Quantum algorithms for powering stable Hermitian matrices

Guillermo González1,2,†, Rahul Trivedi1,2,†,∗, and J. Ignacio Cirac1,2

1Max-Planck-Institut für Quantenoptik, Hans-Kopfermann-Str. 1, 85748 Garching, Germany.
2Munich Center for Quantum Science and Technology (MCQST), Schellingstr. 4, D-80799 Munich, Germany.
†equal contribution
(Dated: March 16, 2021)

Matrix powering is a fundamental computational primitive in linear algebra. It has widespread applications in scientific computing and engineering, and underlies the solution of time-homogeneous linear ordinary differential equations, simulation of discrete-time Markov chains, or discovering the spectral properties of matrices with iterative methods. In this paper, we investigate the possibility of speeding up matrix powering of sparse stable Hermitian matrices on a quantum computer. We present two quantum algorithms that can achieve speedup over the classical matrix powering algorithms — (i) an adaption of quantum-walk based fast forwarding algorithm (ii) an algorithm based on Hamiltonian simulation. Furthermore, by mapping the $N$—bit parity determination problem to a matrix powering problem, we provide no-go theorems that limit the quantum speedups achievable in powering non-Hermitian matrices.

I. INTRODUCTION

Recent years have seen rapid progress in the development of quantum computing hardware, and there have already been experimental demonstrations of quantum computations that are believed to be hard to simulate on classical computers [1, 2]. While this progress in hardware has brought us closer to the monumental goal of building a fault tolerant quantum computer, it has also provided us with access to noisy quantum hardware which might already solve problems that are hard for classical computers [3]. From a theoretical standpoint, it has become important to discover algorithms that can provide speedup over their classical counterparts on both fault tolerant quantum computers and near-term noisy quantum hardware.

Quantum computers are known to offer exponential speedup in simulating the physics of quantum systems — near-optimal algorithms have been developed for the simulation of Hamiltonian dynamics [4–6], Lindbladian dynamics [7–10] and steady state (finite temperature or ground state) properties of Hamiltonians [11–14]. Several techniques used for simulating quantum systems have been generalized to accelerate more fundamental linear algebra computational primitives — exponential quantum speedup in the solution of systems of linear equations have been obtained [15–17], and quantum speedups have also been shown in solving ordinary differential equations [18, 19] and partial differential equations [20].

Another fundamental computation that can be accelerated on quantum computers is matrix powering i.e. computing a matrix-element $v^tA^tu$ given access to the matrix $A \in \mathbb{C}^{N \times N}$, a positive integer power $t$ and vectors $v,u \in \mathbb{C}^N$. This is a computational primitive which appears in various applications, including but not limited to solving linear differential equations, simulating discrete-time Markov chains as well as matrix inversion and eigenvalue computation using Krylov subspace methods. Classically, this problem can be solved by repeated matrix multiplication in time $O(\text{poly}(N)Dt)$, where $D$ is the sparsity of the matrix $A$. Without any assumptions on the matrix $A$, one approach to solve the matrix-powering problem on a quantum computer is to map it to a matrix inversion problem and use quantum algorithms for solving linear equations — this approach has been investigated in Refs. [18, 19] in the context of solving linear time-homogeneous ODEs and has a run-time $O(\text{polylog}(N)\text{poly}(\varepsilon^{-1})\kappa_V \|v\|^2 \|u\|^2Dt)$, where $\kappa_V$ is the condition number of the eigenvector matrix of $A$, to obtain $v^tA^tu$ to a precision $\varepsilon$ for stable matrices thereby providing an exponential speedup in the matrix size over classical algorithms.

Furthermore, several authors have studied the problem of powering a stochastic matrix which arises in the context of simulating the dynamics of a discrete-time Markov chain [21–23]. Classically, the problem of powering a stochastic matrix can be solved efficiently to precision $\varepsilon$ with the Monte Carlo algorithm in time $O(Dt\|v\|^2\|u\|^2/\varepsilon^2)$ — using the Monte Carlo algorithm is thus exponentially faster than using repeated matrix multiplication. While the quantum algorithms based on linear-equation solve do not provide an exponential speedup for stochastic matrix powering when compared to the classical Monte Carlo algorithm, there have been two proposals for achieving polynomial quantum speedups for this specific problem. One of the proposed algorithms is to use a reversible implementation of the classical Monte Carlo together with quantum amplitude estimation to achieve a quadratic improvement in the dependence of the run-time on precision as compared to the classical Monte Carlo algorithm [24]. This idea has been applied to propose solutions to the heat equation [25], and stochastic differential equations [26]. A different quantum speedup can be obtained for symmetric stochastic matrices by employing quantum walks [27–30]. In particular, Ref. [31] prepares a quantum state within an $\varepsilon$—radius of $A^tu/\|A^tu\|$ in $O(D\|A^tu\|^{-1}\sqrt{t}\log^{1/2}(\varepsilon^{-1}\|A^tu\|^{-1}))$. The same author generalized this algorithm to arbitrary Hermitian matrices $A$ in Ref. [32]. While this algorithm obtains a quantum speedup over classical methods (an exponential speedup in $N$ over repeated matrix multiplication, and quadratic speedup in $t$ over...
Monte Carlo algorithm), the dependence of the run-time on $\|A^t u\|^{-1}$ can often make it polynomially slow in $N$.

In this paper we introduce two algorithms to compute $v^1 A^t u$ for stable Hermitian matrices $A$ i.e. Hermitian matrices all of whose eigenvalues have magnitudes less than 1. The first algorithm, which combines the construction of Ref. [31] with a Hadamard test [33], has a run-time $O(D\sqrt{T} \|v\| \|u\| \epsilon^{-1} \|A\|_1^t)$. For matrices where it is known that $\|A\|_1 \leq 1$, this provides a quantum speedup over repeated matrix multiplication, since its run-time does not scale polynomially with the size of the matrix. Furthermore, it provides a quadratic speedup in $t$ over classical Monte Carlo algorithm for symmetric stochastic matrices (in which case $\|A\|_1 = 1$). For problems such as the simulation of diffusive discrete-time Markov chains, where $\|A^t u\|^{-1} = O(\sqrt{N})$ at large $t$, this algorithm provides an exponential speedup in the size of matrix $A$ over Ref. [31]. The second algorithm has a run-time of $O(DT^2 poly(\|v\| \|u\| \epsilon^{-1}))$ to compute $v^1 A^t u$ for all $\tau \in \{0, 1, 2, \ldots t\}$. While this is slower than the quantum-walk based algorithm, it only uses Hamiltonian simulation as a primitive and thus is more suitable for near-term quantum hardware. It also achieves a run-time comparable to the quantum algorithms based on linear equation solve [18, 19]. Furthermore, for matrices that are not stochastic and consequently cannot be classically powered with the Monte Carlo algorithm, this algorithm achieves a quantum speedup over repeated matrix multiplication since its run-time does not scale polynomially with the size of the matrix. Finally, following a construction similar to Ref. [34], we provide no-go theorems that limit the speedups achievable with a quantum computer for powering non-Hermitian matrices.

II. PROBLEM DEFINITION, PRELIMINARIES AND SUMMARY OF RESULTS

We consider the problem of powering a Hermitian matrix $A \in \mathbb{C}^{N \times N}$ that is stable i.e. all of its eigenvalues have a magnitude less than 1, or equivalently $\|A\|_2 \leq 1$. Furthermore, we will assume the matrix to be $D$-sparse i.e. every row or column of the stochastic matrix has at most $D$ non-zero elements. Hermitian matrices arising in practice will typically have $D = O(1)$ or $O(polylog(N))$. The matrix powering problem that we consider is precisely defined below.

