

A case study against QSVT: assessment of quantum phase estimation improved by signal processing techniques

Sean Greenaway,^{1,*} William Pol,¹ and Sukin Sim^{1,†}

¹*PsiQuantum, 700 Hansen Way, Palo Alto, California 94304, USA*

In recent years, quantum algorithms have been proposed which use quantum phase estimation (QPE) coherently as a subroutine without measurement. In order to do this effectively, the routine must be able to distinguish eigenstates with success probability close to unity. In this paper, we provide the first systematic comparison between two approaches towards maximizing this success probability, one using the quantum singular value transform and the other leveraging window functions, which have been previously studied as priors of the phase value distribution. We find that the quantum singular value transform is significantly outclassed by the window function approach, with the latter able to achieve between 3 and 5 orders of magnitude improvement in the success probability with approximately 1/4 the query cost. Our circuit simulation results indicate that QPE is not a domain which benefits from the integration of QSVT and we show that the use of the Kaiser window function is currently the most practical choice for realizing QPE with high success probability.

I. INTRODUCTION

Quantum computation holds the promise to accelerate calculations in a wide variety of domains for particular tasks, including search and optimization [1], simulation of chemistry [2, 3], and cryptographic applications such as factoring prime numbers [4, 5]. A ubiquitous workhorse algorithm used in all of these domains is the quantum phase estimation (QPE) algorithm [6, 7], an algorithm which allows one to sample from the eigenspectrum of an input operator. Two common tasks which have exhibited computational speedups that make use of QPE include factoring prime numbers [5, 8, 9] and solving for the ground state energy of quantum mechanical systems in chemistry and condensed matter [10–15].

For an operator \hat{O} and one of its eigenstates $|\psi\rangle$ obeying the relation $\hat{O}|\psi\rangle = e^{2\pi i \theta}|\psi\rangle$, QPE aims to output an estimate of θ , the eigenphase corresponding to $|\psi\rangle$.¹ Two salient criteria which characterize the performance and cost of QPE are the accuracy ϵ to which we estimate θ , and the probability of success of returning an ϵ -close estimate. Two factors influence this probability of success: the overlap of the initial state input to QPE with the true eigenstate $|\psi\rangle$, and more subtly, a phenomenon referred to as *bit discretization error*, which arises from using a finite number m bits to encode eigenphases and eigenenergies (which may have no exact finite-bit representation). These two factors contribute to the overall probability of success of returning an ϵ -close estimate in different ways. The overlap of the initial state and $|\psi\rangle$ determines the likelihood of projecting onto $|\psi\rangle$ and returning a corresponding m -bit phase estimate. Assuming one successfully projects onto the eigenstate $|\psi\rangle$, bit discretization error determines the likelihood that a particular returned m -bit phase readout is ϵ -close to the true target eigenphase θ . Addressing the influence on probability of success due to bit discretization error is an ongoing line of research in the literature and the focus of this study.

Two methods have emerged in the literature that aim to address bit discretization error: the quantum singular value transform (QSVT) framework applied to QPE [16–18], which we will refer to as “QSVT QPE,” and the use of window or taper functions [13, 19, 20]. The QSVT framework is a powerful conceptual tool, as it recasts many tasks in quantum computation as attempting to implement a function on the eigenvalues or singular values of an operator (using the language of signal processing). Window functions are tools borrowed from classical signal processing that are used to reduce errors in the frequency spectra of signals. Here we compare and contrast the relative success probabilities and resource costs of these two methods.

The favorability of one method over the other depends on how QPE is used in a full computation, and the requirement on the success probability. It is useful to distinguish between *coherent* and *incoherent* uses of QPE. In an incoherent use of QPE, we measure the phase qubits register immediately after performing a single iteration of QPE. In contrast, when we use QPE coherently, we do not measure the phase qubits register in order to avoid collapsing

* Corresponding author: sgreenaway@psiquantum.com

† Corresponding author: ssim@psiquantum.com

¹ In our case, the operators of interest are typically Hamiltonians, so we will use the terminology “eigenenergy” and “eigenvalue” interchangeably.

the resulting superposition state. For some applications, maintaining coherence is necessary [21–27]. One such application requiring high probability and coherent use of QPE is estimating the expectation value of an observable \hat{F} with respect to a state $|\psi\rangle$ that is *not* its eigenstate [28]. This application requires repeated application of a unitary that performs a reflection about the state $|\psi\rangle$. Constructing such a reflection is in general non-trivial; instead, QPE is used to construct a reflection about $|\psi\rangle$'s corresponding eigenphase as a proxy. Errors due to inexact, approximate implementations of reflections play a significant role in the overall success probability of algorithms that make repeated queries to reflections. With repeated queries, error will accrue during the course of the computation, as in amplitude amplification or amplitude estimation. For the reflection implemented in [28], this QPE-based reflection only approximately performs (a proxy for) a reflection about $|\psi\rangle$, precisely because the exact eigenphase is not representable with a finite number of bits. Mitigating bit discretization error and its influence on the success probability of QPE is crucial to this kind of application.

Many of the conclusions we draw on the relative advantage in performance between the use of window functions in QPE and QSVD QPE in the rest of this manuscript are particular to regimes in which the required success probability is close to 1. For the rest of this manuscript, we explore the tradeoffs of window functions and QSVD QPE assuming such a regime. One may consider the coherent use of QPE for calculating observables of operators with respect to bases that do not form part of the operator's eigenbasis (as mentioned above) as a prototypical application. Ultimately, it is found that for this high-accuracy, high-success probability regime, it is more advantageous and less costly to use window functions than it is to use QSVD methods. In Section II, we detail recent developments for both methods and their use in QPE. In Section III, we detail the numerical circuit simulation results comparing the impact of these options on the total success probability and resource cost of the algorithm. Finally, we comment on the broad utility of methods like QSVD for QPE in the conclusion.

II. RECENT DEVELOPMENTS ON QPE

Performance metrics for QPE have been extensively studied in the literature [13, 29–31]. Choosing an appropriate cost function (e.g. minimizing the mean-square-error or the Holevo variance) is highly dependent on the task-at-hand, i.e. estimating the eigenphase from measurements or using the QPE coherently as a subroutine. Among these metrics, the *success probability* of QPE is an important cost function for cases in which a higher-level algorithm or application uses QPE coherently. In our work, we define the success probability of QPE as the sum of probabilities corresponding to the two nearest m -bit fixed-point approximations to the true eigenphase, corresponding to the probability of obtaining an m -bit estimate φ of the true eigenphase ϕ of a unitary U such that $|\varphi - \phi| \leq \frac{1}{2^m}$.² For the textbook implementation of QPE, this success probability is bounded by $P_{\text{succ}} \geq \frac{8}{\pi^2} \approx 0.81$ [32, 33]. One way to verify this notion of success probability is by simulating the output probabilities of the m phase qubits used in QPE. This results in a histogram of 2^m bins, in which each bin in the histogram corresponds a bitstring representation encoding a binary fixed-point number. In practice, we take a finite number of measurements, resulting in an estimated histogram.

A standard method of boosting the success probability given an instance of QPE is to add to the number of phase qubits m then ignore the additional qubits.³ That is, adding phase qubits fine-grains the histogram of QPE outcomes and doubles the number of bins for each additional qubit. By ignoring the additional qubits, say p of them, this is effectively coalescing the bins from 2^{m+p} to the original 2^m bins and summing up the probabilities from the combined bins. In this section, we describe two alternatives, more recent developments that boost the success probability of QPE, namely the use of window or taper functions [13, 19, 34] and the quantum singular value transform (QSVD) [17, 18].

A. Quantum phase estimation using window functions

Studies of window states in QPE originate from ideas in quantum metrology [13, 29–31], where they were often referred to as *control states*. The optimal control state for minimizing a particular cost function of QPE, namely the Holevo variance, was derived by Luis et al [29].⁴ In this section, we summarize an alternative yet likely related interpretation of QPE in the language of signal processing, as illustrated in Fig. 1 and as is suggested, for instance, in Section II of [19]. We first motivate window functions in signal processing before describing their connection to QPE.

² The two closest binary strings are equally good approximations only in the case that the true phase lies precisely halfway between them: for many phases there will be an unambiguous closest m -bit approximation to the true phase ϕ . This satisfies a slightly different inequality $|\varphi - \phi| \leq \frac{1}{2^{m+1}}$. Typically the number of bits m is chosen such that the precision set by $\frac{1}{2^m}$ is sufficient and hence measuring either the closest or second closest bitstring is acceptable.

³ This method can be applied in the case of the reflection about the ground state mentioned above by making use of a multi-controlled Z gate with zero and one controls corresponding to the binary representation of the previous m -bit estimation of the eigenphase. For more detail, readers should refer to Ref. [28].

⁴ In this work, we refer to this control state as the “sine window” as shown in Table I.

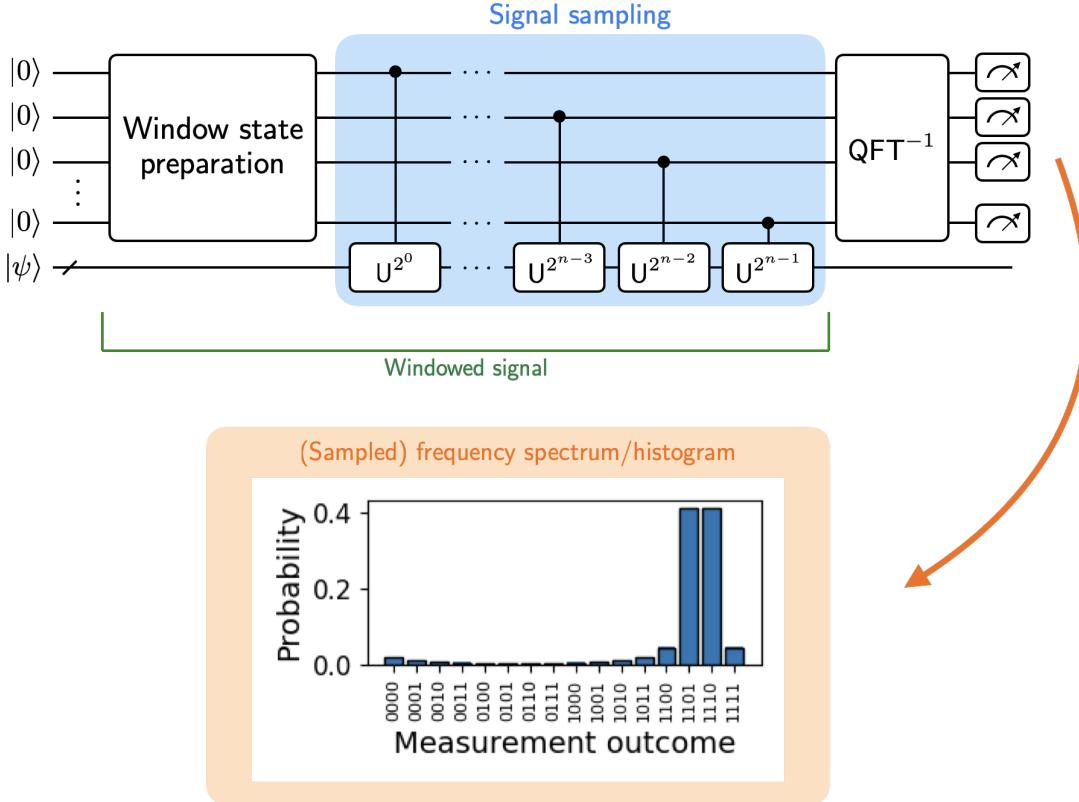


Figure 1. Quantum phase estimation re-interpreted using signal processing language. A window state is first prepared on the phase register. Next, the signal is sampled (in time) by applying controlled-unitaries that contain information about the true phase, resulting in preparation of a windowed signal. Following the inverse quantum Fourier transform and measurement, we obtain a *histogram* of the QPE outcomes corresponding to the frequency spectrum of the windowed signal by repeated measurement of the phase register. By choosing an appropriate window function, the output spectrum can have reduced spreading of frequencies which appear due to time-limited sampling and windowing.

