arXiv:1806.10236v4 [quant-ph] 5 May 2020

Product Decomposition of Periodic Functions in Quantum Signal Processing

Jeongwan Haah

Microsoft Quantum and Microsoft Research, Redmond, Washington, USA

20 September 2019

We consider an algorithm to approximate complex-valued periodic functions $f(e^{i\theta})$ as a matrix element of a product of $SU(2)$-valued functions, which underlies so-called quantum signal processing. We prove that the algorithm runs in time $\mathcal{O}(N^3 \text{polylog}(N/\epsilon))$ under the random-access memory model of computation where N is the degree of the polynomial that approximates f with accuracy ϵ; previous efficiency claim assumed a strong arithmetic model of computation and lacked numerical stability analysis.

1 Introduction

Quantum signal processing [1–3] refers to a scheme to construct an operator V from a more elementary unitary W where $V = \sum_\theta f(e^{i\theta}) |\theta\rangle \langle\theta|$ and $W = e^{i\theta} |\theta\rangle \langle\theta|$ share the eigenvectors but the eigenvalues of V are transformed by a function f from those of W. The transformation requires only one ancilla qubit, and is achieved by implementing control-W and control-$W^\dagger$, interspersed by single-qubit rotations on the control, and final post-selection on the control.[1] This technique produced gate-efficient quantum algorithms for, e.g., Hamiltonian simulations, which is asymptotically optimal when the Hamiltonian is sparse and given as a blackbox, or as a linear combination of oracular unitaries [5, 6]. Furthermore, this technique with rigorous error bounds appears to be useful and competitive even for explicitly described, rather than oracular, local Hamiltonian simulation problems [7, 8]. It is also promised to be useful in solving linear equations [3, 4, 9].

However, in quantum signal processing the classical preprocessing to find interspersing single-qubit rotations for a given transformation function f has been so numercially unstable that it has been unclear whether it can be performed efficiently. In fact, Ref. [7, App. H.3] reports that the computation time is "prohibitive" to obtain sequences of length $\gtrsim 30$ of interspersing unitaries for Jacobi-Anger expansions that we explain in Section 5. The true usefulness of quantum signal processing hinges upon the ability to compute long sequences of interspersing single-qubit rotations.

It has been asserted that there exists a polynomial time classical algorithm in Refs. [2, 4], but these work do not consider numerical instability. If a computational model assumes

[1] Sometimes it is possible to avoid controlled version of W [3, 4], but we contend ourselves with this implementation for its simplicity in presentation. The result of this paper is applicable for the ancilla-free variant; the only change one may have to do is to replace a variable $e^{i\theta/2}$ with $e^{i\theta}$.

that any real arithmetic with arbitrarily high precision can be done in unit time, then an unavoidable conclusion is that not only can one factor integers in time that is linear in the number of digits [10], but also solve NP-hard problems in polynomial time [11].[2] In a seemingly mundane problem involving real numbers, the number of required bits during the computation can be a priori very large. For example, it is still an open problem whether one can decide the larger between $\sum_{j=1}^{k} \sqrt{a_j}$ and $\sum_{j=1}^{k} \sqrt{b_j}$, which are sums of square roots of positive integers a_j, b_j that are smaller than n, in time $\operatorname{poly}(k \log n)$ on a Turing machine; see e.g. [12, 13] for recent results.

The numerical instability of previous methods may be attributed to expansions of large degree polynomials that are found by roots of another polynomial. Crudely speaking, there are two problems in this approach. First, the polynomial expansions can be regarded as the computation of convolutions,[3] which, when naively iterated, may suffer from numerical instability. Second, although the root finding is a well-studied problem, to use the roots to construct another polynomial one has to understand the distribution of the roots to keep track of the loss of precision. These problems were not addressed previously.

Here, we refine a classical algorithm to find interspersing single-qubit rotations and bound the number of required bits in the computation for a desired accuracy to a final answer. We conclude that the classical preprocessing can indeed be done in polynomial time on a Turing machine. We generally adopt the methods of Ref. [2], but make manipulations easier by avoiding trigonometric polynomials. Some generalizations are obtained with simpler calculations. For the numerical stability and analysis, our algorithm avoids too small numbers by sacrificing approximation quality in the initial stage, and replaces polynomial expansions by a Fourier transform. These modifications enable us to handle the problems that are mentioned above. However, it should be made clear that our refinement also requires high precision arithmetic. Specifically, we show that $\mathcal{O}(N \log(N/\epsilon))$ bits of precision during the execution of our algorithm is sufficient to produce a reliable final decomposition, where N is the degree of the polynomial that approximates a given transformation function f of eigenvalues up to additive error ϵ. Previously no such bound was known. On a sequential (rather than parallel) random access machine, our algorithm runs in time $\mathcal{O}(N^3 \operatorname{polylog}(N/\epsilon))$. In our rudimentary numerical experiment we were able to produce a sequence of length over 2000 for the Jacobi-Anger expansion on a laptop by a few hours of computation.[4]

[2] The basic idea in Ref. [11] is to relate the expansion of a Boolean formula in a conjunctive form into a disjunctive form with the expansion of a polynomial from its factors. Then, the satisfiability translates to the positiveness of certain coefficients in the expansion. The variable in the polynomial is set to a sufficiently large number, effectively encoding the entire polynomial in a single big number. An important elementary fact that underlies the power of arithmetic model is that we only need $O(n)$ compositions of squaring operation $x \mapsto x^2$ to reach a 2^n-bit number.

[3] A polynomial $\sum_j a_j t^j$ can be identified with a "coefficient function" $j \mapsto a_j$. The coefficient function of the product of two polynomials is the convolution of the two coefficient functions.

[4] Note added 5 May 2020: This was based on an implementation on Wolfram Mathematica. An implementation in a lower level programming language can be found at https://github.com/microsoft/Quantum-NC/tree/master/src/simulation/qsp with a sample application (https://github.com/microsoft/Quantum-NC/tree/master/samples/simulation/qsp). We have observed that the numerical instability is not harmful for the Jacobi-Anger expansion; the number of bits of precision that we used in a Mathematica implementation was much more than necessary. The reduction in the number of bits of precision and the lower level language implementation, have expedited the calculation by orders of magnitude. Empirically the time complexity appeared to be quadratic in N for the Jacobi-Anger expansion.

We will start by reviewing quantum signal processing in the next section, and then develop an algorithm and analyze it. In two short sections later we provide self-contained treatment of polynomial approximations for Hamiltonian simulation and matrix inversion problems. The section on Hamiltonian simulation contains some running time data of our algorithm. Throughout the paper, we use $U(1)$ to denote the group of all complex numbers of unit modulus. Sometimes we will refer to $U(1)$ as the unit circle. As usual, $i = \sqrt{-1}$, and $X = |1\rangle\langle 0| + |0\rangle\langle 1|, Y = i|1\rangle\langle 0| - i|0\rangle\langle 1|, Z = |0\rangle\langle 0| - |1\rangle\langle 1|$ are Pauli matrices.

2 Quantum Signal Processing

To understand how the eigenvalue transformation (signal processing) works, it is convenient to consider the action of control-W restricted to an arbitrary but fixed eigenstate $|\theta\rangle$ of W. The eigenvalue $e^{i\theta}$ associated with $|\theta\rangle$ is kicked back to the control qubit to induce a unitary $|0\rangle\langle 0| + e^{i\theta}|1\rangle\langle 1|$ on the control qubit. Conjugating the control-W by a single qubit unitary on the control qubit, we see that $|0\rangle$ and $|1\rangle$ can be any orthonormal basis vectors $|0'\rangle, |1'\rangle$ of the control qubit. If we allow ourselves to implement the inverse of the control-W, which is reasonable, we can also implement

$$\begin{aligned}&\Big(|0'\rangle\langle 0'| + e^{i\theta}|1'\rangle\langle 1'|\Big)\Big(|0''\rangle\langle 0''| + e^{-i\theta}|1''\rangle\langle 1''|\Big) \\ =&\left(e^{-i\theta/2}|0'\rangle\langle 0'| + e^{i\theta/2}|1'\rangle\langle 1'|\right)\left(e^{i\theta/2}|0''\rangle\langle 0''| + e^{-i\theta/2}|1''\rangle\langle 1''|\right)\end{aligned} \tag{1}$$

where $\{|0'\rangle, |1'\rangle\}$ and $\{|0''\rangle, |1''\rangle\}$ are arbitrary orthonormal bases.[5] When we alternate an even number of control-W and control-$W^\dagger$, this trick allows us to assume that an implementable unitary on the control qubit is a product of **primitive matrices**

$$E_P(t) = tP + t^{-1}(I - P) = tP + t^{-1}Q \tag{2}$$

where $t = e^{i\theta/2}$ and $P = I - Q$ is a projector of rank 1. Thus, an even number n of control-W and control-$W^\dagger$, together with an extra unitary E_0 independent of t, induces

$$F(t) = E_0 E_{P_1}(t) E_{P_2}(t) \cdots E_{P_n}(t) \tag{3}$$

on the control qubit. The product $F(t)$ can be thought of as an $SU(2)$-valued function over the unit circle in the complex plane. By the same formulas, we define $E_P(t)$ and $F(t)$ on the entire complex plane except the origin $t = 0$. Now if $\langle +|F(t)|+\rangle$ is close to $f(t^2 = e^{i\theta})$ for all $t \in U(1)$, then post-selection on $|+\rangle$ of the control qubit enacts V. Here, the choice of $|+\rangle$ is a convention, as it can be any other state due to E_0. The success probability of the post-selection depends on the magnitude of $f(e^{i\theta})$. A natural question is then what $F(t)$ is achievable in the form of Eq. (3). Note that it makes no difference to insert many unitaries that are independent of t in between $E_{P_j}(t)$'s, rather than a single E_0 at the front, because $UE_P(t)U^\dagger = E_{UPU^\dagger}(t)$ for any unitary U. The answer to the achievability question turns out to be quite simple, as we show in the next section.

[5] The square root function $e^{i\theta} \mapsto e^{i\theta/2}$ has a branch cut, but it hardly matters to us as long as we are consistent that $e^{-i\theta/2}$ denotes the inverse of the square root.