**Problem (Matrix powering)** Given a $D$-sparse stable Hermitian matrix $A \in \mathbb{C}^{N \times N}$, a positive integer power $t$ and vectors $v, u \in \mathbb{C}^N$, compute $v^1 A^t u$ to a specified precision $\epsilon > 0$.

We point out that previous works that solve the matrix powering problem in various contexts adopt a different problem definition wherein they aim to prepare a quantum state encoding $A^t u$. Since in many applications of matrix powering we are finally interesting in computing its inner product, $v^1 A^t u$, with another vector $v$ which is typically known beforehand, the algorithms proposed in this paper directly compute this expectation value without ever explicitly prepare a quantum state encoding $A^t u$. We make two further notes about this problem definition:

1. The precision of the output of this algorithm is assumed to be in a probabilistic sense i.e. the algorithm is said to produce an estimate $X$ of a quantity $x$ with precision $\epsilon$ if $\text{Prob}[|X - x| \leq \epsilon] = k \in [D]$, where $f(j, k)$ is the index of the $k^{th}$ non-zero element in the $j^{th}$ row or column. The oracle $O_F$ provides access to the indices of the non-zero elements of each column of the Hermitian matrix via the implementation of a unitary that satisfies

$$O_F |j, k\rangle = |j, f(j, k)\rangle \quad \forall j \in [N], k \in [D],$$

where $f(j, k)$ is the index of the $k^{th}$ non-zero element in the $j^{th}$ row or column. The oracle $O_A$ provides access to the non-zero elements of the matrix $A$ via the implementation of a unitary that satisfies

$$O_A |j, k\rangle |z\rangle = |j, k\rangle |z \oplus A_{j,k}\rangle \quad \forall j, k \in [N],$$

where $A_{j,k}$ are the complex elements of the matrix $A$ that are represented by a bit-string up to some specified precision $\delta$. On a quantum computer, these oracles can be implemented with quantum circuits of depth $O(D polylog(1/\delta))$ [35]. On near-term hardware, there might be alternative more efficient ways of implementing these oracles for specific matrices $A$ (for instance the matrices corresponding to local Hamiltonians of a lattice of classical spins). In this paper, for clarity, we will express our complexity results in terms of the number of calls to the oracles $O_F$ and $O_A$ and these can easily be translated to the circuit depths for various hardware-specific implementations.

In the remainder of this paper, we provide several quantum algorithms to solve the matrix powering problem and achieve speedups over classical algorithms. The first algorithm builds on Ref. [31] and combines a quantum walk together with a Hadamard test and a classical sampling algorithm to obtain the following result.

---

1. *Notation for norms:* Throughout this paper, for a vector $v \in \mathbb{C}^N$, $\|v\|_k, k \in \{1, 2, \ldots\}$ will refer to the standard $\ell^k$ norm of the vector. Furthermore, for convenience, we will use $\|v\|_2$ to denote the $\ell^2$ norm of $v$. For matrices $A \in \mathbb{C}^{N \times N}$, $\|A\|_k$ denotes the operator norm induced by $\ell^k$ vector norm i.e. $\|A\|_k = \max_{\|v\|_k = 1} \|Av\|_k / \|v\|_k$. In particular, $\|A\|_2$ will be the largest singular value of $A$, which coincides with the largest magnitude eigenvalue of $A$ is Hermitian. Additionally, for Hermitian matrices $A$, $\|A\|_\infty = \|A\|_1$ will be the maximum absolute row (or column) sum of the matrix $A$. 
Theorem 1 Given a constant $C > 0$ such that $\|A\|_1 < C$, the matrix-powering problem can be solved with a quantum algorithm in $O(C^2D^{\sqrt{1}}v^2 \|u\|^2 \varepsilon^{-2})$ calls to the oracles $O_F, O_A$. Furthermore, by employing the linear combination of unitaries (LCU) technique along with quantum amplitude amplification [24], we can obtain a quadratic improvement in the scaling of the run-time with the precision $\varepsilon$.

Theorem 2 Given a constant $C > 0$ such that $\|A\|_1 < C$, the matrix-powering problem can be solved with a quantum algorithm in $O(C^2D^{\sqrt{1}}v^2 \|u\| \varepsilon^{-1})$ calls to the oracles $O_F, O_A$. Here $\hat{O}$ hides any polylog complexity factors. We point out that these algorithms suffer from an exponential scaling with the power $t$ when $\|A\|_1 > 1$ — this is due to the fact that the quantum walk construction we employ can only be used if the sum of magnitude of the elements of each row (or column) of $A$ is smaller than 1. For a number of matrix powering problems, such as simulation of discrete-time Markov chains, $\|A\|_1 = 1$, and the run-time of these scales sublinearly with $t$. The above results improve the fast-forwarding algorithm presented in Refs. [31], whose run-time scales inversely with $\|A^t u\|$ — our approach avoids this scaling at the expense of computing the overlap $\langle u | A^t v \rangle$. This could be of relevance in problems such as the simulation of diffusive discrete-time Markov chains, where $\|A^t u\|^{-1} = O(\sqrt{N})$ at large $t$. Furthermore, compared to the quantum algorithms based on linear-equation solve, this result has a quadratic speedup in $t$. We also remark that when compared to classical algorithms, we obtain an exponential speedup in $N$ over matrix multiplication algorithm and a quadratic speedup in $t$ over the Monte Carlo algorithm when the matrix $A$ is stochastic.

While the quantum walk based algorithms provided above are able to achieve ‘fast-forwarding’, i.e. a sublinear run-time with respect to the matrix power $t$, they are difficult to implement on near-term quantum hardware. Given experimental constraints, it is widely believed that Hamiltonian simulation [4–6] will be one of the first problems to be solved on practical hardware. Furthermore, simulation of several classes of Hamiltonians can also be implemented on analog quantum simulators [36–38] which are significantly easier to experimentally build as compared to fully-programmable quantum computers. Based on a truncated Fourier series expansion of the function $f(x) = x^\alpha$, we provide a quantum algorithm to solve the matrix-powering problem with only the ability to use Hamiltonian simulation.

Theorem 3 The matrix powering problem can be solved simultaneously for all powers from 0 to $t$ using an efficient Hamiltonian simulator in time $\hat{O}(t^2poly(\|v\| \|u\| \varepsilon^{-1})D)$. We note that this result has a worse run-time not only when compared to the quantum-walk algorithms, but also with classical Monte Carlo algorithm if the matrix $A$ is stochastic. However, it achieves an exponential speedup in $N$ over classical repeated matrix multiplication algorithm although at an expense of quadratically worse scaling with $t$, for matrices that are not stochastic. Furthermore, it achieves the same run-time as quantum algorithms based on the linear-equation solve if they are employed to compute matrix powers from 0 to $t$. The key advantage of this algorithm over other quantum algorithms is its feasibility to being implemented on near-term quantum hardware.

Finally, all the algorithms provided above assume the matrix $A$ to be Hermitian, in which case it was possible to obtain a ‘fast-forwarding’ speedup using quantum walks i.e. compute $A^t$ in time $\Theta(\sqrt{t})$. A natural question to ask is if fast-forwarding is possible for non-Hermitian matrices as well. By utilizing a construction similar to the no-go theorems for Hamiltonian simulation [34] and relying on the result that even a quantum computer cannot speedup the calculation of parity of $N$-bits [39, 40], we provide the following no-go theorem.

