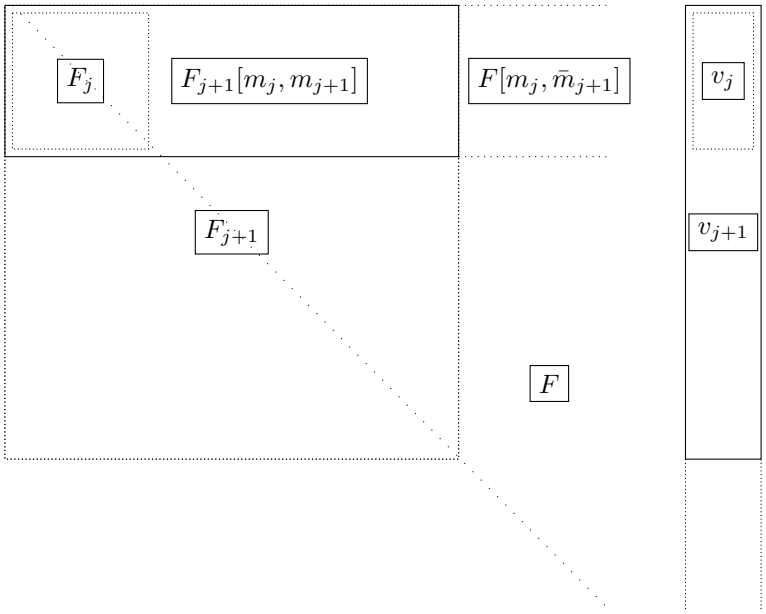


Figure 1: Illustration of 1 step of our process. F_{j+1} restricted to the space of m_j is applied onto v_{j+1} to return v_j.

Proof. Let

$$F(t) := e^{tG}, \tag{32}$$

and

$$F_j(t) := e^{tG_j}. \tag{33}$$

We note that $F(t)$ is identical to $F_j(t)$ on all coordinates where $F_j(t)$ is defined, due to G being block upper triangular, in particular:

$$F_{j+1}(t)[m_j, m_{j+1}] = F(t)[m_j, m_{j+1}] \tag{34}$$

Our error on v_j is a combination of carrying forward the existing error on v_{j+1} relative to u_{j+1}; and the tail of F that is thrown away. (See Figure 1). m_{j+1} is chosen to be sufficiently large to keep the latter error small. Thus we may bound our error vector as:

$$\|v(j) - u(j)\| = \|F_{j+1}(\hat{t}_j)[m_j, m_{j+1}]v(j+1) - F(\hat{t}_j)[m_j, \infty]u(j+1)\| \tag{35}$$

$$= \|F_{j+1}(\hat{t}_j)[m_j, m_{j+1}]v(j+1) - (F(\hat{t}_j)[m_j, m_{j+1}]u(j+1) + F(\hat{t}_j)[m_j, \bar{m}_{j+1}]u(j+1))\| \tag{36}$$

$$\leq \|F(\hat{t}_j)[m_j, m_{j+1}]v(j+1) - F(\hat{t}_j)[m_j, m_{j+1}]u(j+1)\| + \|(F(\hat{t}_j)[m_j, \bar{m}_{j+1}]u(j+1)\| \tag{37}$$

Now, as $F(\hat{t}_j)$ is a norm-preserving linear operator, the restriction $F(\hat{t}_j)[m_j, m_{j+1}]$ will be monotonically non-increasing. Further, $\|u(j)[\bar{m}_{j+1}]\| \leq \|u(j)\| = 1$. Thus we obtain:

$$\|v(j) - u(j)\| \leq \epsilon_{j+1} + \|F(\hat{t}_j)[m_j, \bar{m}_{j+1}]\|. \tag{38}$$

We note further that $F(\hat{t}_j)[m_j, \bar{m}_{j+1}]$ corresponds to the distance of our evolution from a unitary - as the exact evolution is unitary. Let us define k_j, representing the additional copies in our block vector required from the previous step $j+1$ to approximate our vector at step j:

$$k_j := m_{j+1} - m_j. \tag{39}$$

As previously, we treat F as a block matrix, acting on $\{v_1, v_2, v_3, ...\}$; $F_{l,k}$ will refer to the block at coordinates (l, k) in this block matrix. The tail of our truncation, $F(\hat{t}_j)[m_j, \bar{m}_{j+1}]$, may be bounded as:

$$\left\|F(\hat{t}_j)[m_j, \bar{m}_{j+1}]\right\| \leq \sum_{m=1}^{m_j} \sum_{s=k}^{\infty} \|F_{m,m_j+s}(\hat{t}_j)\|. \tag{40}$$

We note here that $m_j + k = m_{j+1}$. We will show that $\|F_{m_j,m_j+k}(\hat{t}_j)\|$ decays exponentially in k and $(m_j - m)$ for the correct choice of m_j, $\hat{t}_j$ and k_j, thus allowing us to bound this summation as a geometric series.

We define a new matrix $\hat{G}$ as :

$$\hat{G}_{i,j} := \|G_{i,j}\| \tag{41}$$

and define $\hat{F}$ as $e^{\hat{G}}$. An important property here is that:

$$\hat{F}_{i,j} \geq \|F_{i,j}\| \tag{42}$$

This is a direct consequence of the sub-additive and sub-multiplicative properties of the operator norm: $\|A + B\| \leq \|A\| + \|B\|$, $\|AB\| \leq \|A\|\|B\|$. $F_{i,j}$ will be expressed as:

$$F_{i,j} = \left(\sum_{k=0}^{\infty} \frac{G^k}{k!}\right)_{i,j} = \left(\sum_{k=0}^{\infty} \frac{(G^k)_{i,j}}{k!}\right) \tag{43}$$

$$\|(G^k)_{i,j}\| = \| \sum_{a_1 \in \mathbb{N}} \sum_{a_1 \in \mathbb{N}} ... \sum_{a_{k-1} \in \mathbb{N}} G_{i,a_1} G_{a_1,a_2} ... G_{a_{k-2},a_{k-1}} G_{a_{k-1},j} \| \tag{44}$$

$$\|(G^k)_{i,j}\| \leq \sum_{a_1 \in \mathbb{N}} \sum_{a_1 \in \mathbb{N}} ... \sum_{a_{k-1} \in \mathbb{N}} \|G_{i,a_1}\| \|G_{a_1,a_2}\| ... \|G_{a_{k-2},a_{k-1}}\| \|G_{a_{k-1},j}\| = (\hat{G}^k)_{i,j} \tag{45}$$

$$\|F_{i,j}\| = \| \sum_{k=0}^{\infty} \frac{(G^k)_{i,j}}{k!} \| \leq \sum_{k=0}^{\infty} \frac{\|(G^k)_{i,j}\|}{k!} \leq \sum_{k=0}^{\infty} \frac{(\hat{G}^k)_{i,j}}{k!} = \hat{F}_{i,j}. \tag{46}$$

Using $\|A_m\| \leq m\|A\|$, $\|B_m\| \leq m\|B\|$, we may diagonalize our matrix $\hat{G}$ as:

$$\hat{G} = \begin{bmatrix} \|A\| & \|B\| & 0 & 0 & \dots \\ 0 & 2\|A\| & 2\|B\| & 0 & \dots \\ 0 & 0 & 3\|A\| & 3\|B\| & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} = SJS^{-1}, \tag{47}$$

$$\hat{G} = \begin{bmatrix} 1 & \frac{\|B\|}{\|A\|} & \frac{\|B\|^2}{\|A\|^2} & \frac{\|B\|^3}{\|A\|^3} & \dots \\ 0 & 1 & \frac{2\|B\|}{\|A\|} & \frac{3\|B\|^2}{\|A\|^2} & \dots \\ 0 & 0 & 1 & \frac{3\|B\|}{\|A\|} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \|A\| & 0 & 0 & 0 & \dots \\ 0 & 2\|A\| & 0 & 0 & \dots \\ 0 & 0 & 3\|A\| & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} 1 & \frac{-\|B\|}{\|A\|} & \frac{\|B\|^2}{\|A\|^2} & \frac{-\|B\|^3}{\|A\|^3} & \dots \\ 0 & 1 & \frac{-2\|B\|}{\|A\|} & \frac{3\|B\|^2}{\|A\|^2} & \dots \\ 0 & 0 & 1 & \frac{-3\|B\|}{\|A\|} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{48}$$

Where $S[m, m+k] = \binom{m+k-1}{k} \left(\frac{\|B\|}{\|A\|}\right)^k$, $J[m,m] = m\|A\|$ and $S^{-1}[m, m+k] = \binom{m+k-1}{k} \left(\frac{-\|B\|}{\|A\|}\right)^k$. Inserting the constant factor of t, we obtain the same S, S^{-1} and $J[m,m] = tm\|A\|$.