3 Polynomial functions $U(1) \to SU(2)$

Definition 1. For any integer $n \geq 0$, let $\mathcal{P}_n$ be the set of all Laurent polynomials $F(t) = \sum_{j=-n}^{n} C_j t^j$ in t with coefficients C_j in 2-by-2 complex matrices, such that $F(\eta) \in SU(2)$ for all complex numbers η of unit modulus. We say that $F(t) \in \mathcal{P}_n$ has **degree** n if $C_n \neq 0$ or $C_{-n} \neq 0$. We define $\mathcal{E}_n$ to be the subset of $\mathcal{P}_n$ consisting of all $F(t)$ where the exponents of t in $F(t)$ belong to $\{-n, -n+2, -n+4, \ldots, n-2, n\}$. Note that $\mathcal{P}_0 = \mathcal{E}_0 = SU(2)$, and for any orthogonal projector P we have $E_P(t) \in \mathcal{E}_1$. We define $F^\dagger(t)$ to be $\sum_j C_j^\dagger t^j$.[6] In a set-theoretic notation, the definitions are as the following.

$$\mathcal{P}_n = \left\{ F(t) = \sum_{j=-n}^{n} C_j t^j \in \mathrm{Mat}(2;\mathbb{C})[t, t^{-1}] \,\middle|\, \forall \eta \in \mathbb{C}, |\eta| = 1 \Rightarrow F(\eta) \in SU(2) \right\} \tag{4}$$

$$\mathcal{E}_n = \left\{ F(t) = \sum_{j=-n}^{n} C_j t^j \in \mathcal{P}_n \,\middle|\, \forall k \in 2\mathbb{Z}+1, C_{-n+k} = 0 \right\} \tag{5}$$

Note that for any $F(t) \in \mathcal{P}_n$, we have $F(t)F^\dagger(1/t) = F^\dagger(1/t)F(t) = I$; this is true for every t on the unit circle, an infinite set, and any (Laurent) polynomial is determined by its values on an infinite set.

Theorem 2. *Any n-fold product $E_{P_1}(t) \cdots E_{P_n}(t)$ belongs to $\mathcal{E}_n$. Conversely, every $F(t) \in \mathcal{E}_n$ of degree n has a unique decomposition into primitive matrices and a unitary, as in Eq. (3). If $F(t) = C_{-n}t^{-n} + \cdots + C_n t^n$, then $P_n = C_n^\dagger C_n / \operatorname{Tr}(C_n^\dagger C_n)$.*

This completely characterizes polynomial functions $U(1) \to SU(2)$ and covers all previous results on "achievable functions" in quantum signal processing [2, 4]. Indeed, for any Laurent polynomial function $U(1) \ni z \mapsto F(z) \in SU(2)$, the Laurent polynomial function $t \mapsto F(t^2)$ belongs to $\mathcal{E}_n$ for some n and has a unique product decomposition of the theorem. Our version is slightly more general since previous results implicitly assume that $\operatorname{Tr}(P_j Z) = 0$.

Proof. The first statement is trivial by definition. The proof of the converse is by induction in n where the base case $n = 0$ is trivial. The induction step is proved as follows. We are going to prove that *for any $F(t) \in \mathcal{E}_n$ of degree $n > 0$ there exists a unique $E_K(t)$ such that* $F(t)E_K(t) \in \mathcal{E}_{n-1}$.[7]

Consider $F(t) = \sum_{j=-n}^{n} C_j t^j$ as a 2-by-2 matrix of four Laurent polynomials. The defining property $\det F(t) = 1$ holds for infinitely many values of t, and therefore it should hold as a polynomial equation. Taking the leading term, we have $t^{2n} \det C_n + (\text{lower order terms}) = 1$. Similarly, taking the leading term in t^{-1}, we have $t^{-2n} \det C_{-n} + (\text{higer order terms}) = 1$. Hence,

$$\det C_n = 0 = \det C_{-n}. \tag{6}$$

[6] For an indeterminant t, we do *not* define $(F(t))^\dagger$. For a unimodular complex number η, the hermitian conjugate of the unitary $F(\eta)$ is $(F(\eta))^\dagger = F^\dagger(\bar{\eta}) = F^\dagger(1/\eta)$ which is *not* equal in general to $F^\dagger(\eta)$.

[7] A reader might find it unusual that the degree of a polynomial is decreasing under multiplication, but in the algebra of matrices two nonzero matrices may multiply to vanish.

Similarly, from the equation $F^\dagger(1/t)F(t) = I = F(t)F^\dagger(1/t)$ we have $t^{2n}C_{-n}^\dagger C_n + \mathcal{O}(t^{2n-1}) = I = t^{2n}C_n C_{-n}^\dagger + \mathcal{O}(t^{2n-1})$ and hence

$$C_{-n}^\dagger C_n = 0 = C_n C_{-n}^\dagger. \tag{7}$$

Since the degree of $F(t)$ is n, at least one of C_n and C_{-n} is nonzero. Suppose $C_n \neq 0$. Let K be a rank-1 projector such that $C_n K = 0$, and let $L = I - K$. Such K is unique since C_n is a two-by-two matrix of rank one; the singular value decomposition of C_n is $|a\rangle\langle b|$ for some unnormalized vectors $|a\rangle, |b\rangle$, and K has to annihilate $|b\rangle$. Then, we claim that $F(t)(tK + t^{-1}L) \in \mathcal{E}_{n-1}$. Indeed, expanding the left-hand side we have

$$t^{-n-1}C_{-n}L + t^{-n+1}(C_{-n}K + C_{-n+2}L) + \cdots + t^{n-1}(C_n L + C_{n-2}K) + t^{n+1}C_n K. \tag{8}$$

(This is the only place we use $\mathcal{E}$ instead of $\mathcal{P}$.) If $C_{-n} = 0$, this implies the claim. If $C_{-n} \neq 0$, then, considering the singular value decomposition of C_{-n}, we have $K \propto C_{-n}^\dagger C_{-n}$ and therefore $C_{-n}L = 0$, implying the claim. The case $C_{-n} \neq 0$ is completely parallel.

Actually, for any $F(t) \in \mathcal{E}_n$ of degree n, $C_n \neq 0$ if and only if $C_{-n} \neq 0$: C_n is a product of n rank-one operators $E_0 P_1 \cdots P_n = E_0 |p_1\rangle\langle p_1|p_2\rangle \cdots \langle p_{n-1}|p_n\rangle\langle p_n|$, where $\langle p_j|p_{j+1}\rangle \neq 0$ for all j, and this implies $Q_j Q_{j+1} = (I - P_j)(I - P_{j+1}) \neq 0$ for all j, which is to say that $C_{-n} = E_0 Q_1 \cdots Q_n \neq 0$. □

3.1 Parity constraints

Any member of $SU(2)$ can be written as $aI + biX + ciY + diZ$ where the real numbers a, b, c, d satisfy $a^2 + b^2 + c^2 + d^2 = 1$, and this decomposition is unique. (The group $SU(2)$ is identified with the group of all unit quaternions.) Thus, a member $F(z) \in \mathcal{P}_n$ can be written uniquely as $F(z) = a(z)I + b(z)iX + c(z)iY + d(z)iZ$. Here, $a(z), b(z), c(z), d(z)$ are Laurent polynomials such that $a(z)^2 + b(z)^2 + c(z)^2 + d(z)^2 = 1$, and each takes real values on $U(1)$.

Recall that under the standard representation of Pauli matrices, Z is diagonal, and X, Y are off-diagonal. Suppose an $SU(2)$-valued function $\theta \mapsto F(e^{i\theta}) = a(e^{i\theta})I + b(e^{i\theta})iX + c(e^{i\theta})iY + d(e^{i\theta})iZ$ has even functions (reciprocal in $t = e^{i\theta}$) in the diagonal and odd (anti-reciprocal in $t = e^{i\theta}$) in the off-diagonal. That is, $a(t) = a(1/t)$, $d(t) = d(1/t)$, $b(t) = -b(1/t)$, and $c(t) = -c(1/t)$. We claim that if $F(t)$ is such an element of $\mathcal{E}_n$, then the primitive matrix $E_{P_n}(t)$ factored from $F(t)$ by Theorem 2 has a property that $\mathrm{Tr}(ZP_n) = 0$. Since for any projector $P = \frac{1}{2}(I + p_x X + p_y Y + p_z Z)$ where $(p_x, p_y, p_z) \in \mathbb{R}^3$ has norm 1, the primitive matrix $E_P(t)$ equals $tP + t^{-1}(I - P) = \frac{t+t^{-1}}{2}I + \frac{t-t^{-1}}{2}(p_x X + p_y Y + p_z Z)$, the condition $\mathrm{Tr}(ZP) = 0$ is to say $p_z = 0$. This implies that $E_{P_n}(t)$ has reciprocal diagonal and anti-reciprocal off-diagonal. To prove the claim we observe that

$$\begin{aligned} Z &= F(t)F^\dagger(1/t)Z \\ &= F(t)(a(1/t) - b(1/t)iX - c(1/t)iY - d(1/t)iZ)Z \\ &= F(t)(a(t) + b(t)iX + c(t)iY - d(t)iZ)Z \\ &= F(t)Z(a(t) - b(t)iX - c(t)iY - d(t)iZ) \\ &= F(t)ZF^\dagger(t). \end{aligned} \tag{9}$$

It follows that $0 = \mathrm{Tr}(Z) = \mathrm{Tr}(F(t)ZF^\dagger(t)) = t^{2n}\,\mathrm{Tr}(C_n Z C_n^\dagger) + \cdots + t^{-2n}\,\mathrm{Tr}(C_{-n}ZC_{-n}^\dagger)$ as a polynomial in t, and hence $\mathrm{Tr}(C_n^\dagger C_n Z) = 0 = \mathrm{Tr}(C_{-n}^\dagger C_{-n} Z)$ when $n > 0$. The matrix

$C_{\pm n}^\dagger C_{\pm n}$ is proportional to the projector P_n or $I - P_n$ in the factor $E_{P_n}(t)$ as shown in Theorem 2, and therefore we have $\mathrm{Tr}(ZP_n) = 0$.