In signal processing, we may observe *spectral leakage* or introduction of new frequency components when we take the Fourier transform of a time-limited (or truncated) signal [35]. By truncating in the time domain, one may introduce discontinuities at the ends of the signal. The Fourier transform assumes the sample to be periodic and will thus introduce unexpected frequency components due to the discontinuities. To reduce this leakage, a *window* or *tapering* function is chosen and multiplied to the sampled signal. A window function is often tapered at its ends, thus multiplying a signal by this function smooths the discontinuities at the ends of the sampled signal. However, a window function is characterized by its own set of frequencies and therefore, multiplying the window function and the sampled signal in the time domain corresponds to convolving in the frequency domain, resulting in some spreading of frequencies in the output spectrum.

In the textbook version of quantum phase estimation, as illustrated in Fig. 1, we first apply Hadamard gates on the phase register of size m , preparing a uniform superposition. This, in fact, realizes a rectangular or ‘‘boxcar’’ window function on the amplitudes. However, we can replace the Hadamards with a different state preparation unitary. We will call the resulting state a *window state* but will use this term and ‘‘window function’’ interchangeably. After preparing the window state and assuming the input eigenstate $|\psi\rangle$ has already been (perfectly) prepared, a sequence of controlled- U^{2^k} is applied where $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$ and k runs from 0 to $m - 1$.⁵ Lastly, the inverse quantum Fourier transform (QFT) is applied to the signal to obtain the histogram of measurement outcomes. In practice, this histogram is estimated by taking some finite number of measurements.

In the case that the true phase ϕ is expressible in m bits or fewer, the signal is periodic on the interval, where we have assumed a rectangular window state. Thus the phase value can be extracted after applying the inverse QFT.

⁵ Note that the window state in QPE has an additional role of preparing a state that can activate the controls in the controlled- U^{2^k} gates. Thus, the window state preparation must come before the application of controlled-unitaries that corresponds to preparing a (time-limited) signal. Alternatively, one could view sending $|0\rangle$ into QPE as preparing a poor choice of window function.

However, for generality, we assume ϕ to be an irrational number. In such cases, applying up to controlled- U^{2^m} leads to preparing a time-limited signal that is not periodic on the interval. Applying the inverse QFT on the time-limited sampled signal results in a frequency spectrum that has a spread about the true phase value due to spectral leakage. To reduce this leakage, as is done in signal processing, we can consider preparing an alternative window state on the phase register. In the literature, several window states have been proposed: cosine [19], sine [13, 29, 30], Kaiser [34], and DPSS [20] windows, also summarized in Table I. It was shown that of the aforementioned windows, the Kaiser window has the most favorable scaling in terms of the number of additional phase qubits required in QPE to achieve an error to within $1/2^m$ [34]. We provide a sketch of deriving the number of additional qubits for the Kaiser window in Appendix B.

To illustrate the capabilities of the Kaiser window, we consider a toy example of a four-qubit QPE in Fig. 2 with the true phase value that sits equally between two bins. The Kaiser window state is defined as follows:

$$|\psi\rangle_{\text{Kaiser}} = \sum_{x=-2^{m-1}}^{2^m-1} \frac{1}{2^m} \frac{I_0(\pi\alpha\sqrt{1-(x/2^{m-1})^2})}{I_0(\pi\alpha)} |x\rangle, \quad (1)$$

where $I_j(\cdot)$ is the modified Bessel function of the first kind of order j . This window state has a parameter α that can be used to tune the tapering capability, as shown in the first column of Fig. 2. When α is very small, as in the first row of the figure, this approaches the rectangular or boxcar window state. When α is large, e.g. 200, this corresponds to a fast-decaying taper function. This raises the question of how to select the right α for the Kaiser window or more broadly how to choose an appropriate window state. Fortunately, we can refer to existing techniques in signal processing for analyzing and choosing an appropriate window state.

In the field of signal processing, several families of window functions have been proposed. These functions can be evaluated by analyzing their frequency spectra, extracting properties such as width of the main lobe or the height and decay rate of the side lobes. In the middle column of Fig. 2, we show the frequency spectra of Kaiser windows with different α values. The main lobe corresponds to the central peak. Main and side lobes of a window function can help inform the degree of spreading of frequencies expected in the final frequency spectrum, which in our context is the histogram of QPE outcomes. For instance, using a window state with a wide main lobe results in a relatively wide spread of frequencies (or phase outcomes) about the true frequency (phase). On the other hand, a tall and slowly decaying side lobe results in spreading of frequencies throughout, including frequencies further away from the true phase. In practice, there is a trade-off between main lobe width and side lobe height. In the last column of Fig. 2, we show the outcomes of the four-qubit QPE produced using state vector simulations. We can compute the success probability of this QPE by summing up the heights or probabilities of bins corresponding to 0.8125 and 0.8750. The frequency spectrum of the Kaiser window where $\alpha = 10^{-5}$ (effectively a rectangular window) features a relatively narrow main lobe but tall and slowly decaying side lobes. Thus, in the corresponding QPE histogram, this results in some spreading of phase values away from the true phase. On the other end, when α is large, e.g. 200, the main lobe dominates, and we observe significant concentration (though broadened) of phase values about the true phase. If we set α to 25, the main lobe width is similar to the case where $\alpha = 10^{-5}$, but the side lobe heights are decreased. Applying this window state reduces the spreading of phase values further away from the true phase. We note that of the three cases, the QPE using the Kaiser window with parameter value $\alpha = 25$ has the highest success probability *as well as* reduced side lobes. This demonstrates the importance of tuning α to an appropriate value for the window state to be effective. We re-visit the discussion on choosing an appropriate α in Appendix A 2.

Lastly, in the context of QPE, we must also consider the cost of preparing window states. Fortunately, costs of the sine and cosine window functions are considerably low, as discussed in [13] for the sine window. We show the circuit for preparing the cosine window state in Fig. 8, in which we expect the total cost of the controlled-unitaries U or block-encodings to be significantly higher. We also provide some rough cost estimate for the Kaiser window based on previous works in Appendix E and again show that its cost may be insignificant compared to the total costs of the controlled-unitaries or block-encodings.

While window functions in QPE are well-motivated, and several previous studies have focused on their asymptotic performance [20], there is still a lack of understanding in how they compare against other existing improvements to QPE such as the use of the quantum singular value transform (QSVT). In the following section, we briefly outline the theory behind QSVT-QPE before comparing the numerical performance of the two approaches via state vector simulations.

B. Quantum phase estimation using QSVT

As shown in the previous section, the success probability of QPE with respect to bit discretization error can be boosted by making an appropriate choice of prior distribution over the states of the phase qubits before performing

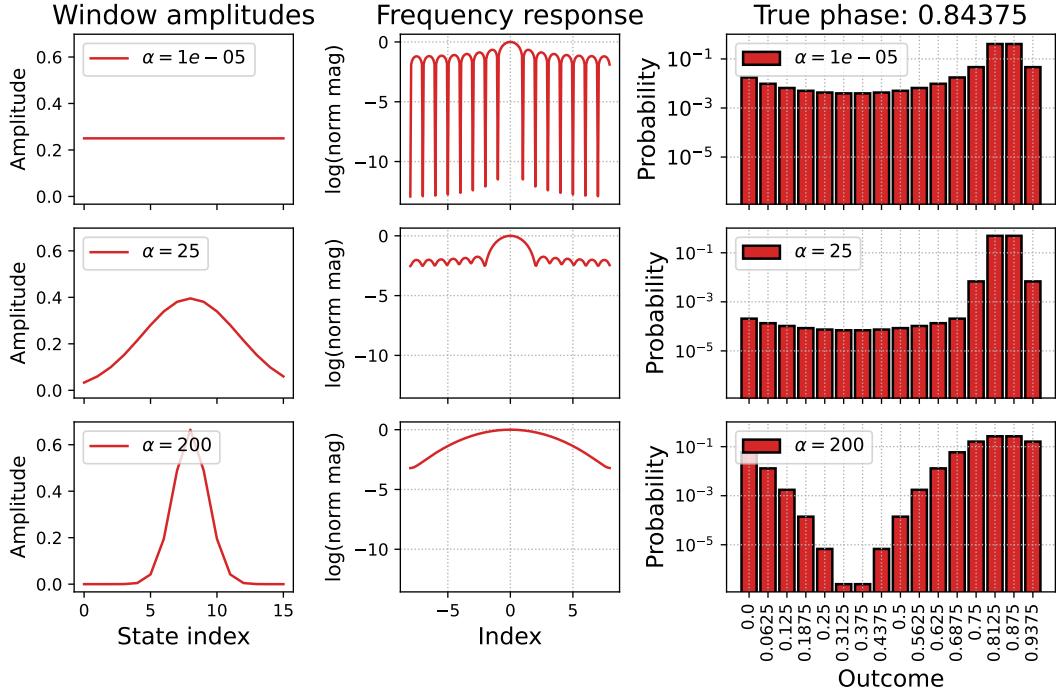


Figure 2. Numerical analysis of the Kaiser window for QPE. This toy QPE example uses four bits of precision, and the true phase is 0.84375, which requires 5 bits to express. This value lies exactly in between two bins (or bitstrings) representing 0.8125 and 0.875. We investigate three different instances of the Kaiser window, in which the parameter α is varied. The first column shows the window function amplitudes for a four-qubit state (i.e. the QPE uses four bits of precision). The second column shows the frequency spectrum of each window state, in which one can see that varying α affects the side lobes in the Fourier spectrum. The third column shows the output probabilities of phase values from simulating the QPE circuit.