Let's now demonstrate this diagonalization. It is clear that the eigenvalues of our infinite matrix $\hat{G}$ are $k\|A\|$ for all strictly positive integers k; we compute our corresponding eigenvectors from this. For some fixed eigenvalue $k\|A\|$,

$$j\|A\|v_k[j] + j\|B\|v_k[j+1] = k\|A\|v_k[j] \tag{49}$$

$$v_k[j+1] = \frac{(k-j)\|A\|v_k[j]}{j\|B\|} \tag{50}$$

Setting $v_k[1]$ to $\frac{\|B\|^{k-1}}{\|A\|^{k-1}}$:

$$\forall j \leq k : v_k[j] = \binom{k}{k-j} \frac{\|B\|^{k-j}}{\|A\|^{k-j}} \tag{51}$$

$$\forall j > k : v_k[j] = 0 \tag{52}$$

Which gives us the desired diagonalization.

Given some diagonalization SJS^{-1}, we know that $e^{SJS^{-1}} = Se^J S^{-1}$. Making use of this fact, we may then compute the non-zero entries of $\hat{F}(t) = e^{\hat{G}t}$ as follows,

$$\hat{F}_{m,m+k}(t) \leq \sum_{j=m}^{m+k} \binom{m + (j-m) - 1}{j-m} \frac{\|B\|^{j-m}}{\|A\|^{j-m}} e^{jt\|A\|} (-1)^{j+m+k-1} \binom{m+k-1}{m+k-j} \frac{\|B\|^{m+k-j}}{\|A\|^{m+k-j}} \tag{53}$$

$$\hat{F}_{m,m+k}(t) \leq \binom{k+m-1}{k} \frac{\|B\|^k}{\|A\|^k} \sum_{j=m}^{m+k} \binom{k}{j-m} e^{jt\|A\|} (-1)^{j+m+k-1} \tag{54}$$

$$\hat{F}_{m,m+k}(t) \leq \binom{k+m-1}{k} \frac{\|B\|^k}{\|A\|^k} e^{mt\|A\|} \sum_{j=0}^{k} \binom{k}{j} e^{jt\|A\|} (-1)^{k-j} \tag{55}$$

Using here, the binomial theorem:

$$\hat{F}_{m,m+k}(t) \leq \binom{k+m-1}{k} \frac{\|B\|^k}{\|A\|^k} e^{mt\|A\|} (e^{\|A\|t} - 1)^k \tag{56}$$

The binomial coefficients that appear above can be upper bounded by elementary functions via:

$$\binom{k+m-1}{k} \leq \left(\frac{e(m+k-1)}{k} \right)^k \tag{57}$$

We establish here that $\hat{t}_j \leq \frac{1}{\|A\|}$ for all j (we will force this later). Then:

$$(e^{\|A\|\hat{t}_j} - 1)^k \leq (\|A\|\hat{t}_j \times (e-1))^k \tag{58}$$

We will further have that

$$m_j \times \hat{t}_j \|A\| \leq k_j \tag{59}$$

Thus:

$$e^{m_j \|A\| \hat{t}_j} \leq e^{k_j}. \tag{60}$$

Set the constant c such that $c = \max\{\|A\|, e^2(e-1)\|B\|\}$. (Note here that $e^2(e-1) < 12.7$.) Then:

$$\|F_{m,m+k_j}(\hat{t}_j)\| \leq \hat{F}_{m,m+k_j}(\hat{t}_j) \leq \left(\frac{\hat{t}_j c(m_j + k_j - 1)}{k_j} \right)^{k_j}. \tag{61}$$

Further, for $k \geq k_j$ we have that

$$\|F_{m,m+k}(\hat{t}_j)\| \leq \left(\frac{\hat{t}_j c(m_j + k - 1)}{k} \right)^k. \tag{62}$$

holds as well.

For a given step j; we will now define

$$k_j = \log_{1+\delta}\left(\frac{1}{\epsilon}\right) + \log_{1+\delta}(j), \tag{63}$$

for some chosen δ. We remember that m_j represents the number of elements in our block vector $v(j)$ at step j; our final block vector at step 1 being of size 1 (we remember here that the index j runs in the opposite direction of our time, for convenience) and thus,

$$m_j = 1 + \sum_{s=1}^{j-1} k_j = \sum_{s=1}^{j-1} k_j \log_{1+\delta}\left(\frac{1}{\epsilon}\right) + \log_{1+\delta}(j), \tag{64}$$

$$m_j \leq j(\log_{1+\delta}\left(\frac{1}{\epsilon}\right) + \log_{1+\delta}(j)). \tag{65}$$

We then chose the size of our time step; defined as previously $\hat{t}_j = t_j - t_{j+1}$.

$$\hat{t}_j = \frac{1}{jc(1+\delta)}. \tag{66}$$

This respects all the bounds required previously. Find the inequality

$$\|F_{m_j,m_j+k_j}(\hat{t}_j)\| \leq \left(\frac{1}{1+\delta}\right)^{\log_{1+\delta}\left(\frac{1}{\epsilon}\right)+\log_{1+\delta}(j)} = \frac{\epsilon}{j}. \tag{67}$$

From equations 62 and 63, we may further obtain the bound:

$$\|F_{m_j,m_j+k_j+r}(\hat{t}_j)\| \leq \left(\frac{1}{1+\delta}\right)^{r} \frac{\epsilon}{j}. \tag{68}$$

Additionally:

$$\|F_{m_j-s,m_j-s+k_j}(\hat{t}_j)\| \leq \left(\frac{\frac{1}{jc(1-\delta)}c(m_j-s+k_j-1}{k_j}\right)^{k_j} \leq \|F_{m_j,m_j+k_j}(\hat{t}_j)\|. \tag{69}$$

From equation 68 and equation 69, we obtain:

$$\sum_{m=1}^{m_j}\sum_{s=k_j}^{\infty} \|F_{m,m_j+s}(\hat{t}_j)\| \leq \sum_{m=1}^{m_j}\sum_{s=k}^{\infty} \left(\frac{1}{1+\delta}\right)^{m_j-m}\left(\frac{1}{1+\delta}\right)^{s-k+1}\frac{\epsilon}{j} \leq \frac{\epsilon}{j\delta^2}. \tag{70}$$

As this represents our distance from the exact operator, which is known to be unitary, our evolution matrix has a distance of at most $\frac{\epsilon}{j\delta^2}$ from a unitary. Combining the previous equation with equations 38 and 40 we may obtain:

$$|u(j) - v(j)| \leq \epsilon_{j+1} + \frac{\epsilon}{j\delta^2}. \tag{71}$$

We may use this equation to describe our error over the evolution. Suppose that we require w steps to simulate our entire evolution. Setting ϵ_{N+1} to 0 (as this is our initial state), the total truncation error ϵ_{tot} may be described as follows:

$$\epsilon_{tot} \leq \sum_{j=1}^{w} \frac{\epsilon}{j\delta^2} \leq \frac{\ln(w)\epsilon}{\delta^2}. \tag{72}$$

□

We now have established the size of our time steps for our linearization process and the error introduced by our truncation at each time step. It remains to simulate these linear equations.