Moreover, it then follows that $F(t)E_{P_n}^\dagger(1/t)$ also has reciprocal diagonal and anti-reciprocal off-diagonals. Therefore, the unique projectors $P_1, \ldots, P_n$ in the decomposition, which define the primitive matrices $E_{P_j}(t)$, have zero Z-component. This means that all projectors P_j are of form

$$P_j = e^{iZ\phi_j/2} \left|+\right\rangle \left\langle+\right| e^{-iZ\phi_j/2} \tag{10}$$

where $\phi_j \in \mathbb{R}$ is some angle. In fact, Ref. [2] exclusively considered $E_P(t)$ of this form. This is contrasted to the general case where P_j is identified with a point on the Bloch sphere. The constraint $\mathrm{Tr}(ZP_j) = 0$ forces P_j to lie on the equator of the Bloch sphere.

Note that E_0, the residual $SU(2)$ factor in the decomposition, has generally nonzero Z-component. Since $E_P(1) = I$ for any projector P, we know $E_0 = F(1) = a(1)I + b(1)iX + c(1)iY + d(1)iZ$, but $b(1) = c(1) = 0$ due to their anti-reciprocity. This implies that $E_0 = e^{iZ\phi_0/2}$ for some angle ϕ_0. Hence, under the parity constraint of this subsection, $F(t) \in \mathcal{E}_n$ is uniquely specified by $n+1$ angles $\phi_0, \phi_1, \ldots, \phi_n$. Note that if $d(1)^2 = 1 - a(1)^2 = \mathcal{O}(\epsilon)$, then $d(1) = \mathcal{O}(\sqrt{\epsilon})$ and $\|E_0 - I\| = \mathcal{O}(\sqrt{\epsilon})$. Hence, there would be quadratic loss in accuracy to omit E_0 (i.e., to set $\phi_0 = 0$), even though $a(1) \approx 1$ suggests that one would not need E_0.

3.2 Complementing polynomials

Quantum signal processing does not use $F(t)$ itself, but rather a certain matrix element of it. Hence, it is important to know what a matrix element can be. Let us first introduce classes of Laurent polynomials.

Definition 3. A (Laurent) polynomial with real $\mathbb{R}$ coefficients is called a **real** (Laurent) polynomial. The **degree** of a Laurent polynomial is the maximum absolute value of the exponent of the variable whose coefficient is nonzero. A Laurent polynomial $f(z)$ is **reciprocal** if $f(z) = f(1/z)$, or **anti-reciprocal** if $f(z) = -f(1/z)$. A Laurent polynomial function $f : \mathbb{C} \setminus \{0\} \to \mathbb{C}$ is **real-on-circle** if $f(z) \in \mathbb{R}$ for all $z \in \mathbb{C}$ of unit modulus. A real-on-circle Laurent polynomial $f(z)$ is **pure** if $f(z)$ is real reciprocal or $if(z)$ is real anti-reciprocal.

The term "pure" is because *a Laurent polynomial $f(z)$ with complex coefficients is real-on-circle if and only if both*

$$f_+(z) := \frac{f(z) + f(1/z)}{2} \text{ and } f_-(z) := \frac{f(z) - f(1/z)}{2i} \tag{11}$$

are real Laurent polynomials. (*Proof*: Write $f(e^{i\theta}) = \sum_j a_j e^{ij\theta}$, with the complex conjugate being $\sum_j \overline{a_j} e^{-ij\theta} = \sum_j \overline{a_{-j}} e^{ij\theta}$. Thus, $a_j = \overline{a_{-j}}$, and the claim follows.) This simply rephrases the fact that a real-valued function $\theta \mapsto f(e^{i\theta})$ has a trigonometric function series with real coefficients. Hence, for any real-on-circle Laurent polynomial $f(z)$, it is real if and only if it is reciprocal. In addition, a real and reciprocal Laurent polynomial is real-on-circle. That is, among the three properties, real, real-on-circle, and reciprocal, any two imply the third. Note that a real-on-circle Laurent polynomial is not necessarily real, and a real Laurent polynomial is not necessarily real-on-circle. Also, note that real-on-circle Laurent polynomials form an algebra over the real numbers.

We are now ready to state a sufficient condition under which a complex polynomial qualifies to be a matrix element of some $F(t) \in \mathcal{P}_n$. We think of $a(z)$ and $b(z)$ below as the real and imaginary parts of a complex function, respectively. A reader might want to compare the following lemma with the first paragraph of Section 3.1.

Lemma 4. *Let $a(z)$ and $b(z)$ be real-on-circle Laurent polynomials of degree at most n such that $a(\eta)^2 + b(\eta)^2 < 1$ for all $\eta \in U(1)$. If $a(z)^2 + b(z)^2$ is reciprocal (e.g., $a(z)$ and $b(z)$ are pure), then there exist pure real-on-circle Laurent polynomials $c(z) = c_+(z)$ and $d(z) = id_-(z)$ of degree at most n such that $a(z)^2 + b(z)^2 + c(z)^2 + d(z)^2 = 1$.*

The conditions in the lemma on reciprocity are due to a technical reason in the proof. If there were some other reason under which one can guarantee the existence of the complementing polynomials, it would be possible to use them in the algorithm below, and the scope of the input functions in our algorithm would be enlarged; the existence of the complementing polynomials is more important than the reciprocity constraints. However, we note that the reciprocity conditions are not severe restrictions since any periodic function is the sum of an even and an odd function, and one can "combine" two functions by "flexible" quantum signal processing [14].

Proof. The Laurent polynomial $1 - a(z)^2 - b(z)^2$ is reciprocal real of degree n' that is at most $2n$; the leading terms of $a(z)^2$ and $b(z)^2$ might cancel each other so that $n' < 2n$. Due to the reciprocity, there are $2n'$ roots in total with multiplicity taken into account and any root r must come in a pair (z, z^{-1}) where one is inside the unit disk and the other outside the unit disk, but neither is on the unit circle. We collect all the roots inside the unit disk:

$$\mathcal{D} = \Big[r \in \mathbb{C} \ : \ 1 - a(r)^2 - b(r)^2 = 0, \ |r| < 1 \Big] . \tag{12}$$

This is a list rather than a set as we take the multiplicities into account; $\mathcal{D}$ has exactly n' elements. Consider a factor of $1 - a(z)^2 - b(z)^2$:

$$e(z) = z^{-\lfloor n'/2 \rfloor} \prod_{r \in \mathcal{D}} (z - r). \tag{13}$$

The monomial in front of the product is to balance the greatest exponent of z with the least exponent; the degree of $e(z)$ is $\lceil n'/2 \rceil$. The list $\mathcal{D}$ is closed under complex conjugation due to the reality of $1 - a(z)^2 - b(z)^2$, and hence $e(z)$ is a real Laurent polynomial.

Then, the product $e(z)e(1/z)$ is real reciprocal and has degree n'.[8] Now the two Laurent polynomials $e(z)e(1/z)$ and $1 - a(z)^2 - b(z)^2$ have the same roots. Therefore they differ by a factor of cz^k for some nonzero number c and an integer k, but the reciprocity fixes $k = 0$ and the reality puts c into $\mathbb{R}$. That is,

$$\alpha = \frac{1 - a(z)^2 - b(z)^2}{e(z)e(1/z)} \in \mathbb{R}.$$

Evaluating this expression at $z = 1$, we see that α is positive. Thus, we finish the proof by observing

$$1 - a(z)^2 - b(z)^2 = \alpha e(z)e(1/z) = \left(\frac{e(z) + e(1/z)}{2} \sqrt{\alpha} \right)^2 + \left(\frac{e(z) - e(1/z)}{2i} \sqrt{\alpha} \right)^2 . \tag{14}$$

[8] The degree of a Laurent polynomial in our definition is only subadditive under multiplication of two Laurent polynomials. For example, the product $(z - 1)(z^{-1} - 2)$ has degree one.

Both the reciprocal ($c(z)$) and anti-reciprocal ($d(z)$) combinations have degree $\le \lceil n'/2 \rceil \le n$. □

Note that given $a(z)$ and $b(z)$ the complementing Laurent polynomials $c(z), d(z)$ are not unique in general. As long as the joint of a conjugate-closed list $\mathcal{D}$ and its reciprocal $[1/r \in \mathbb{C} \ : \ r \in \mathcal{D}]$ is the list of all the roots of $1 - a(z)^2 - b(z)^2$, we can construct $c(z)$ and $d(z)$ satisfying the conditions in the lemma.

4 Efficient implementation with bounded precision

In this section we consider an algorithm to find interspersing single-qubit unitaries given a complex function $A(e^{i\varphi}) + iB(e^{i\varphi})$. The algorithm consists of two main parts: first, we have to find an $SU(2)$-valued function of φ such that a particular matrix element is the input function. It suffices to find a good approximation. Second, we have to decompose the $SU(2)$-valued function into a product of primitive matrices. We have already given constructive proofs for both the steps, but we tailor the construction so that numerical error is reduced and traceable. We will outline our algorithm first, deferring certain details to the next subsection. The computational complexity will be analyzed subsequently.

Input: A real parameter $\epsilon \in (0, \frac{1}{100})$, a list of $2N + 1$ complex numbers ζ_k $(k = -N, \ldots, N)$ specified using at most $\log_2(100N/\epsilon)$ bits in the floating point representation, and two bits p_{re} and p_{im}.

Here, the list is the Fourier coefficients ζ_k for frequencies between $-N$ and N of a complex-valued 2π-periodic function $A(e^{i\varphi}) + iB(e^{i\varphi}) = \sum_{k=-N}^{N} \zeta_k e^{ik\varphi}$ subject to conditions that (i) each of real-valued functions $A(e^{i\varphi})$ and $B(e^{i\varphi})$ has definite parity (even or odd parity as functions of φ), recorded in the two bits $p_{\mathrm{re}}, p_{\mathrm{im}}$, and (ii) $A(e^{i\varphi})^2 + B(e^{i\varphi})^2 \le 1$ for any real φ.

The function $\varphi \mapsto A(e^{i\varphi}) + iB(e^{i\varphi})$ must be sufficiently close to an ultimate target function, where the latter is, strictly speaking, not a part of the input for us. This approximation has nothing to do with the algorithm below, but should be analyzed independently for each quantum signal processing problem.

Output: A unitary $E_0 \in SU(2)$ and an ordered list of 2×2 hermitian matrices $P_1, \ldots, P_{2n}$ where $n \le N$.