Theorem 4 (No-go theorem) There cannot exist a quantum algorithm that solves the matrix-powering problem in $\hat{O}(t^{\alpha}poly(\|v\| \|u\| \varepsilon^{-1}))$ calls to the oracle $O_F, O_A$, with $\alpha < 1$, for any arbitrary irreducible sparse matrix $A$.

We point out that while these no-go theorems rigorously show that it is not possible to fast-forward the matrix-powering problem for generic non-Hermitian matrices, it does not prohibit an improvement of the run-time’s dependence on the size of the matrix. Indeed, matrix-powering methods based on quantum linear equation solvers [18, 19] obtain an exponential improvement over classical algorithms even for non-Hermitian matrices if the matrix is not stochastic.

The remainder of this paper contains proofs of the theorems stated above. In section III we describe the matrix-powering algorithms presented in this paper and prove theorems 1, 2, and 3. In section IV, we prove the no-go theorem 4. We only provide proofs of the most important theorems in the main text, and details are relegated to the appendices.

III. MATRIX MULTIPLICATION ALGORITHM

Before detailing the matrix-powering algorithm, we provide the following lemma that map the computation of $v^\dagger A^t u$ to the overlap of $A^t |\psi>| A^t |\psi>$, with quantum states $|\psi>$ that depend on $u, v$. This transformation is useful since the Hadamard test naturally allows for the computation of such overlaps.

Lemma 1 Given a Hermitian matrix $A$ and vectors $v, u$, it follows that

\[
\text{Re}[v^\dagger A^t u] = \frac{1}{2}\left(\lambda_i^R \langle \psi_i^R | A^t |\psi_i^R \rangle + \lambda_i^R \langle \psi_i^R | A^t |\psi_i^R \rangle \right),
\]

\[
\text{Im}[v^\dagger A^t u] = \frac{1}{2}\left(\lambda_i^I \langle \psi_i^I | A^t |\psi_i^I \rangle + \lambda_i^I \langle \psi_i^I | A^t |\psi_i^I \rangle \right),
\]

where $|\psi_i^R>$ and $\lambda_i^R$, $i \in \{1, 2\}$, are the eigenvectors and eigenvalues of the Hermitian matrix $uu^\dagger + vv^\dagger$ and $|\psi_i^I>$ and $\lambda_i^I$, $i \in \{1, 2\}$, are the eigenvectors and eigenvalues of the Hermitian matrix $i(vu^\dagger - uv^\dagger)$.

Proof: It follows immediately from the Hermiticity of $A$ that $2\text{Re}[v^\dagger A^t u] = \text{Tr}[A^t(uv^\dagger + vu^\dagger)]$ and $2\text{Im}[v^\dagger A^t u] =$
Tr[\( i A^t (v u^t - u v^t) \)]. Using \( w u^t + v u^t = \lambda_1^R |\psi_1^R\rangle \langle \psi_1^R| + \lambda_2^R |\psi_2^R\rangle \langle \psi_2^R| \) and \( i (v u^t - u v^t) = \lambda_1^I |\psi_1^I\rangle \langle \psi_1^I| + \lambda_2^I |\psi_2^I\rangle \langle \psi_2^I| \), we obtain the result in the lemma.

Consequently, we will focus on developing methods to compute the overlap \( \langle \psi| A^t |\psi\rangle \) efficiently. It can also be noted that for problems where \( u \) and \( v \) are sparse, the eigenvectors \( |\psi_{1,2}\rangle \) introduced above are also sparse and consequently efficiently preparable on quantum computers. Furthermore, we note that the eigenvalues \( \lambda_1 \) and \( \lambda_2 \) are bounded by the norms \( u \) and \( v \), which is concretely stated in the following lemma.

**Lemma 2** All eigenvalues \( \lambda \) of \( u v^t + v u^t \) and \( i (v u^t - u v^t) \) satisfy \( |\lambda| \leq 2 \|u\| \|v\| \).

**Proof:** Denoting by \( |\psi\rangle \) the normalized eigenvector corresponding to the eigenvalue \( \lambda \), it follows that \( |\lambda| = \| (u v^t + v u^t) |\psi\rangle \| \leq \| u \| |v^t| |\psi\rangle \| + \| v \| |u^t| |\psi\rangle \| \leq 2 \| u \| \| v \| \). A similar proof holds for the eigenvalues of \( i (v u^t - u v^t) \).

\[ A^t = \sum_{m=0}^{t} p_m T_m(A) , \]

where \( p_m \) is a probability distribution given by

\[ p_m = \begin{cases} \frac{1}{2^m} \binom{t}{(\frac{t}{2})} & \text{for } m > 0, t = m \bmod 2 \\ \frac{1}{2^{t/2}} & \text{for } m = 0, t = 0 \bmod 2 \end{cases} \]

Consequently, a quantum circuit to compute the overlap \( \langle \psi| A^t |\psi\rangle \) can be constructed from a quantum circuit that can compute the overlap \( \langle \psi| T_m(A) |\psi\rangle \) for a specified \( m \in \{0, 1 \ldots t\} \). As is shown below, this can be done with a quantum walk provided that the 1-norm of \( A \) is smaller than 1. Since this isn’t necessary for stable Hermitian matrices, we assume that we have access to an upper bound \( C \) on this norm i.e. \( \|A\|_1 \leq C \) and compute \( \langle \psi| (A/C)^t |\psi\rangle \), albeit to a precision \( C^t \) higher than that required in \( \langle \psi| A^t |\psi\rangle \). Therefore, in the remainder of this section, unless otherwise mentioned, we will assume \( \|A\|_1 \leq 1 \).

A quantum walk construction similar to that used in Refs. [27, 28, 31] together with a Hadamard test allows us to compute these overlaps. However, since the elements of the matrix \( A \) can be complex, it is important to design the quantum walk with care so as to account for the phase of the complex matrix elements [35]. For \( A \in \mathbb{C}^{N \times N} \), we consider a Hilbert space \( \mathbb{C}^{N+1} \otimes \mathbb{C}^{N+1} \otimes \mathbb{C}^2 \) and assume access to a unitary \( V \) that satisfies

\[ W = -(I \otimes I \otimes \sigma_z) V^t S V. \]

We then obtain the following lemma, similar to that obtained in Refs. [31, 32] stating that \( m \) applications of the quantum walk operator effectively applies \( T_m(A) \) on an input state conditioned on the state of the last qubit.

**Lemma 3 (Quantum walk for \( T_m(A) \))** The unitary operator \( W \) defined in Eq. 7 satisfies

\[ W^m |\psi\rangle |0\rangle |0\rangle = T_m(A) |\psi\rangle |0\rangle |0\rangle + |\psi^\perp\rangle |1\rangle \]

for some \( |\psi^\perp\rangle \in \mathbb{C}^{N+1} \otimes \mathbb{C}^{N+1} \).

As is shown in appendix A, the quantum walk operator \( A \) can be implemented with \( O(D) \) calls to the oracles \( O_F, O_A \).
that access the matrix $A$. In order to estimate the overlap $\langle \psi \mid T_m(A) \mid \psi \rangle$ using the walk operator $W$, we introduce a controlled version of $W$, $W^c$ via

$$W^c = I \otimes |0\rangle \langle 0| + W \otimes |1\rangle \langle 1|.$$  \hspace{1cm} (9)

We then have the following lemma to compute the overlap $\langle \psi \mid T_m(A) \mid \psi \rangle$ using a hadamard test with the controlled operator $W^c$.