4.1 Proof of Theorem 2

We wish to simulate our equation for some time T. Using our bounds from Lemma 10, we may evolve our vector by $\hat{t}_j = \frac{1}{jc(1+\delta)}$ at time step j. Thus, to simulate our entire evolution, we require w time steps, such that

$$T \leq \sum_{j=1}^{w} \frac{1}{jc(1+\delta)}. \tag{73}$$

But we know that

$$\sum_{j=1}^{w} \frac{1}{jc(1+\delta)} \geq \frac{\log(w)}{c(1+\delta)}. \tag{74}$$

Thus, a sufficient condition is that,

$$w \geq e^{Tc(1+\delta)}. \tag{75}$$

We will simulate our evolution using a Linear Combination of Unitaries (LCU) method; applied upon a Taylor expansion of e^{Gt}. To evaluate our running time, it is first necessary to ensure that the norm of the matrices that we are simulating remains small. This requires us to splice our time steps further for our previously fixed matrices. We note that this time splicing does *not* require us to expand our matrix further; as our matrix at each time step has already been determined to be sufficiently accurate for that time step.

Lemma 11. *Define:* $\tau_j = \frac{\hat{t}_j}{2(\log_{1+\delta}(\frac{1}{\epsilon})+\log_{1+\delta}(j))}$. *Then for any given* F_j *as defined in equation 33, the following inequality holds:*

$$\|F_j(\tau_j)\| \leq e.$$

Proof. We note here that

$$\|F_j(\tau_j)\| = \|G_j(\tau_j)\| \tag{76}$$

$$\|F_j(\tau_j)\| \leq \left\| \exp\left(\tau_j \begin{bmatrix} \|A\| & \|B\| & 0 & 0 & \dots & 0 \\ 0 & 2\|A\| & 2\|B\| & 0 & \dots & 0 \\ 0 & 0 & 3\|A\| & 3\|B\| & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & m_j\|A\| \end{bmatrix} \right) \right\| \tag{77}$$

Then, as our last matrix is strictly positive, we may bound the norm of the matrix exponential by the norm of the exponential of the norm of the largest row, thus,

$$\|F_j(\tau_j)\| \leq e^{\tau_j m_j(\|A\|+\|B\|)}. \tag{78}$$

As $\hat{t}_j \leq \frac{1}{jc}$, $m_j \leq j(\log_{1+\delta}(\frac{1}{\epsilon}) + \log_{1+\delta}(j))$ and $c \leq \frac{\|A\|+\|B\|}{2}$, the conclusion follows. □

We note (see Figure 2) that our matrix $F_{j+1}[m_j, m_{j+1}]$ may be represented as

$$e^{G_{j+1}\tau_j} - e^{(G_{j+1}-G_j)\tau_j} + I. \tag{79}$$

We may then describe our evolution matrix as:

$$\sum_{k=0}^{\infty} \frac{(G_{j+1}^k - (G_{j+1}-G_j)^k)\tau_j^k}{k!} + I = \sum_{k=0}^{K} \frac{(G_{j+1}^k - (G_{j+1}-G_j)^k)\tau_j^k}{k!} + I + R_K \tag{80}$$

Next, note that $\|G_{j+1}\| \leq \max_j \|G_j\|$ and by triangle ,inequality. $\|G_{j+1} - G_j\| \leq 2\max_j \|G_j\|$. We then have

$$\|R_k\| \leq \sum_{k=K+1}^{\infty} \frac{(3\max_j \|G_j\|\tau_j)^k}{k!} \in O\left(\frac{(\max_j \|G_j\|\tau_j)^K}{K!}\right). \tag{81}$$

After the use of Stirling's inequality, we obtain that

$$\|R_k\| \in O\left(\frac{(e\max_j \|G_j\|\tau_j)^K}{K^K}\right). \tag{82}$$

Solving for $\|R_k\| = \epsilon^*$ for some desired error target ϵ^* yields a Lambert W function whose asymptotic form then reveals that it suffices to choose a value of K such that

$$K = O\left(\frac{\log(\max_j \|G_j\|\tau_j/\epsilon^*)}{\log\log(\max_j \|G_j\|\tau_j/\epsilon^*)}\right), \tag{83}$$

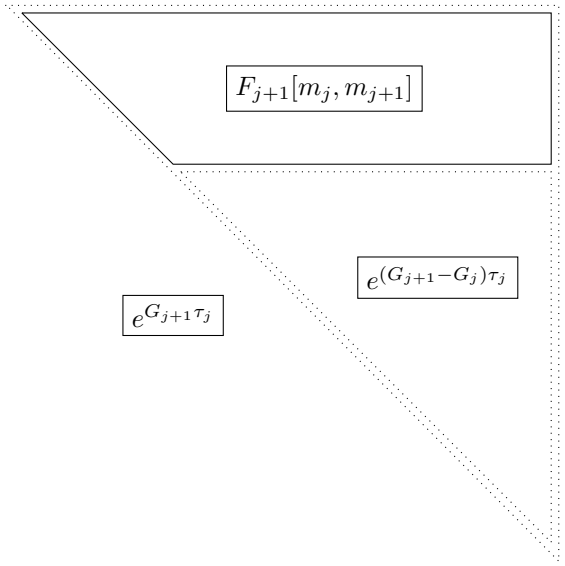


Figure 2: Computing $F_{j+1}[m_j, m_{j+1}]$ from $e^{G_{j+1}\tau_j}$ and $e^{(G_{j+1}-G_j)\tau_j}$

to obtain a ϵ^* approximation of our matrix.

Next we will represent our G_j and $G_{j+1} - G_j$ as sums of unitaries of the form: $\sum_{l=0}^{L} \alpha_l U_l$. We note that A and B may be decomposed into $O(d_A^2|A|\log N)$ unitaries and $O(d_B^2|B|\log N)$ unitaries respectively. This is implemented using edge coloring edge-coloring (where here $N = 2^n$ is our dimension; see [4]), and each unitary may be implemented using $O(1)$ copies of O_A or O_B. Then, as it's easy to construct G_j and $G_{j+1} - G_j$ from A and B using our definition, we can see that the L we require may be bounded by $O(m_j(d_A^2|A| + d_B^2|B|))$. Now; the sum of unitaries may be implemented using the well-known Linear Combination of Unitaries (LCU) results; we will use here a robust version from [5]:

Lemma 12 (Lemma 4 of [5])**.** *Let $\vec{U} = (U_1, ..., U_M)$ be unitary operations and let $V = \sum_{m=1}^{M} a_m U_m$ be $\bar{\delta}$-close to a unitary. Let $a := \sum_{m=1}^{M} |a_m|$. We can approximate V to within $O(\bar{\delta})$ using $O(a)$ select U and select $\bar{U}$ operations.*

We may now proceed with our proof of Theorem 2, which we restate below for convenience.

Theorem 2. *Suppose we have for bounded matrices A and B an equation of the form:*

$$\frac{d\,|u\rangle}{dt} = A\,|u\rangle + B\,|u\rangle^{\otimes 2}$$

Where A is $d_{\|A\|}-$sparse and B is $d_{\|B\|}-$sparse, and the dimension of u is bounded by: $\dim(|u\rangle) = 2^n$. Then there exists a quantum algorithm simulating $u(T)$ for a time $T > 0$ within error E with respect to the Euclidean norm for some arbitrarily small constant δ' with at most

$$O\left(\frac{e^{2Tc(1+\delta')}(d_A^2\|A\| + d_B^2\|B\|)\log(1/E)n}{\log\log(1/E)}\right)$$

oracle queries, where c is in $O(\|B\|)$.

Proof of Theorem 2. We note here that our select operations may be implemented using $O(M)$ calls to our oracles. The total evolution cost for our LCU is then:

$$O(M \times a) = O(LK \times 1) = O\left(\frac{m_j(d_A^2\|A\| + d_B^2\|B\|)\log(1/\epsilon^*)n}{\log\log(1/\epsilon^*)}\right). \tag{84}$$

It remains to establish our choices of ϵ^* produced by the approximate simulation of our $F_{j+1}|m_j$, as well as the ϵ_j distance between the simulated quasi-unitaries and the nonlinear dynamics, to ensure that our total error remains small throughout the evolution.