Here, each P_m is a (approximate) rank-one projector represented by $\Omega(\log(N/\epsilon))$ bits of precision and defines the primitive matrix $E_m(t) = tP_m + t^{-1}(I - P_m)$. When $t = e^{i\phi} \in U(1)$, it holds that $\left| A(t^2) + iB(t^2) - \langle + | \, E_0 E_1(t) \cdots E_{2n}(t) \, | + \rangle \right| \le 30\epsilon$.

Time: The computational time complexity is $\mathcal{O}(N^3 \mathrm{polylog}(N/\epsilon))$ on a random-access memory machine.

Alternatively, an input may be a list of function values from which Fourier coefficients can be computed. Having Fourier coefficients for frequencies between $-N$ and N is equivalent to having a Laurent polynomial $A(z) + iB(z)$ of degree at most N.

Our Lemma 4 would allow more general input functions where the real and imaginary parts are not necessarily of definite parity, but we restrict our algorithm to inputs of definite parity, since we require $A^2 + B^2$ to be of definite parity which may not be satisfied after the rational approximation in Step 1 below if individual components are not of definite parity. As mentioned before, this parity constraint is not too restrictive since quantum signal processing can be easily adopted to handle functions of indefinite parity [14].

4.1 Algorithm

1. Compute rational real-on-circle Laurent polynomials $a(z), b(z)$ from $(1-10\epsilon)A(z), (1-10\epsilon)B(z)$ by taking rational number approximation for each coefficient up to an additive error of ϵ/N. If a coefficient is smaller than ϵ/N in magnitude, then it must be replaced by zero. The polynomials $a(z)$ and $b(z)$ are stored as lists of rational numbers (not floating point numbers). This step in general decreases the Laurent polynomial degree from N to n since some small coefficients may be approximated by zero.

2. To additive accuracy 2^{-R} where $R \gtrsim 2N \log_2(N/\epsilon)$, find all roots of $1 - a(z)^2 - b(z)^2$. See Eq. (37) for a rigorous bound on R. The roots are stored as floating point numbers. From now on all real arithmetic will be performed using R-bit floating point numbers.

3. Evaluate the complementary polynomials computed from the roots of Step 2 according to Lemma 4 at points of T where
$$T := \{e^{2\pi i k/D} \mid k = 1, \ldots, D\} \tag{15}$$
and D is a power of 2 that is larger than $2n+1$. One should *not* expand $c(z), d(z)$ before evaluation, but should substitute numerical values for z with accuracy 2^{-R} in the factorized form of $e(z)$ in Eq. (13), and then read off the real $(c(z))$ and imaginary $(d(z))$ parts.

4. Set $F(z) = a(z)I + b(z)iX + c(z)iY + d(z)iZ$. Compute the discrete fast Fourier transform of the function value list to obtain
$$C_{2j}^{(2n)} = \int_0^{2\pi} \frac{\mathrm{d}\theta}{2\pi}\, e^{-ij\theta} F(e^{i\theta}) \tag{16}$$
for $j = -n, -n+1, \ldots, n-1, n$. (In exact arithmetic, we would have $F(z) = \sum_{j=-n}^{n} C_{2j}^{(2n)} z^j$.)

5. Set $F^{(2n)}(t) = \sum_{j=-n}^{n} C_{2j}^{(2n)} t^{2j}$. For $m = 2n, 2n-1, \ldots, 2, 1$ sequentially in decreasing order, (i) compute a primitive matrix $E_m(t)$ by
$$E_m(t) = tP_m + t^{-1}(I - P_m) = t(I - Q_m) + t^{-1}Q_m \tag{17}$$
$$\text{where} \quad P_m = \frac{C_m^{(m)\dagger} C_m^{(m)}}{\mathrm{Tr}(C_m^{(m)\dagger} C_m^{(m)})}, \quad Q_m = \frac{C_{-m}^{(m)\dagger} C_{-m}^{(m)}}{\mathrm{Tr}(C_{-m}^{(m)\dagger} C_{-m}^{(m)})},$$
and (ii) compute the coefficient list of $F^{(m-1)}(t) = F^{(m)}(t) E_m^\dagger(1/t)$ by
$$C_k^{(m-1)} = C_{k-1}^{(m)} Q_m + C_{k+1}^{(m)} P_m \tag{18}$$
where $k = -m+1, -m+3, \ldots, m-3, m-1$.

6. Output $E_0 = C_0^{(0)}$ and $P_1, \ldots, P_{2n}$ using $\log_2(20N/\epsilon)$-bit floating numbers. Then,
$$\langle +| E_0 E_1(t) \cdots E_{2n}(t) |+\rangle \tag{19}$$
is 30ϵ-close to $A(t^2) + iB(t^2)$ for all $t \in U(1)$.

4.2 Further details and analysis

Step 1. Let the rational approximation be performed by truncating the binary expressions of real numbers. To inherit parities, the rational approximation should be done only for terms with nonnegative powers of z, from which we should infer the negative power terms. Then, $a(z)$ and $b(z)$ satisfy

(i) $a(z)^2 + b(z)^2$ is a real reciprocal Laurent polynomial,

(ii) $|a(z) + ib(z) - A(z) - iB(z)| \le 26\epsilon$ for $z \in U(1)$,

(iii) $a(z)^2 + b(z)^2 \le 1 - \epsilon$ for $z \in U(1)$, and

(iv) every nonzero coefficient in $a(z)$ or $b(z)$ has magnitude $\ge \epsilon/N$.

The first and the fourth conditions are clear by construction. The second condition is because $|A(z)-a(z)| \le |A(z)-(1-10\epsilon)A(z)|+|(1-10\epsilon)A(z)-a(z)| \le 13\epsilon$ and similarly $|B(z)-b(z)| \le 13\epsilon$. The third is because $|a(z)+ib(z)| \le |(1-10\epsilon)(A(z)+iB(z))-a(z)+ib(z)|+(1-10\epsilon) \le 1-\epsilon$.

The reason we speak of rational Laurent polynomials is mainly for the convenience of analysis, as its evaluation can be made arbitrarily accurate since the coefficients of $a(z)$ and $b(z)$ are exact; each coefficient is stored as a rational number, a pair of integers, rather than a floating point number. In a concrete implementation of our algorithm, floating point numbers that are carefully handled may substitute rational numbers. If $\mu = f \times 2^d$ is a real number where $\frac{1}{2} \le f < 1$ and $d \in \mathbb{Z}$, then the rational approximation of μ may be $\tilde{\mu} = 0.b_1 b_2 \cdots b_p \times 2^d$ for some p where b_j are bits in the binary representation of f. Some care may be needed in order not to discard any bits of $\tilde{\mu}$ in arithmetic. For example, when $\tilde{\mu}$ is added to another floating number of $p' > p$ bits of precision, then $\tilde{\mu}$ should be mapped to a floating number of whatever needed bits of precision by padding zeros. In practice, this should cause hardly any complication since most high precision arithmetic libraries (e.g. The GNU Multiple Precision Arithmetic Library) treat inputs as exact numbers, but control the precision of the result according to other rules set by a user. The number of bits to represent all the rational coefficients is $\mathcal{O}(N \log(N/\epsilon))$.

Step 2. There exists a root-finding algorithm with computational complexity $\tilde{\mathcal{O}}(n^3 + n^2 R)$ under the assumption that all the roots have modulus at most 1 [15]. In our case, the rational Laurent polynomial $p(z) = 1 - a(z)^2 - b(z)^2$ does not satisfy the modulus condition; however, this is a minor problem. Every coefficient of $p(z)$ is the Fourier coefficient of the periodic function $p(e^{i\theta}) < 1$, and hence is bounded by 1. By the condition (iv) of Step 1, the leading coefficient of $p(z)$ is $\Omega((\epsilon/N)^2)$ in magnitude. (The reason is as follows. Since our rational approximation is by truncating binary expressions of real numbers, the denominator of any coefficient is a power of 2 and is at most $2^{\lceil \log_2(N/\epsilon) \rceil}$. Hence, $4^{\lceil \log_2(N/\epsilon) \rceil} p(z)$ has integer coefficients.) Say the polynomial $p(z) = 1 - a(z)^2 - b(z)^2$ has degree n', which may be less than $2n$ even if $a(z)$ and $b(z)$ have degree n. Converting $p(z)$ into a monic polynomial $q(z)$ (after multiplying by $z^{n'}$ and a normalization factor), we have $q(z) = z^{2n'} + \sum_{j=0}^{2n'-1} q_j z^j$ with $|q_j| \le \mathcal{O}((N/\epsilon)^2)$. Note that $q(z)$ has (exactly represented) rational coefficients. If $q(z_0) = 0$ with $|z_0| > 1$, then

$$|z_0|^{2n'} \le \sum_{j=0}^{2n'-1} |q_j||z_0|^j \le \mathcal{O}(N^2 n' |z_0|^{2n'-1}/\epsilon^2) \tag{20}$$

implying $|z_0| \le \mathcal{O}(N^3/\epsilon^2)$. (If $|z_0| \le 1$, this is trivial.) We can use the algorithm of Ref. [15] after rescaling z by a known factor. The overhead due to the potential loss of precision from the rescaling is negligible since $R = \Omega(N \log(N/\epsilon))$.

Step 3. We need to evaluate $e(z)$ of Eq. (13) that is defined by the roots found in Step 2. For $z \in U(1)$, the following Lemma 5 guarantees that any root of $1 - a(z)^2 - b(z)^2$ is at least $\epsilon/(4N^2)$-away (or $\Omega(1/N^2)$-away if $|1 - A(z)^2 - B(z)^2| = \mathcal{O}(\epsilon)$) from the unit circle. In particular, with the prescribed accuracy 2^{-R} there is no numerical ambiguity to determine whether a root is inside the unit disk. That is, the list $\mathcal{D}$ of Eq. (12) can be obviously computed under our bounded precision arithmetic.

Let us analyze the evaluation accuracy of $e(z)$ for $z \in U(1)$ more closely. When evaluating a linear factor $z - r$ of $e(z)$ where both z and r are accurate up to additive error 2^{-R}, the number of lost significant bits is $\mathcal{O}(\log(N/\epsilon))$ which is negligible compared to R. The function value of $e(z)$ is thus evaluated accurately up to relative error $2^{-R+\mathcal{O}(\log(N/\epsilon))}$. Hence, the function value of $c(z)$ (the real part of $e(z)\sqrt{\alpha}$) and $d(z)$ (the imaginary part of $e(z)\sqrt{\alpha}$) by Eq. (14) are determined up to additive error $2^{-R+\mathcal{O}(\log(N/\epsilon))}$.