**Lemma 4 (Chebyshev polynomial overlap)** Consider the state $(W^c)^m \mid \psi \rangle \mid 0, 0, +\rangle$, and measure the last two qubits on the basis $\{ |0, +\rangle, |0, -\rangle, |1, +\rangle, |1, -\rangle \}$. Define a random variable $X_m$ based on the measurement outcome $\mu$ via

$$X_m = \begin{cases} +1 & \text{if } \mu = (0, +) \\ -1 & \text{if } \mu = (0, -) \\ 0 & \text{otherwise} \end{cases}$$

then $E(X_m) = \langle \psi \mid T_m(A) \mid \psi \rangle$.

While this overlap estimation procedure can be used together with the Chebyshev polynomial expansion in Eq. 3 to compute $\langle \psi \mid A^\dagger \mid \psi \rangle$, this would not provide any fast-forwarding since computing $T_1(A)$ would require $t$ applications of the operator $W^c$. However, an important insight into the nature of the coefficients $p_m$ in the expansion in Eq. 4 is that they concentrate around $m \sim \sqrt{t}$. One possible approach to exploit this property is to sample $m$ from the probability distribution given by the coefficients $p_m$ in Eq. 4, and compute the overlap $\langle \psi \mid T_m(A) \mid \psi \rangle$ of the corresponding Chebyshev polynomial using lemma 4 —— this would allow us to reduce, on an average, the number of times the walk operator $W^c$ is applied. This is formalized in the lemma below:

**Lemma 5 (Matrix power overlap with classical sampling)**

The overlap $\langle \psi \mid A^\dagger \mid \psi \rangle$ can be computed by estimating the mean of a random variable $X$ which is generated by first sampling $m \in \{0, 1, \ldots, t\}$ from the probability distribution in Eq. 4, followed by drawing a sample of $X_m$ defined in lemma 4 using the state $\psi$. Furthermore, this estimation can be done with a precision $\epsilon$ with $O(\epsilon^{-2} \sqrt{t})$ calls to the controlled walk operator $W^c$ or equivalently with $O(D \epsilon^{-2} \sqrt{t})$ calls to the oracles $O_F, O_A$.

**Proof:** We can immediately see that

$$E(X) = \sum_{m=0}^{t} p_m E(X_m) = \sum_{m=0}^{t} p_m \langle \psi \mid T_m(A) \mid \psi \rangle = \langle \psi \mid A^\dagger \mid \psi \rangle$$

Furthermore, noting that $X^2 \in \{0, 1\}$, it follows that $\text{var}(X) \leq 1$. Consequently, $E(X)$ can be estimated to a precision of $\epsilon$ with $N = O(1/\epsilon^2)$ samples. Furthermore, the average number of calls to the controlled walk operator $W^c$, $\langle m \rangle$ is given by

$$\langle m \rangle = \sum_{m=0}^{t} m p_m = \frac{1}{2^t} \sum_{n=0}^{t/2} n \left( \frac{t}{t/2 - n} \right) = \frac{2 + t}{2^{t-2}} \left( \frac{t}{t/2 - 1} \right).$$

(11)

For large $t$, using Stirling’s approximation this is

$$t \sim \frac{t}{t/2 - 1} = \frac{t/2 + 1}{t/2} \sim \frac{t}{t/2} \sim \frac{2^t}{\sqrt{t \pi / 2}}.$$\hspace{1cm} (12)

Therefore $\langle m \rangle = O(\sqrt{t})$, and consequently the number of calls to $W^c$ operator to achieve a precision $\epsilon$ given by $N(m) = O(\epsilon^{-2} \sqrt{t})$.

**Proof of theorem 1:** Combining lemma 5 with lemma 1, we obtain a procedure for solving the matrix powering problem. The complexity result in theorem 1 can be obtained as follows: Given an upper bound $C$ on $\|A\|_1$, we note that to compute $v^\dagger A^\dagger \mid u \rangle$ precision $\epsilon$, we need to compute $\langle \psi_{1,2} \mid (A/C)^t \mid \psi_{1,2} \rangle$ to a precision at-most $\epsilon/4 N ||v|| ||v|| C^t$ which can be done using the algorithm in lemma 1 with $O(D \epsilon^{-2} \sqrt{t} ||v||^2 ||v||^2 / C^t)$ calls to the oracles $O_F, O_A$.

While the algorithm described above allows a quadratic fast-forwarding for the matrix powering problem, the dependence of the run-time on the precision $\epsilon$ can also be improved by using amplitude amplification, which is precisely stated in the following lemma from Ref. [24]:

**Lemma 6 (Overlap estimation, Theorem 2.5 of Ref. [24])**

Given a state $\mid \psi \rangle$ in terms of its preparation unitary $U$ from a known state $\mid 0 \rangle$: $\mid \psi \rangle = U \mid 0 \rangle$, an observable $V$ and an estimate $\sigma$ of its variance satisfying $\langle \psi \mid V^2 \mid \psi \rangle - (\langle \psi \mid V \mid \psi \rangle)^2 \leq \sigma^2$, $\langle \psi \mid V \mid \psi \rangle$ can be estimated on a quantum computer with precision $\epsilon$ with $O(\epsilon^{-3} \sigma)$ calls to the unitary $U$.

In order to use amplitude amplification and achieve fast forwarding in the same algorithm, we approximate the problem of computing the overlap $\langle \psi \mid A^\dagger \mid \psi \rangle$ for a given $\mid \psi \rangle$ by computing an overlap of the form $\langle \phi_1 \mid V \mid \phi_1 \rangle$ where the operator $V$ is independent of $t$ and the state $\mid \phi_1 \rangle$ can be prepared in $\Theta(\sqrt{t})$ calls to the oracles $O_F, O_A$. This is achieved by using a combination of quantum walks with the hadamard test and the linear combination of unitaries (LCU) technique similar to Ref. [21]. The quadratic fast-forwarding in this approach is also obtained due to the concentration of the coefficients $p_m$ in Eq. 4 around $m = \sqrt{t}$. This is made concrete by the following lemma, according to which the sum in Eq. 3 can be truncated after $\sim O(\sqrt{t} \log(1/\epsilon))$ terms while incurring a specified additive error $\epsilon$.

**Lemma 7** If $A$ is a stable Hermitian matrix and $\mid \psi \rangle$ is a normalized state, then $\forall \epsilon > 0$ and $C \geq 2 \log(2/\epsilon)$

$$\left| \langle \psi \mid A^\dagger \mid \psi \rangle - \langle \psi \mid \sum_{m=0}^{\sqrt{t} \epsilon} p_m T_m(A) \mid \psi \rangle \right| \leq \epsilon$$
Proof: Using lemma 3 from Ref. [31], we obtain that $\forall \varepsilon > 0, C \geq 2 \log(2/\varepsilon)$

$$
|A^t - \sum_{m=0}^{\sqrt{CT}} \rho m T_m(x)| \leq \varepsilon \forall x \in [-1, 1].
$$

(13)

Since $A$ is a stable matrix, its eigenvalues will have a magnitude less than equal to 1. Furthermore, since $A$ is also Hermitian, its eigenvalues are real and lie in $[-1, 1]$ and hence satisfy Eq. 13. Denoting by $\lambda_i, |\phi_i\rangle$ the eigenvalues and eigenvectors of $A$ and using its spectral decomposition $A = \sum_i \lambda_i |\phi_i\rangle \langle \phi_i|$, we obtain

$$
|\langle \psi | A^t | \psi \rangle - (\psi \sum_{m=0}^{\sqrt{CT}} \rho m T_m (A) | \psi \rangle) | \leq \sum_k |\langle \phi_k | \psi \rangle|^2 \lambda_k - \sum_{m=0}^{\sqrt{CT}} \rho m T_m (\lambda_k) | \leq \varepsilon
$$