Define a new variable $v_{sim}(j)$ to represent our simulated value of v_j at time t_j. Within a given $\hat{t}_j$, we have $2(\log_{1+\delta}\left(\frac{1}{\epsilon}\right) + \log_{1+\delta}(j))$ substeps. Let

$$\epsilon^* = \frac{\epsilon}{j\delta^2}. \tag{85}$$

By triangle inequality, we are then $\frac{2\epsilon}{j\delta^2}$ close to a unitary. Lemma 12 will then allow us to simulate approximation of $F_{j+1}|m_j$ within $O(\frac{\epsilon}{j\delta^2})$. Given that our error accumulates approximately linearly when small, our simulation will have that

$$|v_{sim}(j) - u(j)| = O\left(\frac{(\log_{1+\delta}\left(\frac{1}{\epsilon}\right) + \log_{1+\delta}(j))\epsilon}{j\delta^2}\right) + |v_{sim}(j+1) - u(j+1)|. \tag{86}$$

The error from the entire evolution may then be described as

$$|v_{sim}(0) - u(0)| = O\left(\sum_{j=1}^{w} \frac{(\log_{1+\delta}\left(\frac{1}{\epsilon}\right) + \log_{1+\delta}(j))\epsilon}{j\delta^2}\right), \tag{87}$$

$$|v_{sim}(0) - u(0)| = O\left(\frac{T^2\epsilon}{\delta^2}\right). \tag{88}$$

For a desired error of E, it is then sufficient to set

$$\epsilon = O\left(\frac{E\delta^2}{T^2}\right). \tag{89}$$

Our total evolution cost will be over all values of j. Taking our value of w, the cost of step j from equation 84 and 85; we obtain that the total number of queries required by our algorithm is

$$O\left(\sum_{j=1}^{w=e^{Tc(1+\delta)}} \frac{j\log_{1+\delta}(\frac{jT^2}{E\delta^2})(d_A^2\|A\| + d_B^2\|B\|)\log(T^2j/E)n}{\log\log(T^2j/E)}\right) \tag{90}$$

$$= O\left(\frac{e^{2Tc(1+\delta')}(d_A^2\|A\| + d_B^2\|B\|)\log(1/E)n}{\log\log(1/E)}\right), \tag{91}$$

where

$$c \leq \max\{\|A\|, 12.7\|B\|\}, \tag{92}$$

and $\delta' > \delta$ remains an arbitrarily small constant. This concludes our proof of Theorem 2. □

We have provided a quantum algorithm for our initial problem of solving a nonlinear differential equation as described in equation 4; with a scaling in time and in the norm of our nonlinear term that is similar to our lower bound from Lemma 1, and logarithmic in the inverse of the error. We note however that what we obtain is not equivalent to a classical vector from which we may directly read values. For a fair comparison with our classical algorithm, we will thus compute the expected value of our output vector.

Corollary 13. *Suppose we have an equation of the form*

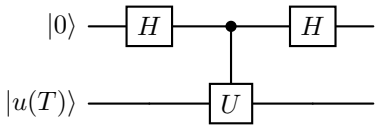
$$\frac{d\,|u\rangle}{dt} = A\,|u\rangle + B\,|u\rangle^{\otimes 2}$$

where $A \in \mathbb{C}^{2^n \times 2^n}$, $B \in \mathbb{C}^{2^n \times 2^{2n}}$ are $d-$sparse complex matrices. Let U be some unitary matrix of matching dimension which may be queried in time $O(1)$. Then there exists a quantum algorithm computing the expectation value $\langle u(T)|\,U\,|u(T)\rangle$ with probability greater than or equal to $2/3$ for a time $T > 0$ within error ϵ using at most

$$\tilde{O}\left(\frac{e^{Tc}d^2\log(\frac{1}{\epsilon})}{\epsilon}\right)$$

oracle queries where $c \in O(\|A\| + \|B\|)$ from (92).

Proof. We may assume without loss of generality that our inputs are real-valued and that we operate strictly within the reals (see the introduction for the necessary mapping). We will make use of the Hadamard test to obtain an estimate of $\langle u(T)| U |u(T)\rangle$. We add an qubit $|0\rangle$, then the circuit



creates the state

$$|0\rangle |u\rangle \to \frac{1}{\sqrt{2}}(|0\rangle |u\rangle + |1\rangle |u\rangle) \to (|0\rangle |u\rangle + |1\rangle U |u\rangle) \to \frac{1}{2}(|0\rangle (|u\rangle + U |u\rangle) + |1\rangle (|u\rangle - U |u\rangle)) \quad (93)$$

The probability of measuring the first qubit to be 0 is

$$\frac{(|u\rangle + U |u\rangle)^\dagger (|u\rangle + U |u\rangle)}{4} = \frac{\langle u| |u\rangle + \langle u| U |u\rangle + \langle u| U^\dagger |u\rangle + \langle u| UU^\dagger |u\rangle}{4} \quad (94)$$

$$= \frac{1 + (\langle u| U |u\rangle)^* + (\langle u| U |u\rangle) + 1}{4} \quad (95)$$

$$= \frac{1 + \mathrm{Re}(\langle u| U |u\rangle)}{2}. \quad (96)$$

The above Hadamard test circuit can be modified to yield the expectation value of the imaginary part can be estimated by simply applying a $S^\dagger$ gate to the control qubit before U is applied. Given that $\langle u| U |u\rangle$ will be strictly real, this is equivalent to $\frac{1}{2}(1 + \langle u| U |u\rangle)$. We may employ the *amplitude estimation* heuristic from [8] find a such that $|a - \frac{1}{2}(1 + \langle u| U |u\rangle)| \leq \frac{\epsilon'}{2}$ with probability of failure less than 1/6. Repeating this for the imaginary part yields from the union bound a probability of failure of at most 1/3. These two processes combined require using at most $O(\frac{1}{\epsilon}')$ iterations of our circuit. Multiplying our result by 2 and removing 1 then gives us our desired estimate of $\langle u(T)| U |u(T)\rangle$ within error ϵ'.

The amplitude estimation process described above requires a reflection about the state $|u(T)\rangle$. This reflection can be implemented using at most a single query to an oracle that prepares $|u(T)\rangle$ from $|0\rangle$ and the inverse of this oracle. The result of Theorem 2 provides us with an approximate necessary oracle to $|u'(T)\rangle$ such that $\| |u'(T)\rangle - |u(T)\rangle \| \leq \epsilon''$ using at most

$$N_{\text{queries}} \in O\left(\frac{e^{2Tc(1+\delta')}(d_A^2\|A\| + d_B^2\|B\|)\log(1/\epsilon'')n}{\log\log(1/\epsilon'')}\right) \quad (97)$$

queries to our oracles O_u, O_A and O_B. Setting $\epsilon' = \epsilon/2, \epsilon'' = \epsilon/2$ our result follows.

□

The optimal scaling that can be achieved for simulation of non-linear differential equations in time is given by Lemma 1. Thus the quantum algorithm is near-optimal in general.

4.2 Continuity with Hamiltonian Simulation

We will notice that our result does not appear to demonstrate the expected behavior for a small nonlinearity. Indeed, our algorithm would result in exponential time scaling, even when our non-linearity is set to 0. In this section, we will establish that we may indeed find good algorithms when $\|B\|$ is sufficiently small.

Assume that $\|B\| < \frac{1}{10e^{\|A\|T}T}$. Then, we propose to separate our time steps into s *equal* slices $t_j < \frac{1}{\|A\|}$, and set each k_j to k. As per the previous definitions, $m_j = jk$ in this definition.