More concretely but still loosely, let us assume $24N^2 2^{-R}\epsilon^{-1} \le 1/(16N)$. The relative error of a linear factor $z - r$ is at most $2 \cdot 2^{-R}(4N^2/\epsilon)$. The factor of $e(z)$ for the complex roots can be evaluated to relative error at most $3 \cdot 2 \cdot 2^{-R}(4N^2/\epsilon)$. There are at most N factors in $e(z)$, so the value of $e(z)$ is determined up to relative error $\delta = (1 + 24N^2 2^{-R}\epsilon^{-1})^N - 1 \le 48N^3 2^{-R}\epsilon^{-1} \le 1/8$. This in turn gives an upper bound $200N^3 2^{-R}\epsilon^{-1}$ on the additive error of the real part $c(z)$ and the imaginary part $d(z)$ since they have magnitude at most 1 on the unit circle.

Lemma 5. *If a real-on-circle Laurent polynomial $f(z)$ of degree $d \ge 1$ satisfies $0 < m \le f(z) \le M$ for all $z \in U(1)$, then every zero of f is at least $m/(4Md^2)$-away from $U(1)$.*

Proof. Pick any root z_0 and choose the closest point $u \in U(1)$ so that $f(z_0 = u + \eta) = 0$. We will lower bound the magnitude of η. If $|\eta| \ge 1/(2d)$, then there is nothing to prove, so we assume $|\eta| < 1/(2d)$. Since f is analytic except at $z = 0$, the Taylor series of f at u converges at $z_0 = u + \eta$.

$$0 = f(u+\eta) = \sum_{k \ge 0} \frac{f^{(k)}(u)}{k!}\eta^k. \tag{21}$$

Let us estimate the magnitude of derivatives at $u \in U(1)$. Since the coefficients a_j of the polynomial $f(u) = \sum_{j=-d}^{d} a_j u^j$ are the Fourier coefficients, meaning that a_j is a "weighted" average of the function values on the unit circle, we know $|a_j| \le M$. Thus

$$|f^{(k)}(u)| \le 2d \cdot M \cdot d(d+1)\cdots(d+k-1); \tag{22}$$

there are $2d$ terms (or one more if $k = 0$ in which case the inequality is true anyway) and the maximum absolute value of the exponent increases by at most 1 every time we differentiate due to the negative degree term. Therefore,

$$m \le |f(u)| \le \sum_{k \ge 1} 2dM \binom{d+k-1}{k} |\eta|^k = 2dM \left(\frac{1}{(1-|\eta|)^d} - 1 \right) \le 2dM \cdot 2d|\eta| \tag{23}$$

where the first inequality is the assumption, the second inequality is by rearranging Eq. (21) and applying triangle inequality, and in the last inequality we use the fact that $(1-x)^{-d}-1$ is a convex increasing function of positive x and valued at most 1 at $x = 1/(2d)$ for $d \geq 1$. □

Step 4. This is essentially expanding the polynomials $c(z)$ and $d(z)$ found in the root finding step, but we use the fast Fourier transform (FFT) for its better accuracy. It has been shown [16] that the FFT on a k-component input F where each component (that is assumed to be a complex number in [16]) is accurate to relative error δ, gives Fourier coefficients $\tilde{\hat{F}}_\omega$ with error

$$\max_\omega |\tilde{\hat{F}}_\omega - \hat{F}_\omega| \leq \mathcal{O}(k^{-1/2}\log k) \cdot \delta \sqrt{\frac{1}{k}\sum_\omega |\hat{F}_\omega|^2} \tag{24}$$

where $\hat{F}_\omega = k^{-1}\sum_{\ell=1}^{k} e^{i\ell\omega/k} F(e^{i\ell/k})$ is the true Fourier spectrum. (Note the normalization factor k^{-1} here.) In our case, F consists of 2-by-2 matrices, and Eq. (24) also holds with Frobenius or operator norms in place of absolute values. Since the input "vector" F in this Step is a list of unitary matrices, the root-mean-square factor is $\mathcal{O}(1)$, and the distinction between relative and absolute error is immaterial. By the analysis of Step 3 above, δ is $2^{-R+\mathcal{O}(\log(N/\epsilon))}$. Thus, the (additive) error δ_{2n} in any Fourier coefficient $C_{2j}^{(2n)}$ is at most $2^{-R+\mathcal{O}(\log(N/\epsilon))}$. The error here is the operator norm of the difference between the computed and the true $C_{2j}^{(2n)}$.

Crude and concrete bounds can be obtained more directly (without using Eq. (24)): The coefficients of $a(z)$ and $b(z)$ are exactly known. Those of $c(z)$ and $d(z)$ are computed from the value table of $e(z)$ from the previous Step, which has entry-wise additive error $\delta \leq 48N^3 2^{-R}\epsilon^{-1}$. The (slow) discrete Fourier transform on a k-component vector v is the matrix multiplication by $V = k^{-1/2}U$ for a $k \times k$ unitary U. (Hence, $\|V\| \leq k^{-1/2}$.) If we compute the input-independent trigonometric factors in the FFT, often called the twiddle factors, accurately up to additive error 2^{-R}, then $k^{-1/2}U$ is accurate up to additive error $\delta_{FFT} \leq k^{1/2}2^{-R}$ in operator norm. (This bound is loose compared to that in the previous paragraph, since in this paragraph we do not consider the fast Fourier transform that is numerically more stable.) Since $k \leq 2N+1$, the conversion from ℓ_∞-norm to 2-norm incurs a factor of at most $\sqrt{2N+1}$. Hence, the additive error δ_{2n} in $C_{2j}^{(2n)}$ for any j is at most $\|\tilde{V}\tilde{v} - Vv\|_{\max} \leq \|\tilde{V}\tilde{v} - Vv\|_2 \leq \|\tilde{V} - V\| \cdot \|\tilde{v}\|_2 + \|V\| \cdot \|\tilde{v} - v\|_2 \leq \delta_{FFT}\sqrt{2N+1} + \delta$. That is,

$$\delta_{2n} \leq 400N^3\epsilon^{-1}2^{-R}. \tag{25}$$

Step 5. This is an implementation of Theorem 2. Thanks to the condition (iv) of Step 1, we know

$$\left\| C_{\pm 2n}^{(2n)} \right\| \geq \epsilon/N. \tag{26}$$

Put $F(t^2) = E_0E_1(t)\cdots E_{2n}(t)$. Then, for any $m = 1, 2, \ldots, 2n$, we observe that $C_{\pm m}^{(m)}$ is the product

$$C_m^{(m)} = E_0P_1P_2\cdots P_m, \qquad C_{-m}^{(m)} = E_0Q_1Q_2\cdots Q_m. \tag{27}$$

In particular, by the submultiplicative rule of operator norm we have

$$\left\|C^{(m)}_{\pm m}\right\| \geq \left\|C^{(2n)}_{\pm 2n}\right\| \geq \epsilon/N; \tag{28}$$

that is, the leading coefficients never become smaller in norm through the loop over m.

Let δ_m be the maximum additive error of $C^{(m)}_k$ for any k. We assume that $\delta_m \leq \epsilon/(2N)$. For brevity, let $C = C^{(m)}_{\pm m}$, and let $\tilde{C}$ be the approximate C due to numerical error. Then, $\delta_m \leq \epsilon/(2N) \leq \|C\|/2$ and $\|C\|/2 \leq \|C\| - \delta_m \leq \left\|\tilde{C}\right\| \leq \|C\| + \delta_m \leq 2\|C\|$. Now,

$$\begin{aligned}\left\|\tilde{C}^\dagger\tilde{C} - C^\dagger C\right\| &\leq \left\|\tilde{C}^\dagger\right\|\left\|\tilde{C} - C\right\| + \left\|\tilde{C}^\dagger - C^\dagger\right\|\|C\| \\ &\leq 2\|C\|\cdot\delta_m + \delta_m\cdot\|C\| \\ &= 3\|C\|\delta_m\end{aligned} \tag{29}$$

$$\left|\operatorname{Tr}(\tilde{C}^\dagger\tilde{C}) - \operatorname{Tr}(C^\dagger C)\right| \leq \sum_{j=0}^{1}\left|\langle j|\,\tilde{C}^\dagger\tilde{C} - C^\dagger C\,|j\rangle\right| \leq 6\|C\|\delta_m \tag{30}$$

$$\operatorname{Tr}(C^\dagger C) \geq \|C\|^2 \tag{31}$$

$$\operatorname{Tr}(\tilde{C}^\dagger\tilde{C}) \geq \left\|\tilde{C}\right\|^2 \geq \frac{1}{4}\|C\|^2 \tag{32}$$

Hence, we see that the additive error β_m in $P^{(m)}$ (or $Q^{(m)}$) is

$$\begin{aligned}\left\|\tilde{P}^{(m)} - P^{(m)}\right\| &\leq \frac{\left\|\tilde{C}^\dagger\tilde{C} - C^\dagger C\right\|}{\operatorname{Tr}(\tilde{C}^\dagger\tilde{C})} + \frac{\left\|C^\dagger C\right\|\left|\operatorname{Tr}(C^\dagger C) - \operatorname{Tr}(\tilde{C}^\dagger\tilde{C})\right|}{\operatorname{Tr}(\tilde{C}^\dagger\tilde{C})\operatorname{Tr}(C^\dagger C)} \\ &\leq \frac{4}{\|C\|^2}\cdot 3\|C\|\delta_m + \frac{4}{\|C\|^2}\cdot\frac{1}{\|C\|^2}\cdot\|C\|^2\cdot 6\|C\|\delta_m \\ &\leq 36N\epsilon^{-1}\delta_m.\end{aligned} \tag{33}$$