(14)

Consequently, we can effectively approximate $A^t$ as weighted linear combination of $O(\sqrt{T})$ Chebyshev polynomials of $A$ — while the quantum walk operator introduced in Eq. 7 can be used to individually implement the Chebyshev polynomials, in order to implement their linear combination we use the LCU technique [6]. Below, we show the construction of an operator to effectively apply $\sum_{m=0}^{\sqrt{T}} \rho m W^{m}$ to given quantum state. We do this by introducing auxiliary qubits and implementing the following unitary $V_P$ depending on the coefficients $\rho m$:

$$
V_P |\phi\rangle |0,0\rangle = \sum_{m=0}^{\tau} \sqrt{\rho m} |\phi\rangle |m,0\rangle + \left[1 - \sum_{m=0}^{\tau} \rho m \right]^{1/2} |\phi\rangle |0,1\rangle.
$$

(15)

Furthermore, we assume access to a controlled quantum walk operator $W_\tau$ defined by

$$
W_\tau = \sum_{m=0}^{\tau} W^{m} \otimes |m\rangle \langle m| \otimes I
$$

(16)

The operator $W_\tau$ requires $\tau$ calls to the quantum walk operator $W$. Therefore, following the result in appendix A, it can be constructed with $O(D\tau)$ calls to the oracles $O_F, O_A$. The operator $V_P^\dagger W_P V_P$ then effectively applies the linear combination $\sum_{m=0}^{\tau} \rho m W^m$ to an input state.

Lemma 8 (LCU adapted from Refs. [6, 31]) The unitary operator $V_P^\dagger W_\tau V_P$, with $V_P$ and $W_\tau$ defined in Eqs. 15 and 16 respectively, satisfy

$$
V_P^\dagger W_P V_P |\phi\rangle |0,0\rangle = \sum_{m=0}^{\tau} \rho m W^{m} |\phi\rangle |0,0\rangle + |\phi_{\perp}\rangle |1\rangle,
$$

(17)

for some $|\phi_{\perp}\rangle$.

In order to compute $\langle \psi | \sum_{m=0}^{\tau} \rho m T_m (A) | \psi \rangle$, we consider a controlled version of the operator defined in lemma 8:

$$
W_\tau^\varepsilon = I \otimes |0\rangle \langle 0| + V_P^\dagger W_P V_P \otimes |1\rangle \langle 1|.
$$

(18)

It then follows from lemmas 3 and 8 that computing the expectation value of the operator $|0\rangle \langle 0| \otimes \sigma_x$ on the last two qubits in the circuit of $W_\tau^\varepsilon$ on a state prepared by application of $HW_\tau^\varepsilon H$ (where $H$ is a hadamard gate on the last qubit) to $|\psi\rangle |0,0...0\rangle$ allows us to evaluate $\langle \psi | \sum_{m=0}^{\tau} \rho m T_m (A) | \psi \rangle$. By truncating the linear combination to an appropriate number of terms using lemma 7 and using amplitude estimation (lemma 6), we obtain the following lemma for estimating the overlap of the matrix with a given state.

Lemma 9 (Matrix power overlap with LCU) For a Hermitian stable matrix $A$ with $\|A\|_1 \leq 1$ and a state $|\psi\rangle$, $|A^t |\psi\rangle$ can be computed on a quantum computer to a precision $\varepsilon$ with $O(D\sqrt{\varepsilon^{-1}})$ calls to the oracles $O_F, O_A$.

Proof: The overlap estimation can be done using the Hadamard test described above — as shown in lemma 7, it is sufficient to use $\tau = O(\sqrt{T} \log (1/\varepsilon^{-1}))$ in the LCU construction described in lemma 8. Furthermore, the outcome of the Hadamard test has a variance bounded by 1 and consequently a direct application of amplitude amplification (lemma 6) allows us to obtain an estimate of $\langle \psi | A^t | \psi \rangle$ with $O(D\sqrt{T}/\varepsilon)$ calls to the oracles $O_F, O_A$.

Proof of theorem 2: Combining lemma 9 with lemma 1, we obtain a procedure for solving the matrix powering problem. The complexity result in theorem 2 can be obtained as follows: Given an upper bound $C$ on $\|A\|_1$, we note that to compute $v_i^1 A^t u$ to precision $\varepsilon$, we need to compute $\langle \psi_{L,R}^1 (A/C)^t | \psi_{L,R}^1 \rangle$ to a precision at most $\varepsilon/4$, $\|u\| \|v\| C^t$ which can be done using the algorithm in lemma 1 with $O(\sqrt{T} \|u\| \|v\| C^t \varepsilon^{-1})$ calls to the oracles $O_F, O_A$.

B. Matrix powering with hamiltonian simulation

In this section, we describe an approach to solve the matrix powering problem using Hamiltonian simulation as a primitive and prove the complexity result in theorem 3. Formally for our purposes, a Hamiltonian simulation can be considered to be the problem of computing $\langle \psi | e^{-iHt} | \psi \rangle$ to a precision $\varepsilon$ given access to the sparse Hamiltonian $H$ and the states $|\psi\rangle$. A Hamiltonian simulator (implemented on a quantum computer or an analog quantum simulator) is said to be efficient if it can solve this problem in $O(\text{poly}(\varepsilon^{-1})D \|H\|_{\text{max}} t)$ time, where $D$ is the sparsity of the Hamiltonian and $\|H\|_{\text{max}}$ is its maximum element. State of the art algorithms for Hamiltonian simulation on quantum computers achieve such run-times for general sparse Hamiltonians, while such run-times can be achieved on quantum simulators for local Hamiltonians.

We again restrict ourselves to stable Hermitian matrices $A$. In order to compute an overlap $\langle \psi | A^t | \psi \rangle$, we expand $A^t$ into a fourier series — as is shown in the following two lemmas, this can be done to a precision of $\varepsilon$ while retaining only $N_h = O(t/\varepsilon)$ harmonics.
Lemma 10 \( \forall \varepsilon \in (0, 2/\pi), N_h \geq 4t/\pi^2 \varepsilon, \exists a \in \mathbb{C}^{2N_h+1} \) such that
\[
|x^t - \sum_{n=-N_h}^{N_h} a_n e^{in\pi x/2}| \leq \varepsilon \quad \forall \ x \in [-1, 1],
\]
where \( a_n \) are the elements of the vector \( a \) and \( a \) can be computed classically in \( O(N_h t) \) time. Furthermore, \( \|a\|_1 \leq 1 \quad \forall t > 0, N_h > 0. \)

A detailed proof of this lemma, as well as an explicit calculation of the coefficient vector \( a \), is provided in appendix B. Employing this result, we can now compute the overlap of the power \( \langle \psi | A^t | \psi \rangle \) using an efficient Hamiltonian simulator.

Lemma 11 (Matrix overlap with Hamiltonian simulation)
Given a \( D \)-sparse stable Hermitian matrix \( A \), \( \langle \psi | A^t | \psi \rangle \) can be computed for all \( \tau \in \{0, 1 \ldots t\} \) using an efficient Hamiltonian simulator in time \( O(\text{poly}(\varepsilon^{-1})Dt^2) \).