We remind ourselves of the following equation from 56, 57,

$$\|F_{m,m+k}(t)\| \leq \left(\frac{e(m+k-1)}{k}\right)^k \frac{\|B\|^k}{\|A\|^k} e^{mt\|A\|}(\|A\|\hat{t}_j \times (e-1))^k, \quad (98)$$

and we have then:

$$\frac{m_s t\|A\|}{k} = \frac{(sk)(\frac{T}{s})\|A\|}{k} = \|A\|T \tag{99}$$

$$\|F_{m_s,m_s+k}(t)\| \leq \left(\frac{es}{1+1/s}\frac{\|B\|}{\|A\|}e^{\|A\|T}\frac{T}{s}\|A\|(e-1)\right)^k \tag{100}$$

$$\leq \frac{1}{2^k}. \tag{101}$$

The error introduced at a given time step is bounded by the norm of what we discard. This will be the sum of the tail $\leq \sum_k^\infty \frac{1}{2^j} = \frac{1}{2^{k-1}}$ for our choice of k. We require $w = O(\|A\|T)$ time slices to fulfill the condition $t_j < \frac{1}{\|A\|}$. Let ${F'}_j$ be the matrix we apply at time step j, where F_j is the exact solution. Given an error of at most δ at each time step, and our steps are multiplicative, the global error introduced by our approximate Carleman linearization will be bounded by

$$\|({F'}_j\,|u(0)\rangle - F_j)\,|u(0)\rangle\,\| \leq (1+\delta)^w. \tag{102}$$

If we set $\delta \leq \frac{\epsilon}{2w^2}$, we may bound this by:

$$\|({F'}_j\,|u(0)\rangle - F_j)\,|u(0)\rangle\,\| \leq 2w\delta \leq \epsilon. \tag{103}$$

Thus, setting $k = O(\log(\frac{1}{2\epsilon w^2})$, we may obtain an error of at most ϵ. As each time step requires k Carleman blocks, we require a Carleman matrix with $kw = O(\frac{\|A\|T\log(\|A\|\|T\|}{\log(\epsilon)})$ blocks, which will have a norm of at most $O(\frac{\|A\|^2T^2\log(\|A\|\|T\|}{\log(\epsilon)})$ and sparsity $O((d_A+d_B)\frac{\|A\|T\log(\|A\|\|T\|}{\log(\epsilon)})$, as these both grow linearly with the number of Carleman blocks. We may then proceed using standard Hamiltonian simulation techniques to simulate our approximate Hamiltonian with a linear dependence on norm and sparsity of our approximate Hamiltonian, such as, for example, in [5].

4.3 Dequantized Algorithm

A question remains about the gap between randomized classical algorithms and quantum algorithms. We aim to address this now by replicating the process for classical computation and analyzing the scaling difference we can expect. Difficulties arise in the conversion of quantum states into a classical setting; we cannot efficiently represent the Hilbert space in which the states reside. Upon measurement, we obtain an n-dimensional vector; hence our objective will be to find its expected value.

We cannot give our classical algorithm any direct access to the input as this would be too "powerful"; allowing us to circumvent the lower bounds set by Lemma 1, which relies on the difficulty of state discrimination. We will instead allow our classical algorithm to sample from the input. Given an input state:

$$|\phi\rangle = \sum_j e^{ik_j}c_j\,|j\rangle\,, \tag{104}$$

such that $0 \leq k_j \leq 2\pi$, $c_j \geq 0$ and $\sum_j c_j^2 = 1$, let our classical oracle O_c output the pair $\{j, k_j\}$ with probability c_j^2. We note here that this is still not an ideal comparison to the oracle provided in the context of our quantum algorithm; as it solves the phase estimation problem for free. However, it is sufficient for our lower bound to apply; indeed, in the lower bound arguments introduced in Section 3, the states we wish to distinguish are oriented vertically on the Grover sphere initially, and thus the phase being provided does not make the problem easier. The reader may notice here that equivalently, we may come up with a scheme in which the c_j are directly provided but the k_j are obfuscated, for which the lower bounds would still apply; access to a combination of the two is required for the bound to be broken.

In the context of standard Hamiltonian Simulation, the quantum advantage that we may obtain for certain types of Hermitian operators often relies on a large Hilbert space. The Path Integral

Monte-Carlo method may be used to circumvent these constraints in a probabilistic manner, but it generally scales poorly in other variables, such as the sparsity of the operator, the norm of our Hermitian operators, and/or evolution time.

Determining the exact relation between quantum and classical scaling does not follow clearly for the quadratic differential equation problem. The exponential complexity in time is distinct from how we generally compare quantum and classical simulation; as previously stated, the bound does not exist in the context that classical algorithms are usually defined (access to a deterministic oracle to the input); which would give us the false impression that the classical model has an advantage.

While the partially probabilistic oracle introduced above solves this issue, we note that the Carleman Linearization technique we proposed for the Quantum case creates a linear operator with sparsity and norm scaling exponentially with the time constraint; simple bounds on the path integral Monte Carlo method on such a linear operator would lead to super-exponential scaling in time. Alternative methods may thus be necessary.

First, in order to understand the difference between the performance of quantum and classical algorithms for non-linear differential equations more clearly we need to understand the relationship between the best time scaling achievable for quantum and classical methods. To this end, we provide a classical method that uses sampling that retains many of the same features as the quantum case including saturating the optimal time scaling of the quantum algorithm, although it may scale poorly in other variables in generality. We provide the argument below.

Lemma 14. *Suppose we have for bounded matrices A and B an equation of the form:*

$$\frac{d\left|u\right\rangle}{dt} = A\left|u\right\rangle + B\left|u\right\rangle^{\otimes 2}$$

Where A and B are $d-$sparse, $|u\rangle$ is a norm 1 vector in $\mathbb{C}^{2^n}$ and the assumption is made that the linear transformation $V(t)u\,|u\rangle$ on $|u\rangle$ remains unitary at all times t. Then there exists a classical algorithm which outputs $v(j)$ such that:

$$\|v(j) - \langle u(0)|\, V(T)\, |u(0)\rangle\,\| \leq \epsilon$$

for a time $T > 0$ with respect to the Euclidean norm, using at most

$$O\left(\frac{d2^n T e^{2(\|A\|+\|B\|)T})}{\epsilon^2}\right)$$

queries to classical oracles O_A, O_B such that $O_A(i,j)$ yields the value of the i^{th} non-zero matrix element in row j and similarly $O_B(i,j)$ yields the corresponding matrix element in row j of B and oracles f_A, f_B such that $f_A(x) = y$ if and only if A_{xy} is the i^{th} non-zero matrix element in row x and f_B yields the corresponding locations of the non-zero matrix elements of B.

Proof. Our proof for this claim immediately follows from using the Euler method to approximate the solution to the non-linear differential equation. Specifically, if T/h steps are used in the numerical solution to the differential equation then the error in the solution is [16]

$$|u(t_i) - u(T)| \leq \frac{hMe^{LT}}{2L}, \tag{105}$$

where $M = \max|\ddot{u}(t)|$ and $L = \max_t|\frac{d}{du}(Au + Bu^{\otimes 2})|$. As $u(t)$ has a constant norm, this gives us bounds on both M and L of $M, L \leq \|A\| + 2\|B\|$. Let $u(t_i)$ be the Euler approximation at the i-th step. Thus, setting $h = \frac{4\epsilon/2}{e^{(|A|+2|B|)T}}$, we obtain:

$$|u(t_h) - u(T)| \leq \frac{\frac{4\epsilon^2/2}{e^{(|A|+2|B|)T}}(\|A\| + 2\|B\|)e^{(\|A\|+2\|B\|)T}}{2(\|A\| + 2\|B\|)} \tag{106}$$

$$\leq \frac{\epsilon}{2}. \tag{107}$$

Now, we must account for our initial error. Given $\|\tilde{u}(0) - u(0)\| \leq \delta$, we may represent then $\tilde{u}(0) = u(0) + r(0)$ where the remainder $r(t)$ is defined to be the difference between $\tilde{u}(t)$ and $u(t)$.