In turn, assuming $\beta_m \leq 1$ we have

$$\begin{aligned}\delta_{m-1} &= \max_k \left\|\tilde{C}^{(m)}_{k-1}\tilde{Q}_m + \tilde{C}^{(m)}_{k+1}\tilde{P}_m - C^{(m)}_{k-1}Q_m - C^{(m)}_{k+1}P_m\right\| \\ &\leq \max_k \left\|\tilde{C}^{(m)}_{k-1}\tilde{Q}_m - C^{(m)}_{k-1}\tilde{Q}_m\right\| + \left\|C^{(m)}_{k-1}\tilde{Q}_m - C^{(m)}_{k-1}Q_m\right\| + (\text{similar terms}) \\ &\leq 2(2\delta_m + \beta_m) \\ &\leq 76N\epsilon^{-1}\delta_m.\end{aligned} \tag{34}$$

Combining Eqs. (25) and (34) we conclude that

$$\beta_0 := \delta_0 \leq 400N^3\epsilon^{-1}(76N\epsilon^{-1})^{2N}2^{-R} =: \Gamma. \tag{35}$$

Step 6. Now we have approximate E_0 up to additive error β_0 and P_m up to β_m ($m = 1,\ldots,2n$). The evaluation of $E_m(t)$ is then accurate up to $2\beta_m$ for $t \in U(1)$, and that of $a(t^2) + ib(t^2) = \langle +|\,E_0E_1(t)\cdots E_{2n}(t)\,|+\rangle$ is accurate up to (suppressing t for brevity)

$$\begin{aligned}\left\|\tilde{E}_0\cdots\tilde{E}_{2n} - E_0\cdots E_{2n}\right\| &\leq \sum_{m=0}^{2n}\left\|E_0\cdots E_{m-1}\tilde{E}_m\cdots\tilde{E}_{2n} - E_0\cdots E_m\tilde{E}_{m+1}\cdots\tilde{E}_{2n}\right\| \\ &\leq \sum_{m=0}^{2n}\left(2\beta_m\prod_{\ell=m+1}^{2n}(1+2\beta_\ell)\right).\end{aligned} \tag{36}$$

We want this to be smaller than ϵ. A sufficient condition is thus

$$(2N+1)2\Gamma\,(1+2\Gamma)^{2N+1} \le \epsilon \tag{37}$$

$$\text{or} \qquad R \gtrsim 2N\log_2(N/\epsilon). \tag{38}$$

All the (ad hoc) assumptions in the course of error estimation above, are satisfied by this choice of R.

The correctness of the algorithm is clear from the construction and error analysis above. The final error guarantee is from the condition (ii) of Step 1 and Eq. (37).

4.3 Computational complexity

All arithmetic in the algorithm above operates with at most R-bit numbers, where R is chosen to be $\Theta(N\log(N/\epsilon))$. The number of elementary bit operations (AND, OR, NOT) to perform one basic arithmetic operation $(+,-,\times,/)$ on u-bit numbers is upper bounded by $\mathcal{O}(u \text{ polylog}(u))$ [17].[9] Let us count the number of arithmetic operations.

Step 1: Assuming that the coefficients of the true function $A(z)$ and $B(z)$ are given to accuracy $\mathcal{O}(\epsilon/N)$, it takes $\mathcal{O}(N\log(N/\epsilon))$ arithmetic operations to find rational approximations. There could be $\mathcal{O}(\text{polylog}(N/\epsilon))$ additional cost in full complexity in simplifying rational numbers by Euclid's algorithm.

Step 2: The root finding takes time $\mathcal{O}(N^3 + N^2 R) \le \mathcal{O}(N^3\log(N/\epsilon))$ [15]; this count includes all the cost of bit operations for high precision arithmetic.

Step 3: Selecting roots inside the unit disk requires $\mathcal{O}(N)$ absolute value evaluations and $\mathcal{O}(N)$ comparisons. The polynomial function evaluation involves $\mathcal{O}(N)$ arithmetic operations. We need $\mathcal{O}(N)$ values, resulting in $\mathcal{O}(N^2)$ arithmetic operations.

Step 4: The FFT requires $\mathcal{O}(N\log N)$ arithmetic operations given $\mathcal{O}(\log N)$ trigonometric function values, which can be computed by Taylor expansions of order R, invoking $\mathcal{O}(R\log N)$ arithmetic operations. These result in arithmetic complexity $\mathcal{O}(N\log(N)\log(N/\epsilon))$ for the FFT.

Step 5: Updating the Fourier coefficient list $C_k^{(m)}$ involves $\mathcal{O}(N)$ arithmetic operations, which we do for $\mathcal{O}(N)$ times, resulting in $\mathcal{O}(N^2)$ arithmetic operations.

Overall, the computational complexity is $\mathcal{O}(N^3 \text{ polylog}(N/\epsilon))$, under the random-access memory model of computation.

Practically, it may be useful to run the algorithm with arithmetic precision with, say, $R_0 = 64$ bits initially, test the final decomposition on $\mathcal{O}(N)$-th roots of unity (which takes only $\mathcal{O}(N^2)$ operations with $\mathcal{O}(\log(N/\epsilon))$-bit arithmetic), and repeat the whole process under an exponential scheduling on the number $R_{r+1} = 2R_r$ of bits of precision in the arithmetic, until the test reveals that the answer is acceptable. In this way the arithmetic uses no more than twice the number of bits of precision that is actually needed to handle the numerical instability of our algorithm for a given input, without knowing a tailored number beforehand. For the upper bound $R = \mathcal{O}(N\log(N/\epsilon))$ in the worst case, there will be at most $r_{\max} = \mathcal{O}(\log N + \log\log(N/\epsilon))$ rounds, but the overall time complexity is a constant multiple of the last round's due to the exponential scheduling.

[9] This reference is concerned with multiplications, but the division has essentially the same cost using the Newton's method [18, 4.3.3 R].

5 Application to Hamiltonian simulation

In this section, we review and redo existing analysis [1, 19], complemented with a minor modification to our algorithm exploiting a certain structure of the eigenvalue transformation function.

Suppose we are given with a unitary W whose eigenvalues $\mp e^{\pm i\theta_\lambda}$ are associated with those λ of a Hermitian matrix $H = \sum \lambda |\lambda\rangle\langle\lambda|$ with $\|H\| \le 1$ as

$$\sin\theta_\lambda = \lambda. \tag{39}$$

The correspondence between W and H might seem contrived at this stage, but when H is represented as a linear combination of unitaries [5, 6], it is possible to construct such W as a quantum circuit [3]. The relation of Eq. (39) is in fact common whenever quantum walk is used [20, 21]. (See Appendix A for some detail.) So, the desired transformation is

$$f : \mp e^{\pm i\theta_\lambda} \mapsto e^{-i\tau\sin\theta_\lambda} \approx \langle +| F(e^{i\theta_\lambda/2}) |+\rangle \tag{40}$$

where $F(t)$ should be constructed by quantum signal processing [1]. This will implement $e^{-i\tau H}$. Since the product of n primitive matrices yields a Fourier component of frequency at most $n/2$, $F(t)$ must consist of at least 2τ factors. (The factor of 2 is due to the half-angle in the argument of F.) Note that the success probability of the post-selection is close to 1 since $|f(e^{i\theta_\lambda})| = 1$.

With $e^{i\varphi} = z$, we write

$$\exp(i\tau\sin\varphi) = \exp\left(\tau\frac{z - z^{-1}}{2}\right) = \sum_{k\in\mathbb{Z}} J_k(\tau) z^k \tag{41}$$

where J_k are the Bessel functions of the first kind; one can take Eq. (41) as a *definition* of the Bessel functions. This is the Fourier series of $\varphi \mapsto \exp(i\tau\sin\varphi)$. The substitution $\tau \to -\tau$ and $z \to -z$ together with the uniqueness of the Fourier series implies $J_k(-\tau) = (-1)^k J_k(\tau)$. Similarly, the substitution $z \to -1/z$ implies $J_{-k}(\tau) = (-1)^k J_k(\tau)$. We separate the reciprocal and anti-reciprocal parts of the expansion as

$$\exp\left(\tau\frac{z - z^{-1}}{2}\right) = \underbrace{\sum_{k\in 2\mathbb{Z}} J_k(\tau)\frac{z^k + z^{-k}}{2}}_{A(z)} + i \underbrace{\sum_{k\in 2\mathbb{Z}+1} J_k(\tau)\frac{z^k - z^{-k}}{2i}}_{B(z)}. \tag{42}$$

This expansion, called the Jacobi-Anger expansion, converges absolutely at a superexponential rate. We can use the steepest descent method [22] which is generally applicable. Expressing the Fourier transform as a contour integral we see

$$J_k(\tau) = \frac{1}{2\pi i}\int_C \frac{\mathrm{d}z}{z^{k+1}} e^{(\tau/2)(z - z^{-1})}$$

where C is the unit circle. Since the integrand is analytic except for $z = 0$, we may deform C. For $2k > \tau > 0$, $z \approx 2k/\tau > 1$ is a saddle point for the absolute value of the integrand. We take C to be a circle through this point, to have that

$$\begin{aligned} |J_k(\tau)| &\le \left(\frac{\tau}{2k}\right)^k \int_0^{2\pi} \frac{\mathrm{d}\theta}{2\pi} \exp[(k - (\tau^2/4k))\cos\theta] \le \left(\frac{e\tau}{2k}\right)^k \qquad \text{for } 2k > \tau > 0, \\ |J_k(\tau)| &\le \left(\frac{e|\tau|}{2|k|}\right)^{|k|} \qquad \text{for } k \in \mathbb{Z}\setminus\{0\}, \tau \in \mathbb{R}. \end{aligned} \tag{43}$$

It is important that the convergence of the series depends on the size of the region on which the function is analytic — this is a general fact [22]. For the Jacobi-Anger expansion the function is almost entire, and the convergence is superexponential. Note that [23, 9.1.62] asserts $|J_k(\tau)| \le |\tau/2|^{|k|}/|k|!$ for any $\tau \in \mathbb{R}$ and $k \in \mathbb{Z}$ which is tighter than Eq. (43).