Name/Ref	Window function form	Additional qubits
Rectangular	$\sum_{x=0}^{2^m-1} \frac{1}{\sqrt{2^m}} x\rangle$	$\mathcal{O}(\log(1/\delta))$ [32]
Cosine [19]	$\sum_{x=-2^{m-1}}^{2^{m-1}} \frac{\sqrt{2} \cos(\frac{\pi x}{2^m})}{\sqrt{2^m}} x\rangle$	$\mathcal{O}\left(\log\left(1/\delta^{1/3}\right)\right)$
Sine [13]	$\sum_{x=0}^{2^m-1} \sin\left(\frac{\pi(x+1)}{2^m+1}\right) x\rangle$	Similar to cosine window
Kaiser [34]	$\sum_{x=-2^{m-1}}^{2^{m-1}} \frac{1}{2^m} \frac{I_0(\pi\alpha\sqrt{1-(x/2^{m-1})^2})}{I_0(\pi\alpha)} x\rangle$	$\mathcal{O}(\log\log(1/\delta))$

Table I. Summary table of select window functions. Here, m is the number of phase qubits. Depending on the window function expression, the index x runs from either 0 to $2^m - 1$ or -2^{m-1} to 2^{m-1} , but their corresponding basis states are the same. The Kaiser window function has a tunable parameter α that can balance the effects of the main lobe width and side lobe height in the frequency spectrum. The function forms of these window functions may be un-normalized. The column “Additional qubits” shows the number of phase qubits one could add then discard to achieve a success probability of $1 - \delta$, i.e. the probability of getting an error within $\frac{1}{2^m}$. We expect the additional qubits required for the sine window to be similar to that of the cosine window. We show a sketch of the proof for the number of additional qubits for the Kaiser window in Appendix B.

the phase kickback. This strategy does not alter the eigenphase kicked back by the controlled unitaries in QPE, but rather uses the statistical properties of the state stored in the phase register to achieve the higher success probability. An alternative approach would be to instead alter the unitary whose eigenphase is being kicked back such that the bit discretization error is reduced or (ideally) eliminated. Intuitively, we wish to implement a modified unitary \tilde{U} whose eigenvalues correspond to those of the ideal unitary, truncated precisely to m bits of precision such that the bit discretization error of the modified unitary is zero.

Such a modified unitary can be realized as a matrix function of the original target unitary U , which itself may be implemented using the *quantum singular value transform* (QSVT), a protocol for implementing matrix polynomials on a quantum computer – for the interested reader, a brief overview of QSVT is provided in Appendix C. The main idea behind QSVT is to implement a block encoding of some input unitary

$$\begin{bmatrix} f(U) & * \\ * & * \end{bmatrix}, \quad (2)$$

where the elements labelled $*$ are left undefined and where the function f acts on the singular values of U . In Ref. [18], the authors present a method for using QSVT to boost the success probability of QPE, a routine that we refer to hereafter as *QSVT QPE*. The core of that routine revolves around using QSVT to implement the shifted sign function

$$f(x) = \Theta\left(\frac{1}{\sqrt{2}} - x\right), \quad (3)$$

with $\Theta(x)$ denoting the sign function, using the notation of Ref. [18] (not to be confused with the Heaviside step function). At a high level, one can think of this function as performing the transformation described above, clipping phases that cannot be represented by m bits such that the transformed eigenphase can be represented exactly using m bits. For the interested reader, a detailed overview of the choice of function is given in Appendix D.

One subtlety is that this construction requires that at each iteration, the less significant bits in the eigenphase i.e. those that were previously measured in the QPE protocol, should be rotated out from the unitary so they are not measured again. In textbook QPE, this is implemented by the controlled phase rotations in the inverse quantum Fourier transform. This is a well-known technique that underpins *iterative* quantum phase estimation [32], although unlike iterative methods, the routines we consider here are performed coherently. The specific circuit details for implementing this algorithm are outlined in Appendix A.

C. Challenges of QSVT in practice

The principle challenge associated with implementing routines based on QSVT lies in obtaining the phase factors that implement a given function. There is no closed-form expression for these phase factors for a given general target function, and so numerical methods must be used to obtain them. There are two approaches for this: optimization-based [36] and non-optimization-based methods [16, 37–39]. In practice, we find that the non-optimization-based methods are less numerically stable and do not provide a significant advantage in terms of accuracy or speed compared with optimization-based methods, so we use optimization in this work. For the numerical results presented here, we used a simple `scipy` minimization of the absolute mean squared error between the signal processed unitary matrix and the target function; we also evaluated the performance of the QSPPACK library [40], but found that it did not make any significant difference to the results, possibly due to the relatively low degrees (and therefore relatively simple optimization landscape) explored herein.

The typical workflow for implementing a function in QSVT is to use the function we wish to implement as the target for an optimization, with the phase angles as the optimization parameters. However, since the shifted sign function is discontinuous, an additional decomposition step must be made to obtain a continuous and symmetric approximation to the sign function. The decomposition used in this work [18]⁶ is

$$P_{\Delta,\kappa}(x) := -\frac{1}{1 + \frac{\Delta}{4}} \left(-1 + \frac{\Delta}{4} + P_{\frac{\Delta}{2},\kappa}^{\Theta}\left(\frac{1}{\sqrt{2}} - x\right) + P_{\frac{\Delta}{2},\kappa}^{\Theta}\left(\frac{1}{\sqrt{2}} + x\right) \right), \quad (4)$$

where

$$P_{\frac{\Delta}{2},\kappa}^{\Theta}(x) := \text{erf}\left(\frac{\sqrt{2}}{\kappa} \sqrt{\log\left(\frac{2}{\pi\Delta^2}\right)} x\right). \quad (5)$$

These functions have two parameters which control the quality of the approximation: Δ , which controls the maximum deviation away from the true function and κ , the region around the discontinuity where the error is allowed to be arbitrarily large. These two parameters control the quality of the approximation, and typically the lower their values,

⁶ Note that our notation differs slightly from that presented in Ref. [18] – our κ corresponds to their Δ , and our Δ corresponds to their ϵ .

the higher the polynomial degree needed to implement it (and hence the higher the gate cost). In Ref. [18], it is shown that a value of $\kappa < 2 \left(\cos\left(\frac{3\pi}{16}\right) - \frac{1}{\sqrt{2}} \right) \approx 0.25$ suffices to yield a high success probability. For Δ , it is shown that a value of $\Delta \leq \sqrt{2\delta/(m+1)}$ (where m is the number of bits of precision) is required to obtain a success probability greater than $1 - \delta$. The choice of the polynomial degree for the QSVD function acts as a parameter establishing a tradeoff between cost and performance, analogous to the number of additional bits p used in window functions.

Obtaining values for the function parameters is not the end of the story, however. One must then obtain the optimal phase factors that realize this function. In practice, while we find that the analytical values provide a good starting point for an optimization, it is often useful to vary these values to obtain as small a value for Δ as possible for a given degree. In this work, the QSVD QPE results correspond to fully compiled decompositions with the phase angles obtained through such an optimization.

III. SIMULATION RESULTS

In order to evaluate and compare the impact of window functions and QSVD on QPE, it is useful to consider numerical state vector simulations of the full subroutine, which we implement using a proprietary state vector simulator. The details of the specific circuits used in this work (including corrections to the original QSVD QPE circuit) are outlined in Appendix A. Throughout, we use a simple phase gate to generate the eigenphase we wish to read:

$$P(\phi) := \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i \phi} \end{bmatrix}. \quad (6)$$

The simplicity of this unitary does not affect the generality of the results presented here – the reduction in success probability from unity arises due to bit discretization errors associated with representing the eigenphase with a finite number of bits (assuming, as we do throughout, that we are working with an exactly-implemented input eigenstate). As a result, the same phenomenon will occur regardless of which unitary gives rise to the eigenphase being measured, and therefore the results we present here should be applicable even for large-scale applications. The specific system being investigated will impact the precision required to obtain a result to a desired accuracy (for instance, unitary encodings of Hamiltonians with large norms require more bits of precision than those with smaller norms to achieve the same accuracy), but this is a separate concern from the core question investigated here – explicitly, the setting for this work is: given sufficient bits of precision m to achieve a desired accuracy $\epsilon < 1/2^m$, how many *additional* resources are needed to achieve a high probability that the measured phase is ϵ -close to the true phase?

There are two additional considerations that need to be pinned down in order to obtain concrete numerics: the desired success probability and the method by which it is evaluated over the full range of possible eigenphases. Since the performance of any QPE routine will vary (often significantly) for different target eigenphases, this analysis only makes sense if we consider the full range of phases when deciding if the success probability is sufficiently high. In real applications, we will not know the true eigenphase (this is one of the major reasons for doing QPE in the first place!) and so we cannot know if our target lies in some fortunate, highly performant region of the possible phases, or if it happens to be some pathological case with significantly lower success probability. As such, we consider the minimum success probability over this full range to be the most important value for these numerics. Here, we choose the additional resources (either additional phase qubits for the window function QPE routines or the polynomial degree in the case of QSVD QPE) to be the minimal resources necessary to achieve a minimum success probability of 99%. This value is, in some sense, arbitrary, but we find our results remain consistent for other values.

A. Success probability

In this section, we present results from numerical experiments, highlighting the relative success probabilities of using window states or QSVD in QPE. Fig. 3 shows the success probabilities for different five phase bit QPE implementations as a function of the $P(\phi)$ eigenphase. As previously motivated, the primary figure of interest for this work is the minimal success probability (or equivalently, one minus the maximum failure probability) achieved over the full range of possible eigenphases. By this metric, the best implementation by a wide margin is the QPE using a Kaiser window function and four additional phase qubits, achieving a maximum failure probability of $10^{-7.28}$, followed by QPE using a cosine window and four additional qubits, achieving a maximum failure probability of $10^{-5.07}$. QSVD QPE achieves a maximum failure probability of $10^{-2.28}$, with the rectangular window function having the worst performance, achieving a maximum failure probability of $10^{-2.2}$.

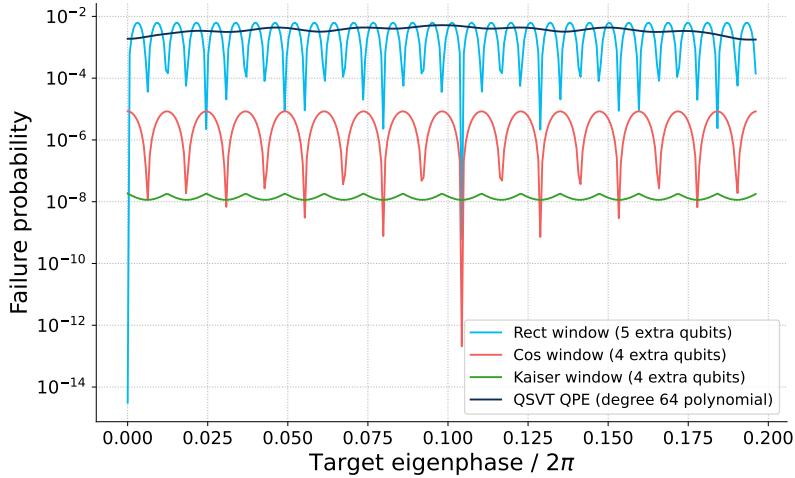


Figure 3. Success probability as a function of the measured eigenphases for quantum phase estimation with different modifications. The dark blue line represents QSQT QPE with a $d = 64$ degree polynomial as outlined in Ref. [18], and the sky blue, red, and green lines represent QPE with rectangular, cosine and Kaiser windows respectively. For the rectangular window, $p = 5$ additional qubits were used, while for the cosine and Kaiser windows $p = 4$ additional qubits were used; for QSQT QPE no additional phase qubits were used. All the QPE routines use $m = 5$ phase qubits to load the approximation of the eigenphases.

B. Costs of the different implementations

From the results in Fig. 3, one may conclude that all considered QPE implementations achieve very high success probabilities. The Kaiser window appears to achieve the highest success probabilities over possible values of the target eigenphase, but other implementations also achieve values close to 1. However, to establish a fair comparison, we additionally provide the relative costs of different QPE implementations, specifically the numbers of unitaries called in QPE. We assume that the unitary is a *block encoding* U_A of some matrix A , defined as

$$U_A := \begin{bmatrix} A & * \\ * & * \end{bmatrix}, \quad (7)$$

where the elements labelled $*$ are left undefined. This block encoding is representative of the unitaries that are commonly used in fault-tolerant quantum algorithms that make use of QPE [13, 15].