Proof: Since the matrix \( A \) is stable and Hermitian, all of its eigenvalues are real and lie in \([-1, 1]\) thus satisfying Eq. 18. Denoting by \( \lambda_k, |\phi_k\rangle \) the eigenvalues and eigenvectors of \( A \) and using lemma 10, we obtain that
\[
\left| \langle \psi | A^t | \psi \rangle - \sum_{n=-N_h}^{N_h} a_n |\langle \psi | e^{i\pi nA/2} | \psi \rangle \right| \leq \varepsilon/2
\]
for an appropriately chosen \( N_h = O(\tau/\varepsilon) \). Furthermore, we note that since \( A \) is stable, the magnitude of all of its elements is at most 1 i.e. \( \|A\|_{\text{max}} \leq 1 \). Using an efficient Hamiltonian simulator, we can estimate \( \langle \psi | e^{i\pi nA/2} | \psi \rangle \) to a precision \( \varepsilon/2 \) in time \( O(\text{poly}(\varepsilon^{-1})Dn) \). Since \( \|a\|_1 \leq 1 \), we can then compute \( \sum_{n=-N_h}^{N_h} a_n |\langle \psi | e^{i\pi nA/2} | \psi \rangle \) to a precision \( \varepsilon \) on a classical computer, thus determining \( \langle \psi | A^t | \psi \rangle \) to a precision \( \varepsilon \) — since we need to compute \( \langle \psi | e^{i\pi nA/2} | \psi \rangle \) for \( n \in \{-N_h, -N_h + 1 \ldots N_h - 1, N_h\} \), the total time taken for this computation is \( O(\text{poly}(\varepsilon^{-1})DN_h^2) = O(\text{poly}(\varepsilon^{-1})D\tau^2) \). Furthermore, we note that since we propose to compute the overlaps \( \langle \psi | e^{i\pi nA/2} | \psi \rangle \) individually for all \( n \), we can compute \( \langle \psi | A^t | \psi \rangle \) for all \( \tau \in \{0, 1 \ldots t\} \) with the same set of Hamiltonian simulations in time \( O(\text{poly}(\varepsilon^{-1})Dt^2) \).

Proof of theorem 3: Combining lemma 11 with 1, we obtain a procedure for solving the matrix powering problem. To obtain the complexity result in theorem 3, we note that computing \( v^\dagger A^t u \) to precision \( \varepsilon \), we need to compute \( \langle \psi^{i,R}_{1,2} | A^t | \psi^{i,R}_{1,2} \rangle \) to precision at most \( \varepsilon/4 \|u\| \|v\| \). This can be done using the algorithm in 11 for \( \tau \in \{0, 1, 2 \ldots t\} \) simultaneously in \( O(\text{poly}(\varepsilon^{-1})\|u\| \|v\|Dt^2) \) time.

IV. NO-GO THEOREMS FOR FAST FORWARDING

In this section, we provide no-go theorems stated in section II, showing that fast forwarding the matrix powering problem is not possible for a generic non-Hermitian matrix. These no-go theorems utilize a construction similar to the no-go theorems for Hamiltonian simulation [34], and rely on the fact that even a quantum computer cannot speedup the calculation of parity of \( N \)-bits [39, 40], a result concretely stated as the following Lemma:

Lemma 12 (\( N \)-bit parity problem, Refs. [39, 40])
Consider \( N \)-bits \( b_1, b_2 \ldots b_N \) which can be accessed as an oracle \( O_B : O_B |i, b\rangle = |i, b \oplus b_i\rangle \). There cannot exist a quantum algorithm that can determine the parity \( b_1 \oplus b_2 \oplus \ldots b_N \) with success probability greater than 1/2 with fewer than \( N/2 \) calls to the oracle \( O_B \).

We note that this result rules out even an approximate solution of the \( N \)-bit parity problem on a quantum computer with run-time less than \( O(N) \). In particular, since the parity is known to be an integer, if an algorithm can estimate this parity to a precision \( \varepsilon < 1/2 \) with a confidence level greater than 1/2, then it would have solved the \( N \)-bit parity problem — consequently, in the rest of this analysis we can consider the precision \( \varepsilon = O(1) \). As is shown in the no-go theorem below, we can construct a matrix such that computing its \( N \)-th power, in the sense specified in section II, allows us to solve the \( N \)-bit parity problem, thereby ruling out the possibility of designing a quantum algorithm that can achieve a run-time scaling sublinearly with the matrix power. We first provide a proof of lemma 13, which rules out the possibility of fast-forwarding the powering of a general matrix, and then strengthen it to obtain theorem 4 which rules out the possibility of fast-forwarding even the powering of irreducible matrices.

Lemma 13 (No-go theorem for arbitrary matrix)
There cannot exist a quantum algorithm that solves the matrix powering problem in \( \tilde{O}(\text{poly}(|\|u\|, |\|v\|, \varepsilon^{-1}) \|D\|t^2)) \) calls to the oracles \( O_F, O_A \) with \( \alpha < 1 \), for any sparse matrix \( A \).

Proof: Given \( N \) bits \( b_1, b_2 \ldots b_N \), we can construct the following matrix powering problem that determines the parity of \( b_1 \oplus b_2 \oplus \ldots \oplus b_N \) on computing its \( N \)-th power:

1. The matrix \( A \in \mathbb{C}^{2N \times 2N} \) with the rows (or columns) being indexed by \( (i, \sigma) \) where \( i \in [N] \) and \( \sigma \in \{0, 1\} \) and matrix elements being given by
\[
A_{i,\sigma,i',\sigma'} = \begin{cases} 
1 & \text{if } i' = i-1 \text{ and } \sigma \oplus \sigma' = b_{i-1} \\
0 & \text{otherwise} 
\end{cases}
\]
2. The vectors $u$ and $v$ are given by

$$u_{i,\sigma} = \begin{cases} 
1 & \text{if } i = 0 \text{ and } \sigma = 0 \\
0 & \text{otherwise}
\end{cases},$$

$$v_{i,\sigma} = \begin{cases} 
1 & \text{if } i = N \text{ and } \sigma = 0 \\
-1 & \text{if } i = N \text{ and } \sigma = 1 \\
0 & \text{otherwise}
\end{cases}. \quad (21)$$

The matrix $A$ is a stochastic matrix corresponding to a discrete-time Markov chain (DTMC) shown in Fig. 1a — the states of this DTMC can be grouped as per their $\sigma$ index, and for every bit $b_i$ is identified with a flip of the $\sigma^i$ index at $i$. Consequently, on computing $v^\dagger A^t u$, we can count the number of bits that are 1 thereby computing $b_1 \oplus b_2 \cdots \oplus b_N$. Furthermore, we can note that the oracles $O_F, O_A$ can be constructed with $O(1)$ calls to the oracle $O_B$ since each bit determines the positions of the non-zero elements in at-most two columns. Furthermore, we note that $\|u\|, \|v\| = O(1)$ by construction. Consequently, if there existed a quantum algorithm to solve the DTMC simulation problem with $O(t^{\alpha}\text{poly}(\|u\|, \|v\|, 1/\varepsilon))$ queries to $O_F, O_A$ with $\alpha < 1$, it could solve the $N$-bit parity problem in $O(N^\alpha)$ queries to $O_B$. This contradicts lemma 12 thus proving that no such quantum algorithm can exist.

While this argument proves that no quantum algorithm can exist to fast-forward the matrix powering problem for a general matrix, we note that the specific matrix $A$ used in this argument is reducible. Consequently, this raises the question of whether their exists a quantum algorithm that can fast-forward the powering of arbitrary irreducible matrices. We show that this too isn’t possible by constructing an irreducible stochastic matrix that is very close to the reducible stochastic matrix constructed above, and thus approximately solves the $N$-bit parity problem.