Now, a partial sum of the Jacobi-Anger expansion can be written as

$$\sum_{k:|k|\le N} J_k(\tau) z^k = \underbrace{J_0(\tau) + \sum_{\text{even } k:2\le k\le N} J_k(\tau)(z^k + z^{-k})}_{\tilde{A}(z)} + i \underbrace{\sum_{\text{odd } k:1\le k\le N} J_k(\tau)\frac{z^k - z^{-k}}{i}}_{\tilde{B}(z)} . \tag{44}$$

This is ϵ-close to the full expansion if $N = \Omega(|\tau| + \log(1/\epsilon))$ by Eq. (43) for any $z \in U(1)$.[10] Numerical experiments suggest that the bound is quite tight and it suffices to choose

$$N \approx \frac{e}{2}|\tau| + \ln(1/\epsilon) \approx 1.36|\tau| + 2.30\log_{10}(1/\epsilon). \tag{45}$$

The Laurent polynomials $\tilde{A}(z)$ and $\tilde{B}(z)$ are pure real-on-circle. Applying our algorithm, we obtain real-on-circle Laurent polynomials $a(z) = a_+(z), b(z) = ib_-(z), c(z) = ic_-(z), d(z) = d_+(z)$. The pure Laurent polynomials $c(z), d(z)$ are calculated by Lemma 4 where we choose $c(z)$ to be anti-reciprocal and $d(z)$ reciprocal. (This choice is to have our results in the same convention as those of Ref. [2].) Note that every exponent of z of the polynomial $1 - a(z)^2 - b(z)^2$ here, whose roots must be computed, is even since $a(z)$ has only even exponents and $b(z)$ has only odd exponents, so it is always better to feed a Laurent polynomial g of degree n, instead of $2n$, where

$$g(z^2) = 1 - a(z)^2 - b(z)^2 \tag{46}$$

into the root finding routine. Given the expanded form of $1 - a(z)^2 - b(z)^2$, it takes no effort to find g. In this case, the intermediate polynomial $e(z)$ in Eq. (13) of Lemma 4 is

$$e(z) = \prod_{c\in\mathbb{C}\ :\ g(c)=0,\ |c|<1} \left(z - \frac{c}{z}\right). \tag{47}$$

We have implemented our algorithm with constants chosen as above using Wolfram Mathematica 11, and measured the running time as a function of τ for two fixed values of ϵ. The result is shown in Fig. 1. The running time scales asymptotically as the cubic of τ as expected. We used internal routines of Mathematica for the rational approximation, high precision arithmetic, root finding, and Fourier transform. The computing was by Microsoft Surface Book with Intel Core i7-6600U at 2.6 GHz and 16 GB of RAM.

6 Application to Matrix inversion

While there are slightly more efficient implementations of matrix inversion problems [9] using quantum signal processing [4], here we contend ourselves with an eigenvalue transformation

[10] The proof of this is along the same lines as proving the convergence of Taylor series for e.g. the exponential function, and is left to the reader.

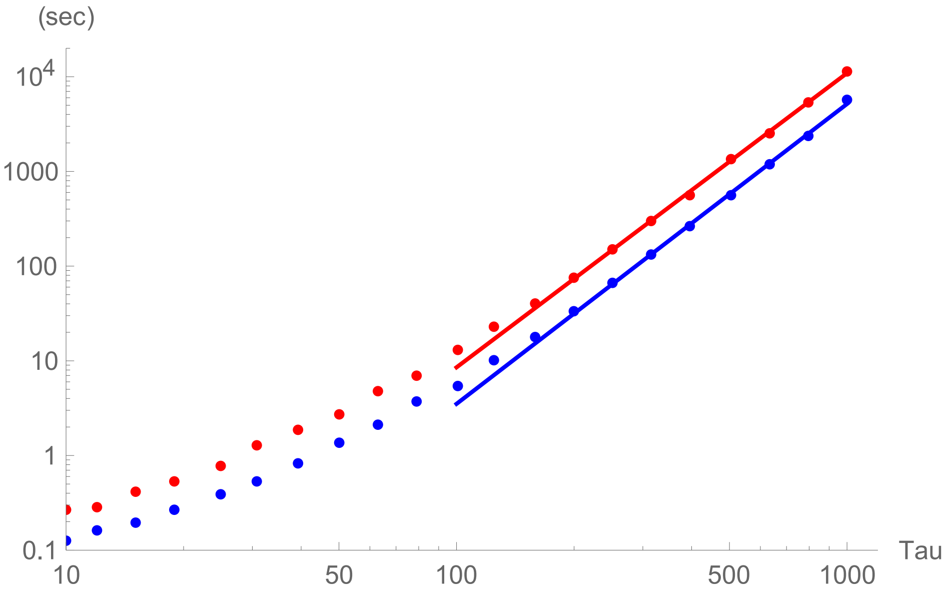

Figure 1: Running time of our algorithm for the Jacobi-Anger expansion as a function of τ, implemented in Wolfram Mathematica 11. The upper red data set has $\epsilon = 10^{-9}$ and the lower blue $\epsilon = 10^{-4}$. The number of decimal digits in the intermediate steps is $(n/3)\ln(n/\epsilon)$ where n is the degree of the input Laurent polynomial and the factor of 3 is empirically chosen to make the numerical error negligible. The straight lines represent functions $\mathrm{const}\cdot\tau^\gamma$ of exponents $\gamma = 3.11$ for $\epsilon = 10^{-9}$ (red) and $\gamma = 3.17$ for $\epsilon = 10^{-4}$ (blue). The top right data point has 2172 primitive matrices in the decomposition.

perspective. The techniques of Refs. [4] reduces the number of ancilla qubits by one or two, and hence relieves some burden of implementing controlled unitary, but the underlying mathematics, regarding polynomial approximations and finding interspersing single-qubit unitaries, is unchanged.

Suppose a hermitian matrix H of norm 1 that we wish to invert is block-encoded in a unitary W so that W has eigenvalues $\mp e^{\pm i\theta_\lambda}$ associated with an eigenvalue λ of H. This encoding is the same as in the Hamiltonian simulation above. The condition for H being hermitian is not too restrictive since, for any matrix M, an enlarged matrix $|0\rangle\langle 1| \otimes M + |1\rangle\langle 0| \otimes M^\dagger$ is always hermitian. Then, we want eigenvalue transformation $\mp e^{\pm i\theta_\lambda} \mapsto 1/\sin\theta_\lambda$. As we should not invert a singular matrix, we assume that eigenvalues of H are bounded away from zero by $1/\kappa$ where $\kappa \ge 1$ is the condition number of H. (Stricly zero eigenvalues are fine if we are interested in a pseudo-inverse.) Thanks to the condition number assumption, we need to find a polynomial approximation to the function $\mp e^{\pm i\theta_\lambda} \mapsto 1/\sin\theta_\lambda$ that is good for values $\sin\theta_\lambda$ away from zero by $1/\kappa$. For this purpose, there is a useful polynomial [24]:

Lemma 6. *Let $\epsilon > 0$, $\kappa \ge 1$ and $z \in U(1)$. Suppose integers $b \ge b' \ge 1$ satisfy $b \ge \kappa^2 \ln(2/\epsilon)$ and $b' \ge \sqrt{b\ln(8/\epsilon)}$. For $\sin\varphi = (z - z^{-1})/(2i) \in \mathbb{R}$ with $|\sin\varphi| \ge 1/\kappa > 0$, we have*

$$\left| \frac{2i}{\kappa(z - z^{-1})} - \underbrace{\frac{2i}{2^{2b}\kappa(z - z^{-1})} \sum_{k=-b'}^{b'} \binom{2b}{b+k} (1 - z^{2k})}_{f(z)} \right| \le \epsilon. \tag{48}$$

Moreover, for all $z \in U(1)$ we have $|f(z)| \le 2b'/\kappa$.

The function $f(z)$ is a genuine Laurent polynomial[11] since the sum vanishes at $z = \pm 1$.

Proof. First, let $b \ge b' \ge 1$ be any integers, and let $z = e^{i\varphi} \in U(1)$ be any complex number. Then,

$$\left| \underbrace{1 - \left(\frac{z+z^{-1}}{2}\right)^{2b}}_{g(z)} - \underbrace{\frac{1}{2^{2b}} \sum_{k=-b'}^{b'} \binom{2b}{b+k} (1 - z^{2k})}_{h(z)} \right| \le 4e^{-b'^2/b} \tag{49}$$

because $\left|2^{-2b} \sum_{k\,:\,|k|>b'} \binom{2b}{b+k}(1-z^{2k})\right| \le 2^{-2b+1} \sum_{k\,:\,|k|>b'} \binom{2b}{b+k}$ and Hoeffding's inequality on the tail of binomial probability distributions implies Eq. (49). If $\sin\varphi = \frac{z-z^{-1}}{2i}$ with $|\sin\varphi| \ge 1/\kappa > 0$, then

$$|1 - g(z)| = \left| \left(\frac{z+z^{-1}}{2}\right)^{2b} \right| \le e^{-b/\kappa^2}, \tag{50}$$

since $(1 - \sin^2\varphi)^b \le e^{-b/\kappa^2}$ whenever $|\sin\varphi| \ge 1/\kappa$.

Thus, for large b we see that $1 - ((z+z^{-1})/2)^{2b}$ vanishes when $\sin\varphi = 0$ (i.e., $z = \pm 1$), but is close to 1 for $|\sin\varphi| \ge 1/\kappa$. By Eq. (49) this function can be replaced with a lower degree polynomial function. Indeed, for $|\sin\varphi| \ge 1/\kappa$ we have

$$\left| \frac{2i}{\kappa(z - z^{-1})} - \frac{2ih(z)}{\kappa(z-z^{-1})} \right| \le |1 - h(z)| \le e^{-b/\kappa^2} + 4e^{-b'^2/b}, \tag{51}$$

which implies Eq. (48).

For the last claim, we observe a chain of (in)equalities: For any integer $k \ge 1$ we have

$$\begin{aligned} (z - z^{-1})(z^{k-1} + z^{k-3} + \cdots + z^{-k+3} + z^{-k+1}) &= z^k - z^{-k}, \\ |(z^k - z^{-k})/(z - z^{-1})| &\le k && \text{for } z \in U(1), \\ |\sin^2 k\varphi / \sin\varphi| \le |\sin k\varphi/\sin\varphi| &\le k && \text{for } \varphi \in \mathbb{R}, \\ |(2 - z^{2k} - z^{-2k})/(z - z^{-1})| &\le 2k && \text{for } z \in U(1). \end{aligned}$$

The function $f(z)$ is the "average" over k of

$$\frac{(2 - z^{2k} - z^{-2k})i}{(z - z^{-1})\kappa}$$

with respect to a subnormalized probability distribution. Therefore the claim follows. □

[11] The polynomial of Eq. (48) is the same as that in [24, Lemma 17-19]. The bound there is similar to ours, but the polynomial degree is worse than ours by a factor of $\log\kappa$. This difference is due to a different normalization — we approximate $1/(\kappa x)$ rather than $1/x$. Our analysis might look simpler, but it is not due to a "new" approach; the "difference" is only in the usage of exponential functions rather than trigonometric functions.