Fig. 4 gives some proper context to the results shown in Fig. 3: while QSQT QPE performs fairly well in terms of minimal success probability, its cost is by far the largest out of the implementations considered, at almost 1984 calls to U_A . The rectangular window function is also costly, albeit much less so than QSQT QPE, coming in at 1023 calls to U_A . By far the cheapest routines were the cosine and Kaiser window function implementations, with 127 calls to U_A each. These low costs reflect the fact that only 4 additional phase qubits are needed to achieve a success probability ≥ 0.99 , as opposed to the 5 that are required for the rectangular window implementation. It should be noted that, asymptotically, the Kaiser window should perform exponentially better than the cosine window, scaling as $\log \log(1/\delta)$ as opposed to $\log(1/\delta)$ as shown in Appendix B. However, since the number of phase qubits required by the cosine window implementation is relatively low, at only 4 additional phase qubits, this asymptotic advantage is not yet realized (i.e. the Kaiser window with only 1 additional phase qubit failed to achieve a consistent success probability greater than 0.99), likely due to the neglected constant factors in the asymptotic expressions. We show the crossover point between the two window functions in the following section.

For QSQT QPE, the cost is significantly higher due to the number of calls to U_A required to implement the sign function. This disparity of costs is unlikely to be overcome simply by obtaining a better polynomial approximation or performing a better optimization, which can be seen by explicitly calculating the maximum allowable degree as constrained by the cosine and Kaiser window costs. The cost for both these implementations is $2^{m+p} - 1 = 2^{5+4} - 1 = 511$. In order to yield a QSQT QPE routine with a total cost less than this, we can set the degree to $d = \lfloor 127/(2^m - 1) \rfloor = \lfloor 511/31 \rfloor = 16$. Thus, in order for QSQT QPE to be competitive with the cosine and Kaiser window functions, a degree-16 approximation to the sign function is required to yield a maximum failure probability less than 10^{-7} . Even with improvements to the QSQT QPE protocol, it seems unlikely that this performance gap can be overcome given the additional resources available.

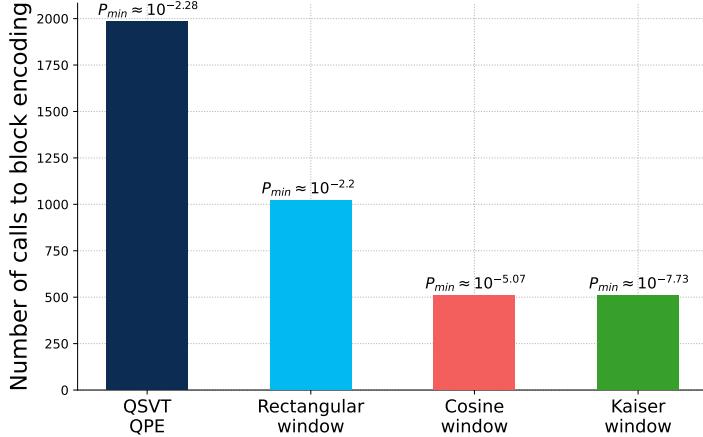


Figure 4. Graph of the cost in terms of queries to the block encoding unitary for QSFT QPE with a degree $d = 64$ polynomial (green), vanilla QPE with a rectangular window function (dark blue), vanilla QPE with a cosine window function (red) and vanilla QPE with a Kaiser window function (sky blue). All results are for $m = 5$ bit quantum phase estimation routines, with the degree of the polynomial approximation (for QSFT QPE) and the number of additional phase qubits (for the window functions) chosen such that the minimal success probability was greater than 0.99. Numbers above the bars are the maximum failure probability, or one minus the minimum success probability, over possible values of the target eigenphase (see Fig. 3).

C. Pushing the success probability even higher

The above analysis shows that the use of window functions (and in particular the Kaiser window function) yields QPE routines with higher success probability than QSFT QPE, but the chosen value of 0.99 for the success probability may not be indicative of the performance of these subroutines in real applications. For algorithms that make use of coherently controlled QPE as a subroutine, there is often a correlation between the success probability of the QPE routine and the performance of the overall algorithm (although this is not the only factor to consider when designing such an algorithm). A pertinent question is therefore whether the window functions maintain their higher performance over QSFT QPE at higher success probabilities, or whether there is some cross-over point after which it becomes favorable to utilize QSFT QPE.

As a first step towards answering this question, we evaluate the impact of increasing the number of additional phase qubits (that are subsequently discarded) for the two additional window functions considered here, the cosine and the Kaiser windows. Fig. 5 shows the maximum failure probability for these two windows as a function of the number of additional phase qubits. The cosine window shows an exponential improvement in failure probability with additional phase qubits, reaching a minimum failure probability of 10^{-7} using 6 additional phase qubits, while the Kaiser window shows an exponential improvement *over that*, with the obtained failure probabilities being limited by floating point error after only 4 additional phase qubits. This performance closely matches the expected asymptotic performance, as shown by the fits to the expected $\log(1/\delta)$ and $\log \log(1/\delta)$ scalings for the cosine and Kaiser windows respectively that are shown in Fig. 5.

It is worth highlighting just how overwhelmingly more performant the window functions are over QSFT QPE: 6 additional phase qubits is only slightly more expensive than performing a 64 degree polynomial approximation in QSFT QPE, which yields a maximum failure probability of approximately 10^{-2} as shown in Fig 3 compared with 10^{-7} for the cosine window. For the Kaiser window, the limitations of double precision floating point arithmetic mean that the largest number of additional phase qubits with reliable success probabilities is 4, corresponding to a QPE routine with a query cost approximately 1/4 that of the 64-degree polynomial. Despite this, the achieved failure probability is less than 10^{-7} , some five orders of magnitude less than the QSFT QPE results. One may argue that QSFT QPE may be more advantageous at extremely low failure probabilities (for example that resulting from the 10^{-30} target polynomial error used in Ref. [17]). However, we argue that such a cross-over is unlikely given the numerical evidence presented here, and that even if it does occur, it would be well beyond the success probabilities needed for practical applications.

The data in Fig. 5 were obtained by evaluating the failure probability across the periodic range of eigenphases in $[0, \frac{2\pi}{\Delta}]$ and taking the maximal failure probability. These data therefore represent the *worst-case* failure probability and therefore one would expect to do even better than this in the typical case (although one cannot predict where the

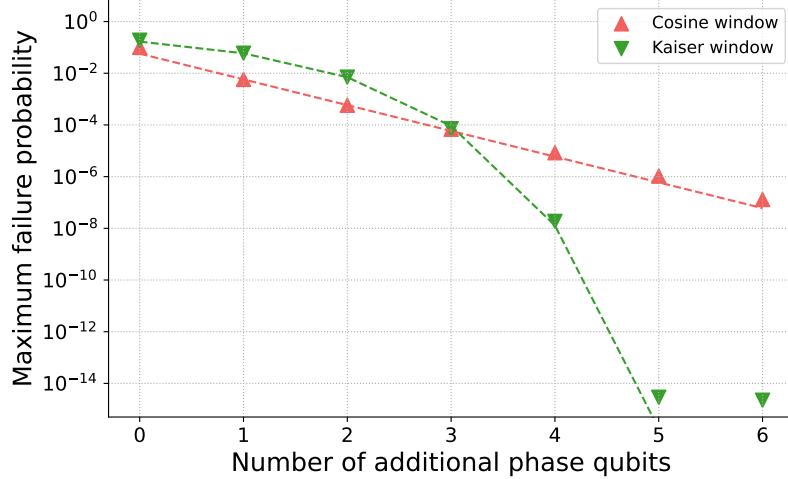


Figure 5. Maximum QPE failure probability (defined as $1 - P_{\text{succ}}$) as a function of the number of additional phase qubits that are added and then discarded to obtain a more accurate estimate of the eigenphase using the cosine (red triangles) and Kaiser (light blue inverted triangles) window functions. P_{succ} is the minimum success probability. Both show an exponential scaling from one additional phase qubit onwards, although the Kaiser window performs significantly better. For 5 and 6 additional phase qubits, the Kaiser window failure probabilities are at the threshold for floating point error and so those values should not be taken as being representative of the QPE performance with those numbers of qubits. The estimates of the eigenphase were made using 5 bits of precision. The fitted lines show the expected $\log(1/\delta)$ and $\log \log(1/\delta)$ scalings for the cosine and Kaiser windows respectively (see Table I).

true eigenphase lies, and so this worst-case bound is the most appropriate figure of merit for evaluating the viability of algorithms). It should also be noted that for QSVD QPE, the assumption that the failure probabilities are periodic is not valid due to the choice of polynomial approximation, with this choice of eigenphases being the worst-case choice for QSVD QPE – Appendix F outlines the reason for the breaking of this symmetry.

D. Numerical success probability scaling with bits of precision

In the previous sections, we presented numerical evidence that the success probability of QPE can be increased by either using window states or by using QSVD as outlined in Ref. [18], arguing that window functions use far fewer resources to accomplish similar increases in success probability over the standard QPE routine. However, a natural question arises as to whether these results are reliable for all domains of interest – the number of bits of precision required to estimate an eigenphase to a given precision varies significantly (e.g. in quantum chemistry applications, it scales with the norm of the system Hamiltonian) and in practice can be large enough to make classical simulation impractical. In this section, we present numerical evidence that allows us to make a heuristic argument about the performance of these strategies for larger numbers of phase qubits.

We evaluate the scaling of the different QPE methods by numerically generating the success probability for different values of eigenphases for m bits of precision from 1 to 14, once again using the phase unitary as the test bed. Although this is likely fewer bits of precision than would be required to estimate, for example, the ground state of FeMocco to chemical accuracy [15], we argue that there is no a priori reason to believe that the scaling observed for these parameters should break down at higher system sizes. For the window functions, an efficient emulation was implemented that allowed us to sample 10 000 points along a range of eigenphases corresponding to a period of success probabilities, or $[0, 1/2^n]$ for $n = m+p$ total phase qubits, while for QSVD QPE a full statevector simulation of the circuit using 10 000 points over the full domain $[0, 2\pi]$. The reason for this discrepancy is that the periodicity of the QPE routine is broken by the imperfect polynomial approximation (see Appendix F for details), meaning that unlike the window functions, sampling over a single period does not result in comparable success probabilities to sampling over the entire domain. Fig. 6 shows the results of this numerical evaluation. The performance of the window functions is independent of the number of bits of precision required, with both the mean and standard deviations remaining constant for all simulations. This provides a significant boost to the utility of these functions: if one needs some guarantee on the accuracy of their QPE routine, then they can determine how many additional qubits will be required using only a small numerical model.