**Proof of theorem 4:** We consider an instance of the matrix-powering problem with $u, v$ as defined in Eq. 21 and a matrix $A_\delta = A + B\delta$ where $A$ is defined in Eq. 20, $\delta \in (0, 1)$ and matrix $B$ has elements given by:

$$B_{i,\sigma,i',\sigma'} = \begin{cases} 
-1 & \text{if } i' = i - 1 \text{ and } \sigma \oplus \sigma' = b_{i-1} \\
1 & \text{if } i' = i + 1 \text{ and } \sigma \oplus \sigma' = b_i \\
1 & \text{if } i = i' \in \{1, N\}, \sigma \neq \sigma' \\
0 & \text{otherwise}
\end{cases} \quad (22)$$

The matrix $A_\delta$ is the stochastic matrix corresponding to a discrete-time Markov chain shown in Fig. 1b. It is easy to see that $A_\delta$ is irreducible for $\delta \neq 0$. Furthermore, we can easily bound the difference between the result of powering $A_\delta$ and $A$:

$$|v^\dagger A_\delta^t u - v^\dagger A^t u| \leq \|v\|\|u\| (\|A\|_2 + \|B\|_2 \delta^t - \|A\|_2^t) \leq \|v\|\|u\| \|A\|_2^t \left( e^{\|B\|_2 \delta t/\|A\|_2} - 1 \right). \quad (23)$$

For the choice of $u, v$ under consideration, $\delta = \log(1 + \varepsilon/2\sqrt{2})/8N^2$ and $t = N$ it follows that

$$|v^\dagger A_\delta^N u - v^\dagger A^N u| \leq \frac{\varepsilon}{2} \quad (24)$$

wherein we have used $\|v\| = \sqrt{2}$ and $\|u\| = 1$ and $1/\sqrt{2N} \leq \|A\|_2 \leq 1$, $\|B\|_2 \leq 4\sqrt{2N}$. This shows that being able to compute $v^\dagger A_\delta^N u$ to precision $\varepsilon/2$ allows us to determine $v^\dagger A^N u = b_1 \oplus b_2 \cdots \oplus b_N$ to precision $\varepsilon$. Consequently, from lemma 13, the no-go theorem follows for irreducible matrices as well.

**V. CONCLUSION**

This paper studied the problem of computing the power of a stable Hermitian matrix. Following the construction of Ref. [31], we show that fast-forwarding is possible while powering stable Hermitian matrices and present algorithms based on quantum walks that improve their results. We also present a complementary algorithm to solve the matrix powering problem using only Hamiltonian simulators which could potentially be used on near-term quantum hardware. Finally, by establishing a map between the $N$-bit parity determination problem to a matrix powering problem, we show that quantum computers cannot fast-forward powering of non-Hermitian matrices.

**ACKNOWLEDGMENTS**

We acknowledge support from the ERC Advanced Grant QUENOCOBA under the EU Horizon 2020 program (grant agreement 742102) and from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the project...
number 414325145 in the framework of the Austrian Science Fund (FWF): SFB F7104.

We can write

\[ V |i,0,0\rangle = \sum_{k=1}^{N} \sqrt{|A_{k,i}|} e^{i\varphi_{k,i}/2} |i,k,1\rangle + \left(1 - \sum_{k=1}^{N} |A_{k,i}|\right)^{1/2} |i,N+1,1\rangle \text{ if } i \neq N+1, \]

\[ V^\dagger |i,j,1\rangle = \sqrt{|A_{j,i}|} e^{-i\varphi_{j,i}/2} |i,0,0\rangle + |\phi^\perp\rangle |1\rangle \text{ for some } |\phi^\perp\rangle \text{ if } i \neq N+1, \]

\[ V |N+1,j,b\rangle = |N+1,j,b\rangle. \]

We can write \( V \) as

\[ V = |N+1\rangle \langle N+1| \otimes I \otimes I + \sum_{i=1}^{N} |i\rangle \langle i| \otimes V_i \]  \hspace{1cm} (A2)

Therefore, \( V \) applies \( V_i \) depending on the second and third registers depending on the state of the first qubit. The \( V_i \) act as

\[ V_i = |\psi_i\rangle \langle 0,0| + \sum_{j=1}^{N} |j,0\rangle \langle j,0| + \sum_{j=1}^{N+1} |\psi^\perp_{i,j}\rangle \langle j,1| \]  \hspace{1cm} (A3)

\[ |\psi^\perp_{i,j}\rangle, \text{ along with } |\psi_i\rangle, \text{ form an orthonormal basis. The expression for } |\psi_i\rangle \]

\[ |i\rangle \langle \psi_i| = U |i,0,0\rangle \]

\[ = \sum_{k=1}^{N} \sqrt{|A_{k,i}|} |i,k,1\rangle + \left(1 - \sum_{k=1}^{N} |A_{k,i}|\right) |i,N+1,1\rangle \]

where the square root is defined as in A1a. Therefore, we need to show that we can apply the operator \( U \) that prepares \( |\psi_i\rangle \) with \( O(D) \) queries to the oracles. This can be proved with a slight modification of the procedure in [41].

To do this, we start with the state \(|i\rangle \langle 0|\). We then prepare a list with all the neighbours of \( i \): \(|i\rangle \langle f(i,1)| \ldots |f(i,D)|\). This can be done with \( O(D) \) calls to \( O_F \). Now we can prepare a list containing all the non-zero probabilities in the i-th row \(|i\rangle |A_{i,f(i,1)}\rangle \ldots |A_{i,f(i,D)}\rangle\). This can be done with \( O(D) \) calls to \( O_M \). Appending an extra register to this list, we can compute \(|i\rangle |A_{i,f(i,1)}\rangle |A_{i,f(i,D)}\rangle |1 - \sum_{k=1}^{D} |A_{i,f(i,k)}|\rangle\) Using both lists, following the procedure in [41], we can prepare the state

\[ \sum_{j=1}^{D} \sqrt{|A_{i,f(i,j)}^*|} |j\rangle |\rangle + 1 - \sum_{k=1}^{D} |A_{i,f(i,k)}| |i,N+1\rangle \]  \hspace{1cm} (A5)

Querying again \( O_F \) we obtain the desired state. Therefore, it follows that the quantum walk operator \( W \) can be implemented with \( O(D) \) calls to \( O_F \) and \( O_A \).

**Appendix A: Costs for quantum walk operators in terms of matrix oracles**

In this appendix we show how the quantum walk operator \( W \) can be implemented with \( O(D) \) calls to the oracles \( O_F, O_A \) that access the matrix \( A \), where \( D \) is the sparsity. The quantum walk operator is \( W = V^\dagger SV \). We have assumed access to a unitary operator \( V \) such that

\[ {}^V \langle i| V^\dagger |j\rangle = |A_{j,i}| e^{-i\varphi_{j,i}/2} |i\rangle \langle j| \]

\[ V |i,0,0\rangle = \sum_{k=1}^{N} \sqrt{|A_{k,i}|} e^{i\varphi_{k,i}/2} |i,k,1\rangle + \left(1 - \sum_{k=1}^{N} |A_{k,i}|\right)^{1/2} |i,N+1,1\rangle \text{ if } i \neq N+1, \]

\[ V^\dagger |i,j,1\rangle = \sqrt{|A_{j,i}|} e^{-i\varphi_{j,i}/2} |i,0,0\rangle + |\phi^\perp\rangle |1\rangle \text{ for some } |\phi^\perp\rangle \text{ if } i \neq N+1, \]

\[ V |N+1,j,b\rangle = |N+1,j,b\rangle. \]