We know that $\frac{dr}{dt}$ is bounded by $Lr(t)$ by our definition of L as the maximum derivative in u; thus, we have the following differential equation:

$$\frac{dr}{dt} \leq Lr(t), \quad r(0) = \delta. \tag{108}$$

Integrating the above bound on the derivative we obtain

$$r(T) \leq \delta e^{LT}. \tag{109}$$

Noting here that $L \leq \|A\|+2\|B\|$ and setting here $\delta = \frac{\epsilon}{2e^{(\|A\|+2\|B\|)T}}$ we obtain that $\|u'(0)-u(0)\| \leq \frac{\epsilon}{2}$. From Equation 106 and Equation 109 we then have

$$\|\tilde{u}(t_h)-u(T)\| \leq \|\tilde{u}(t_h)-\tilde{u}(T)\|+\|\tilde{u}(T)-u(T)\| = \|\tilde{u}(t_h)-\tilde{u}(T)\|+\|r(T)\| \leq \epsilon/2+\epsilon/2 = \epsilon \tag{110}$$

Which is the desired precision bound.

To achieve our initial error of $\delta = \frac{\epsilon}{2e^{(\|A\|+2\|B\|)T}}$, we will need $O(\frac{e^{2(|A|+2|B|)T}}{\epsilon^2})$ queries. Performing 1 Euler step by directly querying all values will be $O(d2^n)$ queries, and we require T/h Euler steps. This leads to the total scaling of $O(\frac{d2^n T e^{2(|A|+|B|)T}}{\epsilon^2})$ queries for the problem. □

This close to matches our quantum algorithm, and quantum lower bound in its time scaling. The error scaling is quadratically worse than our quantum algorithm, when an appropriate comparison is made (see Corollary 13). However, the scaling in the dimension is potentially exponentially worse. This is the result of having to compute $f(t, u(t) = Au(t) + Bu(t)^{\otimes 2}$ for each of our iterations.

We may note here, however, that our initial vector only has $O(\frac{e^{2(|A|+2|B|)T}}{\epsilon^2})$ non-zero entries. While this would result in superexponential scaling in generality (as the number of non-zero elements will be multiplied by up to $(d_A + d_B + 1)$ at each Euler step), there are cases for which the growth is more reasonable. The simplest case is such that A is diagonal and B is a diagonal block matrix with each block $B_{j,j}$ being a length n vector, and $B_{j,k} = 0$ for $k \neq j$. In this case, the number of non-zero entries remains fixed at each Euler step. More broadly, we will be able to define the following class of equations which we can more efficiently solve in a classical setting.

Definition 15 (Geometrically Local Equation)**.** *Given an equation $\frac{du}{dt} = f(u)$ such that $u \in \mathbb{C}^{2^n}$ and $f : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$. Let $|x\rangle$,$|y\rangle$ be elementary basis vectors in $\mathbb{C}^{2^n}$.*

1. *We say that $|x\rangle$, $|y\rangle$* interact *if there exists $u \in \mathbb{C}^{2^n}$, $c \in \mathbb{C}$ such that $\langle x| f(|u\rangle) \neq \langle x| f(|u\rangle + |y\rangle)$.*
2. *We say that the equation is* geometrically k-local in a graph G *if we may map all basis vectors in $\mathbb{C}^{2^n}$ onto the vertex set of a graph G in such a way that for two basis vectors $|x\rangle$ and $|y\rangle$ that do not interact, the graph distance $d(x, y)$ is strictly greater than k.*

We may note here that this is a different definition to that of a geometrically local Hamiltonian, from the quantum setting. We borrow the terminology as it shares the intuition of geometric locality.

Definition 15 will allow us to bound the number of elements we need to consider in the Euler method. Specifically, given $\tilde{u}(t_{y+1}) = \tilde{u}(t_y) + (T/h)f(\tilde{u}(t_y))$, as per the Euler method. We may see that if $\tilde{u}_j(t_{y+1})$ is non-zero, then there must be some k interacting with j such that $\tilde{u}_k(t_y)$ is non-zero.

Theorem 16. *Suppose we have for bounded matrices A and B an equation of the form:*

$$\frac{d\,|u\rangle}{dt} = A\,|u\rangle + B\,|u\rangle^{\otimes 2},$$

where $|u\rangle$ is a norm 1 vector in $\mathbb{C}^{2^n}$ and the assumption is made that $|u(t)\rangle$ remains a unit vector at all times t. Assume further that our equation is geometrically $k-$local in $\mathbb{Z}^D$. Then there exists a classical algorithm which outputs $v(j)$ in $\mathbb{C}^{2^n}$ such that:

$$\|v(j) - |u(T)\rangle\| \leq \epsilon$$

for a time $T > 0$ with respect to the Euclidean norm, using at most

$$O\left(\frac{(kT)^{D+1}e^{2(D+1)(|A|+2|B|)T}}{(2\epsilon)^{D+2}}\right)$$

queries to classical oracles O_A, O_B such that $O_A(i,j)$ yields the value of the i^{th} non-zero matrix element in row j and similarly $O_B(i,j)$ yields the corresponding matrix element in row j of B and oracles f_A, f_B such that $f_A(x) = y$ if and only if A_{xy} is the i^{th} non-zero matrix element in row x and f_B yields the corresponding locations of the non-zero matrix elements of B.

Proof. The method is identical to 14, we reevaluate the cost of performing our Euler steps. As each Euler iteration is written as $v(j) + f(v(j), t)$, where $f(u(j)) = |u\rangle + B|u\rangle^{\otimes 2}$, all non-zero elements of $v(0)$ may be a distance of at most jk from a non-zero element in $v(j)$. Our initial vector has at most $O(\frac{e^{2(|A|+2|B|)T}}{\epsilon^2})$ non-zero values. Given that $f(u) = A|u\rangle + B|u\rangle^{\otimes 2}$ is geometrically local in $\mathbb{Z}^D$, the number of non-zero values of $u(j)$ will thus be bound by the following equation.

$$O(\frac{e^{2(|A|+2|B|)T}}{\epsilon^2}) \times (jk)^D \tag{111}$$

Now, summing over all $m = T/h$ Euler steps,

$$\sum_{j=1}^{m} O(\frac{e^{2(|A|+2|B|)T}}{\epsilon^2}) \times (jk)^D \tag{112}$$

$$= O(\frac{e^{2(|A|+2|B|)T}}{\epsilon^2}) \times k^D \times \sum_{j=1}^{Om} j^D \tag{113}$$

$$= O(\frac{e^{2(|A|+2|B|)T}}{\epsilon^2}) \times k^D \times O(m^D) \tag{114}$$

Noting here that $m \in O(T\frac{e^{(|A|+2|B|)T}}{4\epsilon/2})$:

$$= O\left(\frac{(kT)^D e^{2(D+1)(|A|+2|B|)T}}{(2\epsilon)^{D+2}}\right) \tag{115}$$

We note that we need $O(k)$ calls to our oracle O_A and O_B for each non-zero vector value in $u(j)$. Further, the cost of computing our Euler steps here strictly exceeds the cost of creating an initial vector of sufficient precision. The result follows. □

A notable set of equations that are geometrically $k - local$ in $\mathbb{Z}^D$ are band matrices (allowing D bands of width up to k in A and width up to $2^n(k-1)$ in B).

We will now attempt to write a classical algorithm that will scale polynomially with n in generality, by imitating the methods used for our quantum algorithm. As we will see, this leads to even worse scaling with T.

Part of our arguments will depend on the variation in the probabilistic approximation of our target evolution; for this, we will require a numerical value that we can approximate. We will use here the expected value as defined at the end of section 4.1; $\langle u(T)|\, U\, |u(T)\rangle$, for some unitary U.

Our algorithm will produce a random uniform $P(V = v_j) = P(V = v_k)$ for any choice of j, k, where $\sum_j v_j \approx |u(T)\rangle$. We will sample from this at random. A trap to avoid here is that randomly sampling $v_j^\dagger U v_j$ will not average out at $\langle u(t)|\, U\, |u(t)\rangle$; instead, we will need to randomly sample among $v_j^\dagger U v_k$. This introduces a quadratic factor that we will accept. Call this random variable V, and we will let its variation be $\mathbb{V}(V)$. We won't precisely calculate this variance, but instead use it as one of the variables bounding our running time; with a discussion on how large this value could be after the proof.