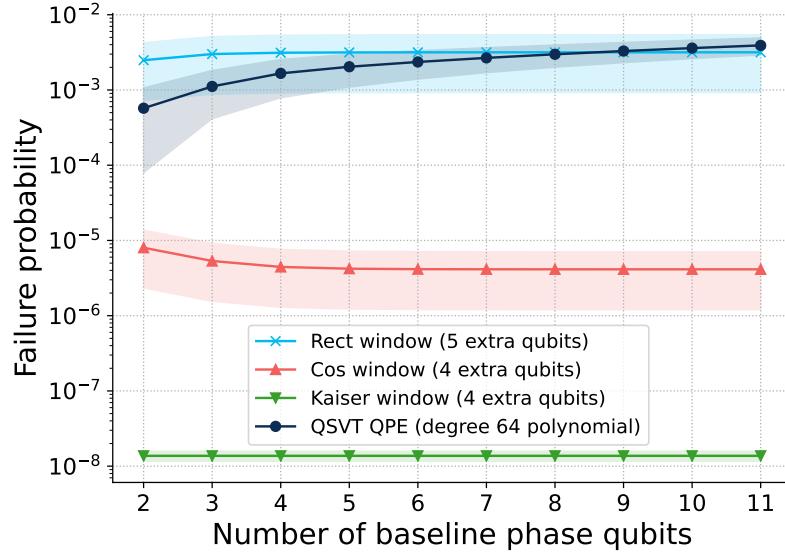


Figure 6. QPE success probability for the three different window states (rectangular, cosine and Kaiser) and QSVT QPE as a function of the number of bits of precision in the phase register. The points indicate the mean of the success probabilities evaluated over 10000 points evenly distributed in $[0, \frac{2\pi}{2^b}]$ and the shaded regions correspond to the standard deviations of the same data. For all three window states, both the mean and the standard deviations remain constant over all bits of precision considered, providing solid evidence that they will remain highly effective even at the larger register sizes required for practical applications. QSVT QPE shows a linear decrease in mean success probability.

For QSVT QPE, the success probability decreases as the number of bits of precision increases, although it remains high for the system sizes considered here. This decrease means that the success probabilities obtained in the data presented in the main text will not be representative of the performance at large numbers of bits of precision, but rather form an upper bound on the performance. That being said, a reasonable guess as to the required degree can be made by extrapolating the plot out to larger system sizes from data such as this.

The reason that the success probability of QSVT-QPE decreases as a function of the number of bits of precision is due to the fact that the number of applications of the function (and the exponentiated unitaries) increase as a function of the number of bits. Each QSVT application remains in the desired eigenspace with probability approximately equal to $1 - \Delta$ (ignoring the region of discontinuity around phase values close to $1/\sqrt{2}$, which will further decrease the success probability), meaning that for m bits of precision, the overall probability of successfully implementing the desired function, rather than projecting into some orthogonal eigenspace, is $1 - \delta \approx (1 - \Delta)^m$. Thus, for any finite degree approximation to the sign function, the success probability decreases exponentially in the number of phase bits. By contrast, the window functions implement a well-defined state on the phase register regardless of the number of bits of precision, and therefore the performance of the window functions is independent of the number of phase bits (although the resources required to implement the window function will of course depend on the size of the phase register, and in the worst case exponentially – this is relatively unproblematic however, since the number of block encoding applications also scales exponentially with the number of bits of precision, and the cost of even a single block encoding will typically dwarf the cost of implementing the window functions). Given that the main results presented here show that window functions achieve significantly better success probabilities with lower costs than QSVT QPE, this provides further evidence that QSVT is not an effective strategy for increasing the performance of QPE.

IV. CONCLUSIONS AND OUTLOOK

In this work we compare the effectiveness of two approaches towards increasing the success probability of QPE, a property that will need to be maximized in order for coherent usage of the subroutine to be possible.

Utility of QSVT QPE and window functions Our numerical results show that while QSVT QPE performs as well as expected from previous asymptotic analysis, it is significantly outperformed by using window functions together with additional phase qubits. QSVT QPE using a degree 64 polynomial yielded a slightly higher minimum success probability than the rectangular window function with 5 additional phase qubits, but required almost twice

the number of calls to the block encoding. The performance gap is even wider when comparing QSVT QPE to the cosine and Kaiser window functions, which achieve substantially higher success probabilities for 1/4 the query cost. The Kaiser window shows a particularly overwhelming improvement in performance compared with QSVT QPE, with the failure probability hitting the precision floor for double precision floating point numbers with only $m = 5$ additional phase bits, still half the query cost QSVT QPE requires to achieve a success probability of 0.99.

These results indicate that QPE is not an algorithm which benefits significantly from employing QSVT as a subroutine. It is possible that other attempts to combine QSVT and QPE could have some advantage – the implementation presented in Ref. [17] has an improved asymptotic scaling, for example. However, given the performance of the Kaiser window and the extremely high success probabilities it was able to achieve with fewer than 6 additional phase qubits, the likelihood is that for most applications, the Kaiser window will be more than sufficient for implementing QPE with high success probabilities.

Future directions There are many future directions that could be taken to expand upon the work we present here. One idea would be to combine QSVT QPE with the window functions in order to potentially improve the performance of both. Using the QSVT QPE framework as is, we do not expect the two methods to be compatible as QSVT-QPE uses a thresholding function to reduce ambiguity in determining each bit value. This corresponds to the bit sitting exactly on top of a bin, a case in which using the rectangular window function is optimal. We attempted a simple implementation of this by taking the QSVT QPE routine and using the cosine window in place of the rectangular window, which resulted in a marked decrease in performance compared with the standard QSVT QPE routine. It may be possible that some modification to the QSVT QPE routine could be made that directly accounts for the different window functions. The viability of such an approach would be an interesting study for future work. Additionally, it would be worth verifying whether the alternative implementations of QSVT QPE mentioned above can outperform the implementation investigated here. However, given the remarkable performance of the Kaiser window, in which the failure probability using four extra bits already approaches the floating point error, it seems unlikely that any implementation would be more efficient than that. This would require the QSVT QPE to use a 16-degree polynomial for the same task to be competitive with the use of the Kaiser window.

The state vector simulations using the Kaiser window were not implemented with a full gate decomposition – rather they were implemented by directly injecting the desired state into the state vector before applying the QPE routine. A cost analysis for the implementation of the Kaiser window is given in Appendix E, but this is merely a worst-case upper bound on the cost, assuming that arbitrary state preparation is required. Given the significant amount of structure in the definition of the Kaiser window and the fact it can be exactly expressed in terms of Bessel functions, it is likely that more efficient implementations can be obtained. It would also be interesting to analyze the impact of implementing an approximation to the Kaiser window, which may allow for more favorable balances between cost and success probability.

In our simulations, we also assume that we are exactly implementing the input eigenstate such that the only error in the phase estimation arises from the finite bit precision of the phase register. In practise, such an assumption will not hold for real systems. It would therefore be instructive to investigate the properties of the different approaches when the input eigenstate is only approximately realized.

As a more general point, our results have highlighted a limitation with the applicability of QSVT in QPE. As a framework, QSVT is highly general, and can be applied to a wide variety of applications. It would be highly valuable to investigate whether other proposed algorithms, such as Ref. [41], making use of QSP or QSVT have similar limitations, or whether they can genuinely be relied upon to yield efficient quantum algorithms. This is especially true in light of the recent publication of generalized quantum signal processing [42] which relaxes many of the constraints on the target functions of QSVT and hence makes the technique more appealing for applications that can take advantage of this improvement. The target function used in QSVT QPE, however, already naturally accommodates these restrictions and so QSVT QPE would not benefit from this generalization.

Concluding remarks In this work we present a systematic numerical comparison of two different methods for increasing the success probability of QPE, one using window functions from signal processing and one using the quantum singular value transform (QSVT QPE). From these numerical results, using a Kaiser window appears to be the best method for improving the success probability of QPE, providing the greatest increase for the lowest cost. Such a conclusion would be difficult to ascertain from an analysis of the asymptotic scaling alone, highlighting the importance of concrete verification protocols in quantum computation. In addition to the practical utility for quantum algorithms designers our work entails, we hope this study spurs further work into both numerical verification procedures and the systematic evaluation of existing protocols in quantum computation.

AUTHOR CONTRIBUTIONS

SS proposed comparison of window functions and QSVT for QPE. SG and SS developed codes for QSVT and window functions, respectively. SG developed and ran simulations for QPE using QSVT and both SG and SS for window functions. WP approximated the cost for preparing the Kaiser window state. SG, WP, and SS discussed the project and wrote the manuscript.

ACKNOWLEDGEMENT

The authors would like to thank Eric Johnston for his support in developing and debugging our simulation code. We would also like to thank Mark Steudtner, Jessica Lemieux, Harriet Apel, and Sam Pallister for insightful discussions on QPE.

- [1] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (Association for Computing Machinery, New York, NY, USA, 1996) p. 212–219.
- [2] R. P. Feynman, Int. J. Theor. Phys. **21**, 467–488 (1982).
- [3] I. M. Georgescu, S. Ashhab, and F. Nori, Rev. Mod. Phys. **86**, 153 (2014).
- [4] P. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994) pp. 124–134.
- [5] P. W. Shor, SIAM Journal on Computing **26**, 1484 (1997), <https://doi.org/10.1137/S0097539795293172>.
- [6] A. Y. Kitaev, “Quantum measurements and the abelian stabilizer problem,” (1995), arXiv:quant-ph/9511026 [quant-ph].
- [7] D. S. Abrams and S. Lloyd, Phys. Rev. Lett. **83**, 5162 (1999).
- [8] C. Gidney, “Factoring with $n+2$ clean qubits and $n-1$ dirty qubits,” (2018), arXiv:1706.07884 [quant-ph].
- [9] D. Litinski, “How to compute a 256-bit elliptic curve private key with only 50 million toffoli gates,” (2023), arXiv:2306.08585 [quant-ph].
- [10] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, Molecular Physics **109**, 735 (2011), <https://doi.org/10.1080/00268976.2011.552441>.
- [11] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, Proceedings of the National Academy of Sciences **114**, 7555 (2017), <https://www.pnas.org/doi/pdf/10.1073/pnas.1619152114>.
- [12] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, Phys. Rev. X **8**, 011044 (2018).
- [13] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Physical Review X **8**, 041015 (2018), 1805.03662.
- [14] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush, Quantum **4**, 296 (2020).
- [15] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, PRX Quantum **2**, 030305 (2021).
- [16] A. Gilyen, Y. Su, G. H. Low, and N. Wiebe, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing , 193 (2019), 1806.01838.
- [17] P. Rall, Quantum **5**, 566 (2021), 2103.09717.
- [18] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, PRX Quantum **2**, 040203 (2021), 2105.02859.
- [19] G. Rendon, T. Izubuchi, and Y. Kikuchi, Physical Review D **106**, 034503 (2022), 2110.09590.
- [20] D. Patel, S. J. S. Tan, Y. Subaşı, and A. T. Sornborger, arXiv (2024), 2403.18927.
- [21] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).
- [22] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete, Nature **471**, 87–90 (2011).
- [23] M.-H. Yung and A. Aspuru-Guzik, Proceedings of the National Academy of Sciences **109**, 754 (2012).
- [24] J. Lemieux, B. Heim, D. Poulin, K. Svore, and M. Troyer, Quantum **4**, 287 (2020).
- [25] A. Montanaro, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **471**, 20150301 (2015).
- [26] A. W. Harrow and A. Y. Wei, in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, 2020) pp. 193–212.
- [27] S. Arunachalam, V. Havlicek, G. Nannicini, K. Temme, and P. Wocjan, Quantum **6**, 789 (2022).
- [28] M. Steudtner, S. Morley-Short, W. Pol, S. Sim, C. L. Cortes, M. Loipersberger, R. M. Parrish, M. Degroote, N. Moll, R. Santagati, and M. Streif, arXiv (2023), 10.48550/arxiv.2303.14118, 2303.14118.
- [29] A. Luis and J. Perina, Physical Review A **54**, 4564 (1996).
- [30] W. v. Dam, G. M. D'Ariano, A. Ekert, C. Macchiavello, and M. Mosca, Physical Review Letters **98**, 090501 (2007), quant-ph/0609160.
- [31] P. Najafi, P. C. S. Costa, and D. W. Berry, arXiv (2023), 10.48550/arxiv.2303.12503, 2303.12503.
- [32] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 339 (1998), quant-ph/9708016.