In this appendix, we provide a proof of lemma 10 introduced and used in the main text. Our goal is to calculate a fourier series expansion of \( x^t \) which converges pointwise when \( x \in [-1,1] \), and estimate the number of terms of the fourier series that we need to retain to achieve a certain precision in this expansion. We point out that for odd \( t \), a fourier series expansion of \( x^t \) is not continuous. Consequently, we instead consider the function \( f_t(x) \) on the interval \([-1,1]\) could exhibit a Gibb’s phenomena at \( x = \pm 1 \) since the periodic extension of \( x^t \) is not continuous. Consequently, we instead consider the function \( f_t(x) \) on the interval \([-2,2]\) defined below

\[ f_t(x) = \begin{cases} x^t & \text{for } |x| \leq 1, \\ (2-x)^t & \text{for } 1 < x \leq 2, \\ (-2-x)^t & \text{for } -2 \leq x < -1. \end{cases} \]  \hspace{1cm} (B1)

The periodic extension of this function is continuous, since \( f_t(2) = f_t(-2) \forall t \) and \( f_t(x) \) is continuous within the interval \((-2,2)\). Furthermore, for \( |x| \leq 1 \) this function coincides with \( x^t \). We can now write down a fourier series expansion for this function which converges point-wise for all \( x \in [-2,2] \) — for ease of analysis, we treat the cases when \( t \) is even and odd.
seperately:
\[
f_t(x) = \begin{cases} 
\sum_{p=0}^{\infty} c_p(t) \cos(p \pi x) & \text{if } t \text{ is even}, \\
\sum_{p=0}^{\infty} s_p(t) \sin((2p + 1)\pi x/2) & \text{if } t \text{ is odd},
\end{cases}
\]

where
\[
c_p(t) = \int_0^2 f_t(x) \cos(p \pi x) dx \\
= 2 \int_0^1 x^t \cos(p \pi x) dx, \quad (B3a)
\]
\[
s_p(t) = \int_0^2 f_t(x) \sin((2p + 1)\pi x/2) dx \\
= 2 \int_0^1 x^t \sin((2p + 1)\pi x/2) dx. \quad (B3b)
\]

We point out that the Eq. B2 can be rewritten in terms of complex exponentials to obtain a fourier series of the form used in lemma 10. Furthermore, \( c_p(t) \) and \( s_p(t) \) can be explicitly evaluated to obtain
\[
c_p(t) = 2(-1)^p \sum_{k=1}^{t/2} \frac{(-1)^{k+1} 2^{-k} \prod_{i=0}^{k-2} (t - i)}{(p^2 \pi^2)^k} \quad (B4a)
\]
\[
s_p(t) = 2(-1)^p \sum_{k=1}^{(t+1)/2} \frac{(-1)^{k+1} 2^{-k} \prod_{i=0}^{k-2} (t - i)}{(p^2 \pi^2)^k} \quad (B4b)
\]

We point out that \( s_p(t) \) and \( c_p(t) \) can be computed in \( O(t) \) time on a classical computer using a recursive implementation of the summations in Eq. B4. We now consider a truncated fourier series expansion i.e. we construct the function \( f_t^N(x) \) from the coefficients \( c_p(t), s_p(t) \) where
\[
\hat{f}_t^N(x) = \begin{cases} 
\sum_{p=0}^{N} c_p(t) \cos(p \pi x) & \text{if } t \text{ is even}, \\
\sum_{p=0}^{N} s_p(t) \sin((2p + 1)\pi x/2) & \text{if } t \text{ is odd}.
\end{cases}
\]

We then obtain that \( \forall x \in [-1, 1], |x^t - f_t^N(x)| \leq e_N, \) where
\[
e_N(t) = \begin{cases} 
\sum_{p=0}^{N+1} |c_p(t)| & \text{if } t \text{ is even}, \\
\sum_{p=0}^{N+1} |s_p(t)| & \text{if } t \text{ is odd}.
\end{cases}
\]

It now remains to provide bounds on \( e_N(t) \) in terms of \( t \) and \( N \). We note from Eq. B4 that
\[
\forall p > t/\pi, \quad |c_p(t)| \leq 2 \sum_{k=1}^{t/2} \frac{1}{(p^2 \pi^2)^k} \prod_{i=0}^{2k-2} (t - i) \leq 2 \sum_{k=1}^{\infty} \frac{t^{2k-1}}{(p^2 \pi^2)^k} = \frac{2t}{p^2 \pi^2 - t^2}. \quad (B7)
\]

A similar bound holds for \( |s_p(t)| \):
\[
\forall p > t/\pi, \quad |s_p(t)| \leq 2 \sum_{k=1}^{(t+1)/2} \frac{1}{((p + 1/2)^2 \pi^2)^k} (t - i) \leq 2 \sum_{k=1}^{\infty} \frac{t^{2k-1}}{(p^2 \pi^2)^k} \quad (B8)
\]

Consequently, it then follows that
\[
\forall N > t/\pi, \quad e_N(t) \leq 2 \sum_{p=N+1}^{\infty} \frac{t}{(p \pi)^2} \leq \frac{2t}{\pi^2} \leq \frac{2t}{\pi^2} \quad (B9)
\]

To ensure that \( e_N(t) \) is smaller than a given precision \( \varepsilon \), we can then choose \( N \) to be
\[
N = \frac{t}{\pi \tanh(\pi \varepsilon/2)} \geq \frac{2t}{\pi^2 \varepsilon}. \quad (B10)
\]

We point out that for this estimate to be correct, the chosen \( N \) should also be larger than \( t/\pi \) (Eq. B9), which is implied by Eq. B10 if the precision \( \varepsilon \) is smaller than \( 2/\pi \) and we obtain the estimate provided in lemma 10.

Finally, we compute the \( l_1 \) norm of the coefficients \( c_p(t) \) and \( s_p(t) \). We note that \( (-1)^p c_p(t) \geq 0 \) and \( (-1)^p s_p(t) \geq 0 \) for all \( p \geq 0 \). This is easily seen as follows --- from Eq. B3, using integration by parts it follows that
\[
c_p(t) = 2(-1)^p \frac{t}{p^2 \pi^2} - \frac{t(t - 1)}{p^2 \pi^2} c_p(t - 2) \\
s_p(t) = 2(-1)^p \frac{t}{(p + 1/2)^2 \pi^2} - \frac{t(t - 1)}{(p + 1/2)^2 \pi^2} s_p(t - 2). \quad (B11a)
\]

Furthermore, from Eq. B3, it also follows that
\[
|c_p(t)| \leq 2 \int_0^1 t^i x^t dx = \frac{2}{t + 1} \forall t \geq 0, \quad (B12)
\]

from which it follows that \( 2t \geq |(t - 1)c_p(t - 2)| \) and \( 2t \geq |t(t - 1)s_p(t - 2)| \). Together with Eq. B11a, it follows that \( (-1)^p c_p(t) \geq 0 \) and \( (-1)^p s_p(t) \geq 0 \) for all \( p \geq 0 \). Therefore,
\[
\sum_{p=0}^{\infty} |c_p(t)| = \sum_{p=0}^{\infty} c_p(t)(-1)^p = \sum_{p=0}^{\infty} c_p(t) \cos p \pi = 1 \\
\sum_{p=0}^{\infty} |s_p(t)| = \sum_{p=0}^{\infty} s_p(t)(-1)^p = \sum_{p=0}^{\infty} s_p(t) \sin \left(\frac{(2p + 1)\pi}{2}\right) = 1.
\]

Therefore, the \( l_1 \)-norm of \( [c_0(t), c_1(t) \ldots c_N(t)] \) and \( [s_0(t), s_1(t) \ldots s_N(t)] \) is less than 1.