Theorem 17. *Suppose we have for bounded matrices A and B an equation of the form:*

$$\frac{d|u\rangle}{dt} = A|u\rangle + B|u\rangle^{\otimes 2}$$

Where A is d_A-sparse, B is $d_B-sparse$, $|u\rangle$ is a norm 1 vector in $\mathbb{C}^{2^n}$ and the assumption is made that $|u(t)\rangle$ remains a unit vector at all times t. Suppose that the decomposition of our Carleman block matrix for this problem generates a variance of

$$\mathbb{V}(V)$$

. Given some Unitary matrix U; there exists an algorithm which outputs an approximation $v(T)$ of the expectation value such that

$$|v(T) - \langle u(T)| U |u(T)\rangle| \leq \epsilon$$

with probability at least $2/3$ for a time $T > 0$ using at most

$$O\left(\frac{\mathbb{V}(V)(d_A^2 + d_B^2)e^{O(T(\|A\|+\|B\|)(1+\delta))}}{\epsilon^2}\right)$$

queries to our classical oracles O_A, O_B and O_u.

Proof. Through graph coloring, we have access to a $O(d^2 \log(N))$ decomposition of a size N d-sparse hermitian matrix. This will hold for both A and the extension of B into a pair of Hermitian matrices

$$B^+ = \begin{bmatrix} 0 & B \\ B^\dagger & 0 \end{bmatrix} \tag{116}$$

$$B^- = \begin{bmatrix} 0 & B \\ -B^\dagger & 0 \end{bmatrix} \tag{117}$$

This directly induces a $d^2 m \log(N)$-sparse decomposition for A_m and $B_m^{+/-}$, from their respective definitions. Our Carleman Block, with a slight abuse of notation placing each block into its correct position, will be described by

$$G = \sum_{j=1}^{w} A_j + \frac{1}{2}B_j^+ + \frac{1}{2}B_j^-. \tag{118}$$

We note that the set of all even j and the set of all odd j act independently; thus, expanding into our $d^2 m \log(N)$-sparse decomposition, this may be expressed as 1 sum of $wd^2 \log(N)$ matrices $\sum_{j=1}^{wd^2 \log(N)} H_j$. Using a trotter formula, our evolution may then be approximated by:

$$e^{tG} = \left(\prod_{j=1}^{wd^2 \log(N)} e^{\frac{t}{r}H_j}\right)^r + O(t^2). \tag{119}$$

This allows us to reduce our error linearly with r. We will set $r = O(\frac{T}{\epsilon})$ to ensure our error remains sufficiently small. Note that:

$$e^{I^{\otimes j} \otimes H \otimes I^{\otimes k}} = I^{\otimes j} \otimes e^H \otimes I^{\otimes j}. \tag{120}$$

Thus, we may effectively approximate e^{A_m}, $e^{B_m^+}$ and $e^{B_m^-}$. For any 1-sparse hermitian H, e^H will be 2-sparse. Ignoring zero-valued terms, our product will take the form

$$e^{tG} = \prod_{k=1}^{p} (N_1(k) + N_2(k)), \tag{121}$$

for some appropriate p. $N_1(k)$ and $N_2(k)$ each specify a unique neighbor for us to "continue" with, given a previous position. We will now take our sum outside of the product to arrive at the following equation,

$$e^{tG} = \sum_{l_1=0}^{wd} \sum_{l_2 \in N(l_1)} \sum_{l_3 \in N(l_2)} \cdots \sum_{l_p \in N(l_{p-1})} \prod_{k=1}^{p} N_{l_i}(k), \tag{122}$$

where we here slightly abuse notation, $l_s \in N(l_{s-1})$ indicating here the set of l with existing $N_{l_i}(k)$ overlapping with the previous set. We call the elements of our sum "paths". Intuitively, our path "travels" through our vector space; with a starting coordinate. The elements of the product describe how the path moves (from coordinate a to coordinate b) in a given time splice of our evolution.

Now; computing this exactly will not be possible, due to the impractically large number of possible paths; the objective is instead to approximate the expectation, using Montecarlo methods. Let V be the random variable of a randomly chosen path multiplied by the number of different paths, applied to our probability vector. Then,

$$\begin{aligned}\mathbb{V}\left(\frac{\sum_{k=1}^{s} V}{s}\right) &= \frac{\mathbb{V}(\sum_{k=1}^{s} V)}{s^2} = \frac{\sum_{k=1}^{s}\mathbb{V}(V)}{s^2} \\ &= \frac{\mathbb{V}(V)}{s}.\end{aligned} \tag{123}$$

Chebyshev's inequality then gives us for $\sigma^2 = \mathbb{V}\left(\frac{\sum_{k=1}^{s} V}{s}\right)$,

$$Pr(|V - \mathrm{E}[V]| \geq k\sigma) \leq \frac{1}{k^2}. \tag{124}$$

Given that $\mathrm{E}\left(\frac{\sum_{k=1}^{s} V}{s}\right) = \mathrm{E}[X]$, we will require $\sqrt{\frac{\mathbb{V}(v)}{s}} = O(\epsilon\sqrt{\delta})$ if we wish to have a probability of $1-\delta$ of being within ϵ of our answer. Thus; the number of samples we require will be $O(\mathbb{V}(V))$. So our result follows by taking $\delta = 1/3$. Taking the cost of individually simulating each path into consideration, we obtain a total cost of

$$O\left(\frac{\mathbb{V}(V) n (d_A^2 + d_B^2) e^{O(Tc(1+\delta))}}{\epsilon^2}\right). \tag{125}$$

This completes our proof of theorem 17 after noting $c \in O(\|A\| + \|B\|)$. □

Defining exact bounds on the variance of the may be difficult, depending strongly on both the exact problem we are working with and the definition of the matrix on which we are defining our expectation value. Generic bounds, using the maximum norm of any path of 1 will be super-exponential in T. Indeed, the number of possible paths we have is super-exponential in T as a result of the exponential sparsity of the largest Carleman block in T. However, it should be evident that such bounds vastly overestimate the variance we obtain in practice. In the case of small B, as introduced in section 4.2, we may find a variance that is exponential in Tc, allowing more reasonable time scaling. Different approaches may however be necessary to obtain optimal time and dimension scaling for the classical case in generality.

5 Conclusion

We have studied nonlinear differential equations with unitary solutions. We have found a quantum algorithm solving such equations with $O(e^{cT}\log\frac{1}{\epsilon})$ oracle queries, for some constant c dependent on the problem. Further we give classical algorithms that may either match the time scaling, or dimension scaling of our quantum algorithm, though not both in generality. The arguments from Section 3 show that any algorithm solving such a problem is necessarily exponential in its time complexity, thus, our algorithms approach the known lower bounds for the problem in time complexity.

Our solutions still offer avenues for significant improvement. Notably, the analysis of the generalized classical algorithm is unlikely to be tight without an in-depth analysis of the variance, and the error scaling can likely be improved. A more suitable classical algorithm may indeed bridge the time-scaling gap with our quantum algorithm. Further, our quantum algorithm could potentially be improved by recent improvements in Hamiltonian simulation, such as the Hamiltonian simulation by qubitization algorithm presented in [25]. These could offer a reduction in the factor

of our time exponent if successfully implemented to simulate our linear approximation. The particular formulation of the near-hermitian matrix we obtain for our problem however makes such an implementation non-trivial.

Establishing a tighter lower bound on the complexity of simulating unitary nonlinear differential equations in terms of both the error and the evolution time would also be an important contribution. This would allow a more direct comparison of the efficiency of our algorithm with complexity theoretic bounds.