- [33] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).
- [34] D. W. Berry, Y. Su, C. Gyurik, R. King, J. Basso, A. D. T. Barba, A. Rajput, N. Wiebe, V. Dunjko, and R. Babbush, arXiv (2022), 10.48550/arxiv.2209.13581, 2209.13581.
- [35] F. Harris, Proceedings of the IEEE **66**, 51 (1978).
- [36] Y. Dong, X. Meng, K. B. Whaley, and L. Lin, Phys. Rev. A **103**, 042419 (2021).
- [37] J. Haah, Quantum **3**, 190 (2019), 1806.10236.
- [38] R. Chao, D. Ding, A. Gilyen, C. Huang, and M. Szegedy, arXiv (2020), 10.48550/arxiv.2003.02831, 2003.02831.
- [39] L. Ying, Quantum **6**, 842 (2022), 2202.02671.
- [40] Y. Dong, J. Wang, X. Meng, H. Ni, and L. Lin, “QSP Phase Factor Solvers,” <https://github.com/qspack/QSPACK> (2013).
- [41] Y. Wang, L. Zhang, Z. Yu, and X. Wang, Physical Review A **108**, 062413 (2023), 2209.14278.
- [42] D. Motlagh and N. Wiebe, “Generalized quantum signal processing,” (2023), arXiv:2308.01501 [quant-ph].
- [43] P. J. J. O’Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, P. V. Coveney, P. J. Love, H. Neven, A. Aspuru-Guzik, and J. M. Martinis, Phys. Rev. X **6**, 031007 (2016).
- [44] H. Apel, J. Lemieux, and M. Steudner, In Preparation (2024).
- [45] G. H. Low and I. L. Chuang, Quantum **3**, 163 (2019).
- [46] G. H. Low and I. L. Chuang, Phys. Rev. Lett. **118**, 010501 (2017).
- [47] S. McArdle, A. Gilyen, and M. Berta, arXiv (2022), 10.48550/arxiv.2210.14892, 2210.14892.
- [48] G. H. Low, V. Kliuchnikov, and L. Schaeffer, arXiv (2018), 10.48550/arxiv.1812.00954, 1812.00954.
- [49] Y. R. Sanders, D. W. Berry, P. C. S. Costa, L. W. Tessler, N. Wiebe, C. Gidney, H. Neven, and R. Babbush, PRX Quantum **1** (2020), 10.1103/prxquantum.1.020312, 2007.07391.
- [50] C. Gidney, Quantum **2**, 74 (2018), 1709.06648.

Appendix A: Circuit simulation details

1. QSFT QPE

In Section III, we present the results of state vector simulations of the different QPE routines, including an implementation of the QSFT QPE routine formulated in Ref. [18]. In this implementation, we identified four small modifications to the circuit presented in Fig. 12 of Ref. [18] that were necessary to obtain valid results. In this section, we detail the specific circuit we used and explicitly note these modifications. The circuit used to implement the QSFT QPE routine is shown in Fig. 7. The main differences from the circuit in Fig. 12 of Ref. [18] are:

1. In our circuit we use qubitization to obtain the unitary for phase estimation.
2. The controlled phase rotations to remove the previously measured phase bits are applied using the opposite bit ordering in our circuit compared with Fig. 12 of Ref. [18] – here, the $\pi/2$ rotation is applied on the most significant bit measured, the $\pi/4$ rotation on the second most significant bit etc., while in Ref. [18] the controls were ordered in the opposite way.
3. The degree of the polynomial in QSFT is (at most) equal to the sum of the number of applications of U_A and U_A^\dagger , so to realize a d -degree polynomial, we only need to repeat the circuit in the orange box $d/2$ times rather than d times in Fig. 12 of Ref. [18].
4. The polynomial we use (Eq. (4)) differs by a sign from that given in Eq. (69) of Ref. [18].

The circuit was numerically verified using small QPE instances including the two-qubit Hamiltonian for molecular hydrogen over various bond lengths considered in [43].

2. QPE with window functions

For simulating QPE using a window function, we show the circuit for preparing the cosine window state [19] in Figure 8. As the cosine window has an exact construction and is not parametrized, no further considerations were needed to implement the simulations. The rectangular window function was even simpler to implement, consisting only of Hadamard gates on the phase register.

Number of additional phase qubits	Best α found
0	0
1	6
2	13
3	25
4	51
5	100*
6	100*

Table II. Table of Kaiser window α values obtained through optimization of the QPE success probability [44]. The asterisks next to the values for 5 and 6 additional phase qubits indicate that these values are only approximate guesses rather than fully optimized values – the optimization for these simulations was unstable since the success probabilities were high enough to be limited by floating point precision errors.

In order to simulate QPE using the Kaiser window, there are two considerations that must be taken into account. Firstly, a choice of the parameter α must be made in order to obtain a concrete definition for a Kaiser window to prepare. Secondly, we have to decide on the specific implementation details for the state preparation. For the choice of α , we find that the more additional phase qubits are added and then thrown away in the QPE routine, the higher the value of α that can be chosen. Additionally, increasing α leads to higher success probabilities up to a certain threshold, after which the success probabilities fall off dramatically. From a separate analysis and optimization of the Kaiser window [44], we report best found values of α for each additional number of phase qubits but make no claim about the optimality of these values. We verified these numbers using a numerical optimization of α and obtained qualitatively similar results. It is likely that different values of α are optimal for different applications, and the investigation of this could be a worthwhile route for future work.

For implementing the Kaiser window in our state vector simulation, we simply injected the (normalized) Kaiser window state into the phase register of the QPE simulation. While this ignores the cost of the window state, it correctly simulates the impact of applying the window state to the QPE algorithm. We leave the concrete cost analysis of the Kaiser window state to future work.

Appendix B: Additional phase qubits in QPE using the Kaiser window state

In this section, we sketch out the proof for estimating the number of additional phase qubits needed to obtain accurate eigenphases in QPE using Kaiser windows by extending the procedure in [32]. Our sketch is based on Appendix C of [34], and we strongly recommend that readers review this reference.

In the standard procedure from [32], to derive the number of additional qubits, one first considers the expression for the amplitude of the state $|(a - t) \bmod 2^n\rangle$, where a is the closest n -bit integer approximation to some phase ϕ , and $-2^{n-1} \leq t < 2^{n-1}$. Here, we treat t as a dummy index used to compute the deviation from the nearest integer a . Eventually, t will be substituted to compute the total probability of being at least some integer k far away from a . We refer to the amplitude of state $|(a - t) \bmod 2^n\rangle$ as α_t (not to be confused with the Kaiser window parameter α). For simpler windows like the rectangular window, the procedure for computing the number of additional phase bits continues by simplifying and bounding α_t before summing the squared amplitudes corresponding to states that are within some distance $k/2^n$ away from a for integer k . This is equivalent to computing the tails of this probability distribution (in terms of k) and subtracting from unit probability. We set this expression equal to some target probability $1 - \delta$ then solve for k . We now have an expression for k in terms of δ but would like to express the number of additional phase qubits in terms of δ . For this, we relate the two quantities, k and the number of additional phase qubits: we set the distance $k/2^n$ equal to a target precision $\frac{1}{2^{m+1}}$, where $m \leq n$. We define $p = n - m$ as the number of additional phase qubits and solve for p using the expression for k in terms of δ and noting that $k = 2^{n-m-1} = 2^{p-1}$.

For the Kaiser window, writing an analytical expression for α_t is less trivial. In [34], the authors model the probability distribution by applying the Laplace approximation method and estimate the normalization constant of the window state. This method approximates the probability distribution using a normal distribution given that the distribution is well-behaved (i.e. symmetric and unimodal). Following the procedure in [34], using the approximated normalization constant and noting that the first zero or sidelobe of the distribution occurs at $(\pi/(2^{n-1}))\sqrt{1 + \alpha^2}$ (for the Kaiser window parameter α), the authors integrate the probabilities over the tails of the distribution and set this total probability (of failure) to some δ . This allows one to solve for the Kaiser window parameter α :

$$\alpha = (1/2\pi) \ln(1/\delta) + \mathcal{O}(\ln \ln(1/\delta)). \quad (\text{B1})$$

With an expression for α , we now set the confidence interval $(\pi/(2^{n-1}))\sqrt{1+\alpha^2}$ equal to some target precision $\frac{1}{2^{m+1}}$ as was done above for the rectangular window function, and we again solve for $p = n - m$:

$$\text{target precision} = \frac{\pi}{2^{n-1}} \sqrt{1+\alpha^2} \quad (\text{B2})$$

$$\frac{1}{2^{m+1}} = \frac{\pi}{2^{n-1}} \sqrt{1+\alpha^2} \quad (\text{B3})$$

$$2^{n-1-m-1} = \pi \sqrt{1+\alpha^2} \quad (\text{B4})$$

$$2^{p-2} = \pi \sqrt{1+\alpha^2} \quad (\text{B5})$$

$$2^{p-2} \approx \pi((1/2\pi) \ln(1/\delta) + \mathcal{O}(\ln \ln(1/\delta))) \quad (\text{B6})$$

$$p = \mathcal{O}(\log \log 1/\delta). \quad (\text{B7})$$

In [34], they note that the $\ln \ln$ term in B1 is larger than the error for approximating $\sqrt{1+\alpha^2}$ using α , justifying the approximation in Eq. B6. In the end, we note that $p = \mathcal{O}(\log \log 1/\delta)$.

While we did not use this derived value to choose the number of additional qubits in our numerical simulations, we numerically verified that the Kaiser window does in fact use fewer additional phase qubits than the cosine window to achieve comparable success probabilities (beyond four qubits). Alternatively, assuming the same number of additional qubits, using the Kaiser window would correspond to a higher success probability than that obtained using the cosine window, beyond some crossover point.

Appendix C: Quantum singular value transformation (QSVD)

In this appendix, a brief overview of the quantum singular value transform subroutine is given. Many quantum algorithms can be defined in terms of functions of matrices. For example, the time evolution operator may be thought of as applying the function $f(x) := \exp(ixt)$ onto a Hamiltonian H . A general method for generating matrix functions on a quantum computer is given by the *quantum singular value transform* (QSVD) [16]. QSVD uses two primitive operations: controlled rotation gates and a block encoding oracle [45], which takes a target matrix A and realizes a unitary block encoding operator

$$U_A := \begin{bmatrix} A & * \\ * & * \end{bmatrix}, \quad (\text{C1})$$

where the terms labelled $*$ are left undefined and can take on any values. The key insight underpinning the quantum singular value transform is that the block encoding operator may be written as a direct sum of projectors into a set of invariant subspaces of the form $\{|0^m\rangle|\lambda_i\rangle, |\perp_i\rangle\}$, where m is the number of ancilla qubits used to encode the matrix A , $|\lambda_i\rangle$ is an eigenstate of A and $|\perp_i\rangle$ is orthogonal to $|0^m\rangle|\nu_i\rangle$. Since the full operator U_A is defined in terms of a direct sum over all eigenstates of A , the result of applying the block encoding oracle onto *any* state of the form $|0^m\rangle|\psi\rangle$ can be understood in terms of 2×2 matrices whose elements correspond to the projectors into this invariant subspace.