Choosing $b' = \lceil \kappa \ln(8/\epsilon) \rceil$, and feeding a real-on-circle anti-reciprocal Laurent polynomial $f(z)/\ln(8/\epsilon)$, which is at most 1 in magnitude by the last claim of Lemma 6, into our algorithm, we obtain a desired eigenvalue inversion quantum algorithm. The success probability can be as small as $\Omega(1/(\kappa \log(1/\epsilon))^2)$, and hence we had better amplify the amplitude for post-selection, enlarging the quantum gate complexity by a factor of $\mathcal{O}(\kappa \log(1/\epsilon))$. Overall, the quantum gate complexity is proportional to the product of the degree of the Laurent polynomial above and the number of iterations for the amplitude amplification.

7 Discussion

We have determined the scope of $SU(2)$-valued periodic polynomial functions and their decomposition (Theorem 2), and analyzed the algorithmic aspects. Our algorithm for the decomposition is not numerically stable in a usual sense — a numerically stable algorithm should only require $\mathrm{polylog}(N/\epsilon)$ bits of precision, rather than $\mathrm{poly}(N \log(1/\epsilon))$. The instability appears to be unavoidable in any method that reduces polynomial degree iteratively by one at a time (such as our Step 5), at least in the early stage of the polynomial degree reduction. The numerical error arises due to the small norm of leading coefficients in our algorithm, and the small leading coefficients of an input polynomial are necessary if a nonpolynomial function admits converging polynomial approximations. However, it might be the case that the leading coefficient matrix $C_m^{(m)}$ becomes large in norm rather quickly in Step 5 of our algorithm, in which case our analysis could be loose. For a schematic example, consider an identity $t^{2n}P + t^{-2n}(I-P) = \left(tP + t^{-1}(I-P)\right)^{2n}$ for any projector P. The numerical instability to decompose the left-hand side into the right-hand side is far less severe than that in the worst case of Step 5, and similar situations might occur during the execution of Step 5 for some class of input functions. This deserves further investigation.

Acknowledgments

I thank Guang Hao Low for valuable discussions, and Robin Kothari for useful comments on the manuscript.

A Jordan's Lemma and block encoding of Hamiltonians

The following is a well-known fact, but we include it here for completeness.

Lemma 7. *Let P and Q be arbitrary self-adjoint projectors on a finite dimensional complex vector space V. Then, V decomposes into orthogonal subspaces V_j invariant under P and Q, where each V_j has dimension 1 or 2.*

For any hermitian operator H, let $\mathrm{supp}(H)$ denote the subspace support of H, i.e., the orthogonal complement of the kernel of H. Clearly, $\mathrm{supp}(H)$ is an invariant subspace of H, and H is invertible within $\mathrm{supp}(H)$.

Proof. We will find a subspace W of dimension at most 2 that is invariant under both P and Q. This is sufficient since the orthogonal complement of W is also invariant and the proof will be completed by induction in the dimension of V.

Put $P' = I - P$ and $Q' = I - Q$, and consider the identities

$$PQP + PQ'P + P'QP' + P'Q'P' = I, \tag{52}$$

$$\operatorname{supp}(PQP) + \operatorname{supp}(PQ'P) + \operatorname{supp}(P'QP') + \operatorname{supp}(P'Q'P') = V, \tag{53}$$

where the second equality is because the intersection of the orthogonal complements of the four supports is zero. Therefore, at least one of the four supports is nonzero, and without loss of generality assume $S = \operatorname{supp}(PQP) \neq 0$. Let $|\psi\rangle \in S$ be an eigenvector of PQP; $PQP\,|\psi\rangle = a\,|\psi\rangle$. The associated eigenvalue a is nonzero by definition of S. Now consider $W = \operatorname{span}\{|\psi\rangle, Q\,|\psi\rangle\}$. Observe that $a\,|\psi\rangle = PQP\,|\psi\rangle = PPQP\,|\psi\rangle = aP\,|\psi\rangle$, and hence $P\,|\psi\rangle = |\psi\rangle$. Moreover, $PQ\,|\psi\rangle = PQP\,|\psi\rangle = a\,|\psi\rangle$. Therefore, W is a nonzero invariant subspace under both P and Q. □

This Jordan's lemma can be applied to two hermitian unitaries (reflections) U_1, U_2 as any hermitian unitary is $2P - I$ for some projector P. It immediately follows that there is a basis where U_1 is diagonal and U_1U_2 is block-diagonal with at most two-dimensional blocks and each block belongs to $U(1)$ or $U(2)$. In any such irreducible two-dimensional block, both unitaries cannot be scalar multiplications because of the irreducibility and hence we have

$$U_1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad U_2 = \begin{pmatrix} \lambda & e^{i\phi}\sqrt{1-\lambda^2} \\ e^{-i\phi}\sqrt{1-\lambda^2} & -\lambda \end{pmatrix} \tag{54}$$

for some real number $\lambda \in [-1, 1]$ and an angle $\phi \in \mathbb{R}$, up to a permutation of rows and columns. Therefore, the product $W = -iU_1U_2$ is a rotation in a two-dimensional subspace that appears in Grover search algorithm [25], and has eigenvalues $\pm e^{\mp i\theta}$ where $\sin\theta = \lambda$. This is relevant in a Hamiltonian simulation problem where the Hamiltonian is "block-encoded" as $H = (\langle G| \otimes I)\, U_2\, (|G\rangle \otimes I)$, which is the case if H is represented as, e.g., a linear combination of Pauli operators. Here $|G\rangle$ is a state on a *proper* tensor factor of the Hilbert space on which U_2 act, and $U_1 = (2\,|G\rangle\langle G| - I) \otimes I$.

References

[1] G. H. Low and I. L. Chuang, "Optimal Hamiltonian simulation by quantum signal processing," Phys. Rev. Lett. **118**, 010501 (2017), arXiv:1606.02685 .

[2] G. H. Low, T. J. Yoder, and I. L. Chuang, "Methodology of resonant equiangular composite quantum gates," Phys. Rev. X **6**, 041067 (2016), arXiv:1603.03996 .

[3] G. H. Low and I. L. Chuang, "Hamiltonian simulation by Qubitization," Quantum **3**, 163 (2019), arXiv:1610.06546 .

[4] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics," in *STOC 2019 Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (2019) pp. 193–204, arXiv:1806.01838 .

[5] A. M. Childs and N. Wiebe, "Hamiltonian simulation using linear combinations of unitary operations," Quantum Information and Computation **12**, 901–924 (2012), arXiv:1202.5822 .

[6] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, "Exponential improvement in precision for simulating sparse Hamiltonians," in *Proceedings of the 46th ACM Symposium on Theory of Computing (STOC)* (2014) pp. 283–292, arXiv:1312.1414 .

[7] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, "Toward the first quantum simulation with quantum speedup," Proceedings of the National Academy of Sciences **115**, 9456–9461 (2018), arXiv:1711.10980 .

[8] J. Haah, M. Hastings, R. Kothari, and G. H. Low, "Quantum algorithm for simulating real time evolution of lattice hamiltonians," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)* (2018) pp. 350–360.

[9] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for solving linear systems of equations," Phys. Rev. Lett. **15**, 150502 (2009), arXiv:0811.3171 .

[10] A. Shamir, "Factoring numbers in $O(\log n)$ arithmetic steps," Information Processing Letters **8**, 28–31 (1979).

[11] A. Schönhage, "On the power of random access machines," in *Automata, Languages and Programming. ICALP 1979. Lecture Notes in Computer Science,*, Vol. 71, edited by M. H.A. (Springer, Berlin, Heidelberg, 1979) pp. 520–529.

[12] J. Qian and C. A. Wang, "How much precision is needed to compare two sums of square roots of integers?" Information Processing Letters **100**, 194 – 198 (2006).

[13] Q. Cheng, X. Meng, C. Sun, and J. Chen, "Bounding the sum of square roots via lattice reduction," Math. Comp. **79**, 1109–1122 (2010), arXiv:0905.4487 .

[14] G. H. Low and I. L. Chuang, "Hamiltonian simulation by uniform spectral amplification," arXiv:1707.05391 .

[15] V. Y. Pan, "Optimal and nearly optimal algorithms for approximating polynomial zeros," Computers & Mathematics with Applications **31**, 97 – 138 (1996).

[16] G. U. Ramos, "Roundoff error analysis of the fast fourier transform," Mathematics of Computation **25**, 757–768 (1971).

[17] D. Harvey and J. van der Hoeven, "Faster integer multiplication using short lattice vectors," Open Book Series **2**, 293–310 (2019), arXiv:1802.07932 .

[18] D. E. Knuth, *The Art of Computer Programming*, 3rd ed., Vol. 2 (Addison-Wesley, 1998).

[19] D. W. Berry, A. M. Childs, and R. Kothari, "Hamiltonian simulation with nearly optimal dependence on all parameters," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (2015) pp. 792–809, arXiv:1501.01715 .

[20] A. M. Childs, "On the relationship between continuous- and discrete-time quantum walk," Commun. Math. Phys. **294**, 581–603 (2010), arXiv:0810.0312 .

[21] D. W. Berry and A. M. Childs, "Black-box hamiltonian simulation and unitary implementation," Quantum Information and Computation **12** (2012), arXiv:0910.4157 .

[22] J. P. Boyd, "The rate of convergence of fourier coefficients for entire functions of infinite order with application to the weideman-cloot sinh-mapping for pseudospectral computations on an infinite interval," Journal of Computational Physics **110**, 360 – 372 (1994).

[23] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions* (National Bureau of Standards, 1964).

[24] A. M. Childs, R. Kothari, and R. D. Somma, "Quantum algorithm for systems of linear equations with exponentially improved dependence on precision," SIAM Journal on Computing **46**, 1920–1950 (2017), arXiv:1511.02306 .

[25] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings, 28th Annual ACM Symposium on the Theory of Computing (STOC)* (1996) pp. 212–219, arXiv:quant-ph/9605043 .