One may want to ask if there exist any families of genuinely nonlinear differential equations under which a quantum algorithm is provably capable of providing exponential speedup over all possible classical algorithms. Given that the linear Hamiltonian simulation problem is a special case of the unitary polynomial equations that we studied, trivially, we may declare the statement to be true; but such an analysis does not demonstrate the capability of quantum computers to handle non-linearity. The work of [22] and [20] provide quantum algorithms capable of solving *dissipative* quadratic differential equations in polynomial time under certain restrictions. However, the argument for a quantum advantage of the algorithm from [22] over a classical counterpart, once again, uses the special case where the norm of the nonlinearity is set to 0.

It remains an interesting open problem whether there exists any family of strictly nonlinear differential equations in which a quantum algorithm may provide a provable exponential advantage over any classical algorithm. Ideally, it would be interesting to either:

1. Demonstrate that a quantum algorithm may offer no exponential advantage over comparable classical counterparts for any family of strictly quadratic nonlinear differential equations (meaning that $\frac{du}{dt} = F_2 u^{\otimes 2}$ for a rectangular matrix F_2).

2. Provide a classical lower bound on the running time of a family of strictly quadratic nonlinear differential equations where an exponentially faster quantum algorithm exists.

Providing clear evidence of such a family would not only improve our understanding of the gap between quantum and classical algorithms for non-linear differential equations but also provide us with a crucial understanding of the extent to which quantum computers will be capable of helping us understand non-linear dynamical systems.

Acknowledgements

N.B. is funded by the Department of Computer Science of the University of Toronto, the Ontario Graduate scholarship Program as well as NSERC. N.W. acknowledges funding from Google Inc. This material is based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-design Center for Quantum Advantage (C2QA) under contract number DE-SC0012704 (PNNL FWP 76274).

References

[1] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342, 2011. DOI: 10.1145/1993636.19936.

[2] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, 2014. DOI: 10.1088/1751-8113/47/10/105301. URL https://doi.org/10.1088%2F1751-8113%2F47%2F10%2F105301.

[3] Dominic W. Berry and Pedro C. S. Costa. Quantum algorithm for time-dependent differential equations using dyson series. *Quantum*, 8:1369, June 2024. ISSN 2521-327X. DOI: 10.22331/q-2024-06-13-1369. URL http://dx.doi.org/10.22331/q-2024-06-13-1369.

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

[5] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809, 2015. DOI: 10.1109/FOCS.2015.54.

[6] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, oct 2015. DOI: 10.1109/focs.2015.54. URL https://doi.org/10.1109%2Ffocs.2015.54.

[7] Dominic W. Berry, Andrew M. Childs, Aaron Ostrander, and Guoming Wang. Quantum algorithm for linear differential equations with exponentially improved dependence on precision. *Communications in Mathematical Physics*, 356(3):1057–1081, 2017. DOI: 10.1007/s00220-017-3002-y. URL https://doi.org/10.1007%2Fs00220-017-3002-y.

[8] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002. DOI: 10.48550/arXiv.quant-ph/0005055.

[9] Andrew M. Childs and Joshua Young. Optimal state discrimination and unstructured search in nonlinear quantum mechanics. *Phys. Rev. A*, 93:022314, 2016. DOI: 10.1103/PhysRevA.93.022314. URL https://link.aps.org/doi/10.1103/PhysRevA.93.022314.

[10] I. Y. Dodin and E. A. Startsev. On applications of quantum computing to plasma simulations. *arXiv:2005.14369*, 2021. DOI: 10.48550/arXiv.2005.14369.

[11] Richard P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982. DOI: 10.1007/BF02650179.

[12] Marcelo Forets and Amaury Pouly. Explicit error bounds for Carleman linearization. *arXiv preprint arXiv:1711.02552*, 2017. DOI: 10.48550/arXiv.1711.02552.

[13] Eugene P Gross. Structure of a quantized vortex in boson systems. *Il Nuovo Cimento (1955-1965)*, 20(3):454–477, 1961. DOI: 10.1007/BF02731494.

[14] B L Hammond, W A Lester, and P J Reynolds. *Monte Carlo Methods in Ab Initio Quantum Chemistry*. World Scientific, 1994. DOI: 10.1142/1170. URL https://www.worldscientific.com/doi/abs/10.1142/1170.

[15] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), 2009. DOI: 10.1103/physrevlett.103.150502. URL https://doi.org/10.1103%2Fphysrevlett.103.150502.

[16] Francis Begnaud Hildebrand. *Introduction to numerical analysis*. Courier Corporation, 1987.

[17] Ilon Joseph. Koopman–von neumann approach to quantum simulation of nonlinear classical dynamics. *Physical Review Research*, 2(4), 2020. DOI: 10.1103/physrevresearch.2.043102. URL https://doi.org/10.1103%2Fphysrevresearch.2.043102.

[18] M G Krein Ju L Daleckii. *Stability of solutions of differential equations in Banach space*. American Mathematical Society, 1974.

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

[20] Hari Krovi. Improved quantum algorithms for linear and nonlinear differential equations. *Quantum*, 7:913, 2023. DOI: 10.48550/arxiv.2202.01054.

[21] Sarah K Leyton and Tobias J Osborne. A quantum algorithm to solve nonlinear differential equations. *arXiv preprint arXiv:0812.4423*, 2008. DOI: 10.48550/arXiv.0812.4423.

[22] Jin-Peng Liu, Herman Øie Kolden, Hari K Krovi, Nuno F Loureiro, Konstantina Trivisa, and Andrew M Childs. Efficient quantum algorithm for dissipative nonlinear differential equations. *Proceedings of the National Academy of Sciences*, 118(35):e2026805118, 2021. DOI: 10.1073/pnas.2026805118.

[23] Seth Lloyd, Giacomo De Palma, Can Gokler, Bobak Kiani, Zi-Wen Liu, Milad Marvian, Felix Tennie, and Tim Palmer. Quantum algorithm for nonlinear differential equations. *arXiv e-prints*, art. arXiv:2011.06571, November 2020. DOI: 10.48550/arXiv.2011.06571.

[24] Guang Hao Low and Isaac L. Chuang. Optimal hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, 2017. DOI: 10.1103/PhysRevLett.118.010501. URL https://link.aps.org/doi/10.1103/PhysRevLett.118.010501.

[25] Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, 2019. ISSN 2521-327X. DOI: 10.22331/q-2019-07-12-163. URL https://doi.org/10.22331/q-2019-07-12-163.

[26] Dorota Mozyrska and Zbigniew Bartosiewicz. On carleman linearization of linearly observable polynomial systems. *Mathematical Control Theory and Finance*, 01 2008. DOI: 10.1007/978-3-540-69532-5˙17.

[27] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*, volume 2. Cambridge university press Cambridge, 2001. DOI: 10.1017/CBO9780511976667.

[28] Yu Tanaka and Keisuke Fujii. A polynomial time quantum algorithm for exponentially large scale nonlinear differential equations via hamiltonian simulation, 2025. URL https://arxiv.org/abs/2305.00653.

[29] Hsuan-Cheng Wu, Jingyao Wang, and Xiantao Li. Quantum algorithms for nonlinear dynamics: Revisiting carleman linearization with no dissipative conditions. *arXiv preprint arXiv:2405.12714*, 2024. DOI: 10.48550/arXiv.2405.12714.

[30] Tao Xin, Shijie Wei, Jianlian Cui, Junxiang Xiao, Iñigo Arrazola, Lucas Lamata, Xiangyu Kong, Dawei Lu, Enrique Solano, and Guilu Long. Quantum algorithm for solving linear differential equations: Theory and experiment. *Phys. Rev. A*, 101:032307, Mar 2020. DOI: 10.1103/PhysRevA.101.032307. URL https://link.aps.org/doi/10.1103/PhysRevA.101.032307.

[31] O.V. Zhdaneev, G.N. Serezhnikov, and A.Y. et al Trifonov. Semiclassical trajectory-coherent states of the nonlinear schrödinger equation with unitary nonlinearity. *Russ Phys J*, page 598–606, 1999. DOI: 10.1007/BF02513223.