This insight may be combined with the notion of *quantum signal processing* [46], a technique in which single qubit rotation gates are interspersed between applications of a single qubit unitary matrix in order to transform the output of that matrix as

$$f(U) = Rz(\phi_0) \prod_{j=1}^d URz(\phi_j), \quad (\text{C2})$$

where d is the number of applications of the unitary matrix which upper bounds the degree of the matrix polynomial that can be realized. By choosing an appropriate set of rotation angles $\{\phi_i\}$, different polynomial functions of the original matrix can be implemented. The technique can be extended to arbitrary matrix transformations by defining a rotation operator Rz_Π whose eigenstates are the projectors into the invariant subspaces defined above. In practice,

this is accomplished by using an Rz gate sandwiched between NOT gates controlled on the all-zero sector of the block encoding ancillae. The resulting gate sequence is given as

$$Rz_{\Pi}(\phi_0) \prod_{j=1}^{d/2} U_A Rz_{\Pi}(\phi_j) U_A^\dagger Rz_{\Pi}(\phi_{j+1}) . \quad (\text{C3})$$

For Hermitian matrices A , the result of applying this sequence of operators is a block encoding of a matrix function defined over the eigenvalues of A ,

$$f(A) := \sum_{k=0}^{\dim(A)-1} f(\lambda_k) |\lambda_k\rangle \langle \lambda_k| . \quad (\text{C4})$$

When A is not Hermitian, the sequence of gates is altered slightly such that we alternate between applications of U_A and U_A^\dagger , with the result being a block encoding of a function over the *singular values* of A instead:

$$f(A) := \sum_{k=0}^{\dim(A)-1} f(\sigma_k) |v_k\rangle \langle w_k| , \quad (\text{C5})$$

where $A = \sum_{k=0}^{\dim(A)-1} \sigma_k |v_k\rangle \langle w_k|$ is the singular value decomposition of A . In both cases, the protocol is successful if the block encoding ancillae are measured to be in the all-zero state. For the remainder of this work, we will focus on QSVD.

There are several constraints on the classes of polynomial functions that can be realized via QSVD, most notably:

1. The degree of the polynomial must be less than or equal to d , the number of applications of U_A .
2. The parity of the polynomial must match the parity of d .
3. For all x such that $|x| \leq 1$, $|f(x)| \leq 1$.

There are other conventions for QSVD that can be used, but these differ by basis changes from the convention chosen here. A recent work [42] discovered that QSP (and therefore QSVD) can be generalized to relax the first two of these constraints; since the target function considered in this work naturally accommodates these constraints, we have not included a discussion here and only note the new work for completeness.

Appendix D: Choice of QSVD function in QSVD QPE

We outline the choice of QSVD function used in this work (taken from Ref. [18]). As mentioned in the main text, the QSVD QPE routine builds upon iterative QPE, meaning that the results of previous QPE iterations are rotated out of the eigenphase being measured. We then seek some transformation of the singular values of the resulting unitary such that at each step k of the QPE routine, if the k th bit of the ideal eigenphase (after rounding to account for the bit truncation to m bits) is 0, we kick back a phase of 1 to the phase register and if the k th bit is 1, we kick back a phase of -1 . Intuitively, it is clear that this function should be a sign function, but the specific choice of sign function requires a more careful construction of a particular block encoding whose singular values perform the desired kickback.

At step k of the QPE protocol a unitary block encoding of the matrix $A_k(\phi)$ is implemented using a Hadamard test on the target unitary with the previously measured eigenphases rotated out:

$$A_k(\phi) := \frac{1}{2} (\mathbb{I} - \exp(-2\pi i 0.0\varphi_{k-1}\varphi_{k-2}\dots) \exp(2\pi i 2^{m-k}\phi)) , \quad (\text{D1})$$

where φ_j are the j th-bit approximate values of ϕ obtained from previous iterations. This unitary $A_k(\phi)$ is a block encoding that encodes the unitary of interest U , for which we want to obtain an estimate of the eigenvalue ϕ . $A_k(\phi)$ has singular values

$$\sigma_k = \frac{1}{2} |1 - \exp(-2\pi i 0.0\varphi_{k-1}\varphi_{k-2}\dots) \exp(2\pi i 2^k\phi)| \quad (\text{D2})$$

$$= \frac{1}{2} |\cos(\pi 2^{m-k}\phi - 0.0\varphi_{k-1}\varphi_{k-2}\dots)| . \quad (\text{D3})$$

Let us consider the case where ϕ can be exactly represented using m bits. In this case, at the zeroth iteration of the QPE protocol, the singular value $\sigma_0 = |\cos(\pi 0.\phi_m)|$ is 1 if $\phi_m = 0$ and 0 if $\phi_m = 1$. The value ϕ_m can then be deterministically loaded into the zeroth bit of the phase register by implementing $1 - \sigma_0$ controlled on the phase qubit using a Hadamard test.

In general, however, ϕ is not exactly expressible using m bits and so this protocol will not be deterministic. By modifying the transforming function, the protocol can be adapted to be deterministic for any phase: note that the most ambiguous value of ϕ that can occur is $0.\phi_m 011\dots 1 < 0.\phi_m 1$. That is, the true phase lies precisely between our desired bin and another bin. We therefore want to realize a function $f(x)$ such that

$$f(x) = \begin{cases} 0 & \text{if } x > \frac{1}{\sqrt{2}} \\ 1 & \text{if } x < \frac{1}{\sqrt{2}} \end{cases}, \quad (\text{D4})$$

where the inputs x to the function for QSVT are the singular values of the matrix that we are seeking to transform. This comes from the fact that the ambiguity has a maximum rounding error in the eigenphase of $1/4$ and $\cos(\pi/4) = 1/\sqrt{2}$. The function that has this property is the shifted sign function

$$f(x) = \Theta\left(\frac{1}{\sqrt{2}} - x\right), \quad (\text{D5})$$

with $\Theta(x)$ denoting the sign function, using the notation of Ref. [18] (not to be confused with the Heaviside step function). We can use QSVT to obtain a polynomial approximation of this function, which will allow for the value of the m th bit to be loaded into the phase register. At each subsequent step, we first rotate out the phase at the bits we have previously calculated using controlled rotations before repeating the QSVT to load the next bit into the phase register. This allows us to reduce the ambiguity when we cannot represent the true eigenphase with m bits, and thereby increase the success probability of QPE.

Appendix E: Back-of-the-envelope gate cost for preparing the window states

To fairly compare the resource costs of the QSVT QPE and the standard QPE using additional phase qubits, the cost of the initial window state preparation must be taken into account. As reported in Fig. 4, the number of queries to the block encoding in QPE is far larger in the QSVT version of QPE than in the version using additional phase qubits. So if we are to determine which version has a lower complexity in practice, we effectively need to determine how expensive preparing window states is in comparison to implementing block encodings and calling them many times for some interesting representative instance size. As a representative instance size, we choose to look at the cost of computing the ground state for the molecule FeMoco, a common benchmark in the resource estimation literature [11, 13, 15] as it is widely believed to be beyond the reach of classical computation due to its strongly-correlated properties, and for which many detailed resource costings exist.

To estimate the gate complexity of these state preparations, we must assume a particular cost model. Because QPE is a large depth circuit, we assume the cost model of a fault-tolerant quantum computer, where the cost of executing non-Clifford gates far exceeds that of Clifford gates. Thus, we will determine the non-Clifford gate complexity; in particular, the Toffoli complexity.

As of the writing of this manuscript, the lowest quoted Toffoli complexity in the literature for block encoding the Hamiltonian for the FeMoco molecule is 16923 Toffolis ($\mathcal{O}(10^5)$), and the total cost of estimating a ground state energy to chemical accuracy is 3.2×10^{10} Toffolis [15]. The rectangular window can be prepared using only Hadamards, and so, it is effectively free. The cost of the cosine window is $\tilde{\mathcal{O}}(m)$ for m phase qubits [13]. For the FeMoco Hamiltonian, $m = 20$ [15]. Assuming we are using an additional $p = 5$ qubits, the total cost for this preparation scales linearly with 25 qubits. This cost is negligible in comparison to the cost of the block encoding.

As far as we know, there are no quoted asymptotic or numeric complexities for the cost of preparing the Kaiser window. Thus, here we present back-of-the-envelope costs for two methods to prepare this state: arbitrary state preparation and state preparation via Quantum Signal Processing (QSP) [47].

We can roughly estimate the Toffoli complexity for performing arbitrary state preparation, where we classically pre-compute the amplitudes given in Table I and coherently load them onto the quantum computer. A variety of methods exist in the literature, the most performant of which scale as $\tilde{\mathcal{O}}(N + \log N)$ for preparing N amplitudes, including a technique called “alias sampling” (introduced in [13]) and another method we refer to as “LKS” (for Low, Kliuchnikov, and Schaeffer) which uses a cascade of data-loaders and adders (introduced in [48]). Because we do not exactly uncompute the window state preparation at the end of QPE, we cannot employ methods that entangle the prepared state with garbage, ancillary qubits. This rules out alias sampling, and so we analyze the cost of LKS state

preparation. LKS requires a bits of precision parameter for how accurately to represent each prepared amplitude. We take a conservative estimate of machine precision, 32 bits. For a QPE phase register of 25 qubits, using 32 bits of precision for each prepared amplitude in the Kaiser window state, LKS gives a Toffoli complexity of order $\mathcal{O}(10^6)$. Though it is an order of magnitude more expensive than the the block encoding of FeMoco, this state is prepared exactly once at the beginning of the QPE, and thus, is a sub-leading cost in comparison to the total 3.2×10^{10} Toffoli complexity from the roughly $\mathcal{O}(10^6)$ queries to the block encoding iterate.

Alternatively, one could prepare the Kaiser window state using QSP methods, as detailed in [47]. There, we must make $\mathcal{O}(\alpha + \ln(\Delta^{-1}))$ many queries to the circuit shown in Fig. 1b. in [47], which in turn makes queries to the circuit shown in Fig. 1a. in the same reference, where α is the window parameter value for the Kaiser window, and Δ is the error in approximation. For comparison with LKS, we can take $\ln(\Delta^{-1})$ to be equal to 32. For a conservative estimate, we take a high value of $\alpha = 100$. The circuit depicted in Fig. 1a in [47] can be prepared using addition with a phase gradient state [49, 50]. Addition over n qubits has cost $n - 1$ [50], so this costs 24 Toffolis for our instance size (the total number of phase qubits). The circuit in Fig. 1b. makes 3 queries to this circuit, for a total of 72 Toffolis. Now, we can multiply this count by the number of total queries given above in terms of α and $\ln(\Delta^{-1})$, which we take to be 100 and 32 (respectively). This gives us a total of $\mathcal{O}(10^4)$ Toffolis, an order of magnitude fewer Toffolis than the block encoding of the FeMoco Hamiltonian.

We have not performed a detailed, explicit compilation of preparing the Kaiser window state. Nor have we spent any time looking into alternative state preparation methods as this is not the focus of this work. Even still, the methods quoted above yield counts that are low enough to be negligible in comparison to the total cost of querying block encodings of Hamiltonians for interesting systems, justifying the comparison of QSVD QPE and textbook QPE with additional phase qubits to be done only in terms of query complexity.

Appendix F: Asymmetry of the QSVD QPE Success Probability

The success probability of QPE arising from bit discretization error is periodic with period $1/2^n$ for $n = m + p$ total bits of precision. While this is true in theory for all of the QPE routines explored here, in practice QSVD QPE is *not* periodic. The reason for this lies in the choice of polynomial decomposition used for the target function. The ideal shifted sign function (Eq. (4) with $\Delta = \kappa = 0$) is symmetric with respect to the different bit values 0 and 1, meaning that QSVD QPE using this ideal function should be periodic. However, it is not possible to use the ideal shifted sign function in a real subroutine, since that would necessitate an infinite number of terms – the *imperfect* shifted sign function resulting from the finite order polynomial decomposition is *not* symmetric with respect to these bit values. Rather, as shown in Fig. 9, the deviation from the ideal values for the 1 bits is higher than the corresponding deviation for the zero bits.

As a result of this asymmetry, the error in the QSVD QPE routine is dependent on the number of 1 bits in the ideal eigenphase, which breaks the periodicity of the results. Fig. 10 shows this effect for statevector simulations of QSVD QPE with 5 phase qubits using a 64 degree polynomial approximation of the shifted sign function. Overlaid with the statevector simulation data (dark blue line) are the number of 1 bits in the closest 5-bit approximation to each ideal eigenphase (red points). The results of the statevector simulation match the number of 1 bits to a high degree, corroborating the explanation for the asymmetry of QSVD QPE.

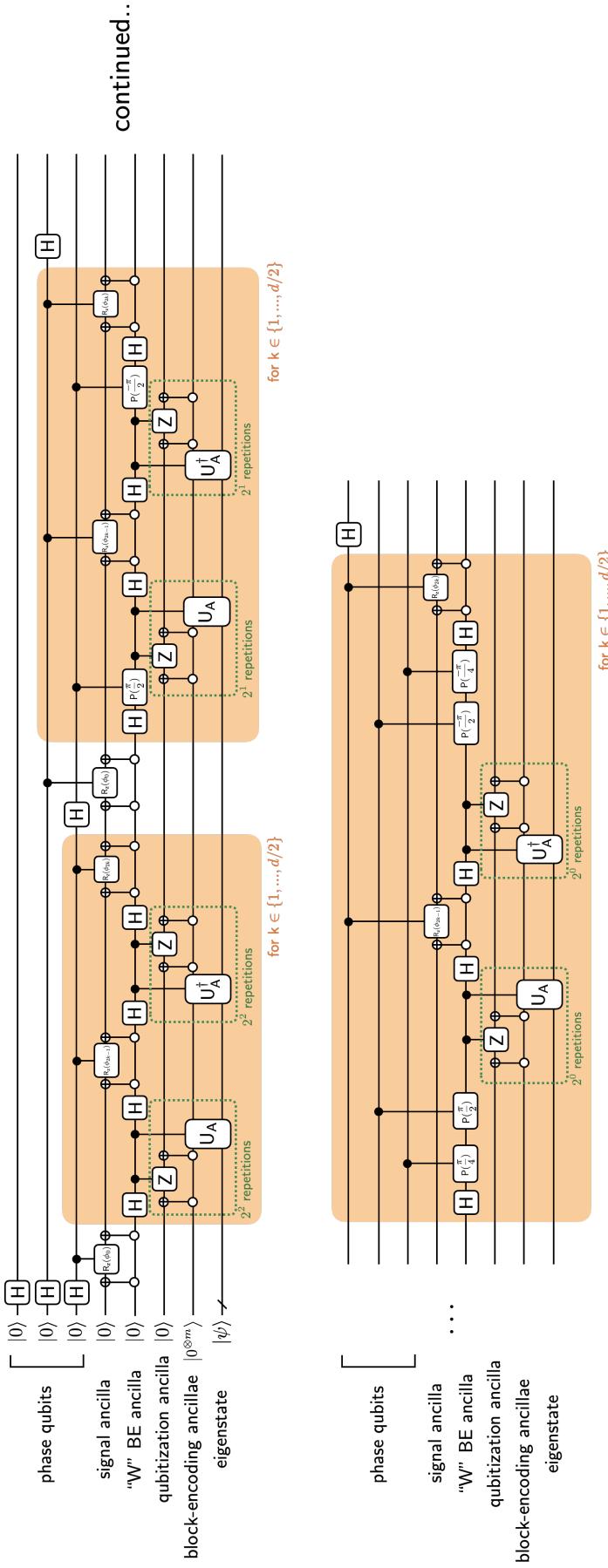


Figure 7. Numerically verified QSVT QPE circuit that obtains the eigenphase of a qubitization iterate as $\exp(i \arccos(\lambda))$, with λ the eigenvalue of interest for the block-encoded matrix A . U_A denotes a block encoding of A , which is combined with a reflection about the all-zero sector of the block encoding ancilla subspace to form the qubitization iterate. This in turn is controlled on an additional ancilla sandwiched by Hadamard gates and (depending on the iteration of the QPE protocol) a set of rotation gates, with the latter used to counterrotate the phase bits measured in previous iterations of the QPE routine. This construction results in the block encoding of a new matrix $W_j(\theta) := \frac{1}{2}(I + \exp(-2\pi i \theta)(Z \Pi U_A)^{2^j})$ using a single block encoding ancilla (labelled "W" BE ancilla in the circuit). Finally, the R_z gates that implement the QSVT phase rotations are applied using projectors into the zero subspace of the "W" block encoding – these rotations are controlled on the QPE phase register in order to load the estimate of the eigenphase into it. Since we are implementing QSVT, the protocol alternates between applications of W and applications of W^\dagger . The units within the orange boxes correspond to the repetitions necessary to realize the QSVT polynomial – each of the $d+1$ phases applied within each box (plus the single rotation before each box) could, in principle, be unique (they are obtained via optimization for the desired target polynomial), but the set of phases for each box are the same. For this specific circuit, 3 phase bits are used, but the simulations and verifications were performed with various numbers of phase qubits up to 11.

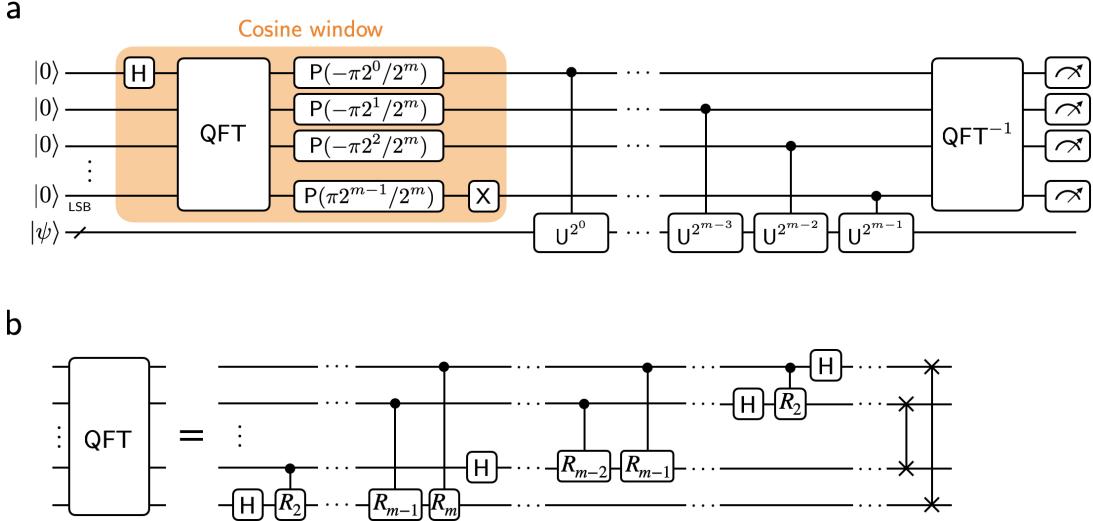


Figure 8. (a) Circuit for QPE using cosine window [19]. The lowermost qubit in the phase register is the least significant bit (LSB). Gate P is the phase gate, defined in Eq. 6. (b) Circuit for QFT. Note that in Ref. [19], this operation is denoted as QFT^{-1} .

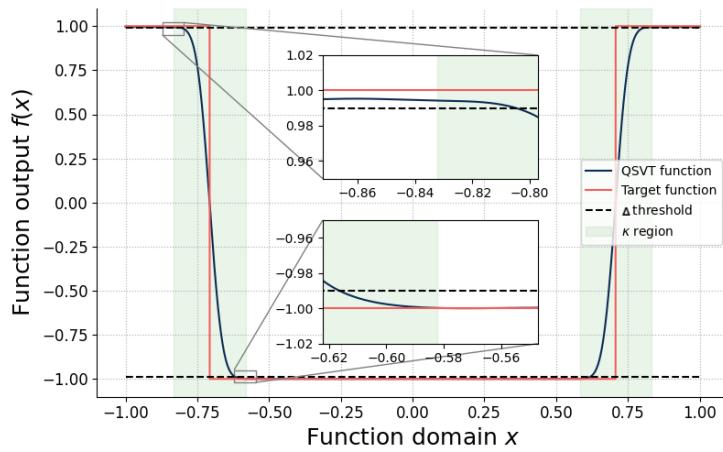


Figure 9. Plot of the target shifted sign function for QSVD QPE, showing both the approximate target function Eq. (4) (red line) and the output from the function resulting from the optimized QSVD phases (dark blue line), with the target κ region and Δ values shown as green shading and black dashed lines respectively. The zoomed insets show that the optimized function approximates the ideal function much more closely for the -1 branches of the target function than for the $+1$, resulting in an asymmetry in the output of QSVD QPE.

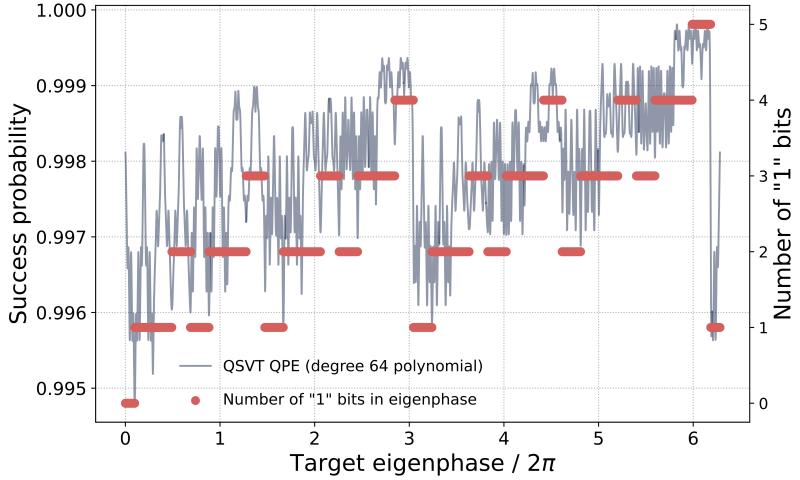


Figure 10. Plot of QSVT QPE success probability as a function of target eigenphase, with the number of 1 bits in the binary fixed point representation of that eigenphase shown as green points overlaid on the QSVT QPE data (red line). The asymmetry of the QSVT QPE success probability is well explained by the number of 1 bits, which is explained by the asymmetry in the target function as shown in Fig. 9.