Quantum state preparation via piecewise QSVT

Oliver O'Brien[1,2] and Christoph Sünderhauf[1]

[1]Riverlane, Cambridge, United Kingdom
[2]Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Wilberforce Road, Cambridge, CB3 0WA, United Kingdom
May 2025

Efficient state preparation is essential for implementing efficient quantum algorithms. Whilst several techniques for low-cost state preparation exist, this work facilitates further classes of states, whose amplitudes are well approximated by piecewise polynomials. We show how such states can be efficiently prepared using a piecewise Quantum Singular Value Transformation along with a new piecewise linear diagonal block encoding. We illustrate this with the explicit examples of $x^\alpha |x\rangle$ and $\log x |x\rangle$. Further, our technique reduces the cost of window boosted Quantum Phase Estimation by efficiently preparing the B-spline window state. We demonstrate this window state requires 50 times fewer Toffolis to prepare than the state-of-the-art Kaiser window state, and we show that the B-spline window replicates the Kaiser window's exponential reduction in tail probability for QPE.

Contents

Oliver O'Brien: opo20@cam.ac.uk
Christoph Sünderhauf: christoph.sunderhauf@riverlane.com

1 Introduction

State preparation is a major bottleneck for numerous quantum algorithms with applications in many fields including quantum chemistry [1–4], quantum machine learning [5], quantum finance [6–9], and quantum computational fluid dynamics [10]. The exponentially large state space afforded by a quantum computer that lends so much power when performing computations becomes a hindrance when we must initialise a state. In order to specify a **generic** n-qubit state **exactly** one must load 2^n amplitudes into our quantum computer, and hence any algorithms with a sub-exponential cost in n become subdominant to the cost of state preparation.

Luckily slight relaxations of these requirements provide us with efficient avenues for state preparation. For example, considering the preparation of only a subset of possible states or by attempting to prepare states approximately through some form of lossy compression. Multiple techniques utilising these approaches have been developed including: preparing Matrix Product States (MPS) [11–17], sparse states [18], truncated series expansions [19–21], training variational circuits [22–24], compressing repeated values [25, 26], preparing states via coherent arithmetic [27, 28], and QSVT polynomial approximation [29, 30]. In this work we extend the QSVT polynomial approximation techniques to allow for piecewise QSVT polynomial approximations.

QSVT (Quantum Singular Value Transformation) [31] is a quantum algorithm that allows one to apply arbitrary polynomial transformations to block-encoded matrices. The flexibility of this algorithm allows it to approximate many other quantum processes, and hence has found numerous applications including amplitude amplification [31], Hamiltonian simulation [31], phase estimation [32], and linear system solvers [31].

Previous work utilised QSVT for state preparation

to apply a global polynomial that approximates the amplitudes of a desired state, avoiding the need for coherent arithmetic and hence significantly reducing both the ancilla and Toffoli cost. Applications were demonstrated for a number of useful states including the Gaussian state and Kaiser window state [29]. However, this technique falls down when trying to simulate functions with sharp discontinuities or non-differentiable points. We navigate around these issues by utilising piecewise polynomial approximations which simply deal with discontinuities by matching them to the boundaries of segments and cope with variability in approximation difficulty between regions by varying the length of segments used. As such, our techniques fair well where previous works failed, in particular we can efficiently load $\sqrt{x}$ which [29] explicitly noted was not possible with a single polynomial. Furthermore, we can load the B-spline window state with orders of magnitude lower cost than the Kaiser window state and we prove that this gives a similarly exponential performance boost to QPE.

In Section 2 we describe the technique for performing piecewise QSVT. First we present a new piecewise exact linear block encoding. Then, we demonstrate how to construct a QSVT circuit that implements piecewise polynomial transformations. We go on to discuss how to convert the transformed block encoding into a state and the resource costs of our quantum algorithm. In Section 3 we explore various applications of our algorithm to loading states of interest exponentially faster than a naive approach.

2 Piecewise QSVT

The aim of piecewise QSVT is to load a state with amplitudes described by different polynomials for different segments of computational basis elements:

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} p_{s_x}\left(1-2\,\frac{x \bmod L_{s_x}}{L_{s_x}}\right)|x\rangle \qquad (1)$$

where $s_x \in \mathbb{Z}_S$ indexes which of the S polynomial segments x lies in, L_{s_x} denotes the length of that segment, and p_{s_x} is the polynomial transformation applied to that segment.

Piecewise QSVT uses a modified QSVT circuit (Section 2.2) to implement different polynomial transformations on different segments of a diagonal block encoded matrix. The diagonal (singular) values of the transformed block encoding are effectively samples from the polynomials taken for x-values corresponding to the original diagonal values. Hence, by knowing the values of the original diagonal block encoding: $1-2\,\frac{x \bmod L_{s_x}}{L_{s_x}}$ (Section 2.1), and controlling the polynomials implemented: p_{s_x} (Section 2.2), it is possible to produce a diagonal block encoding with values we desire: $p_{s_x}\left(1-2\,\frac{x \bmod L_{s_x}}{L_{s_x}}\right)$. By acting on a uniform superposition and performing amplitude amplification (Section 2.3), this diagonal block encoding can be easily converted into the desired state (1).

There are some conditions upon this process:

(1) $|p_i(t)| \leq 1$ for all $t \in [-1,1]$ and $i \in \mathbb{Z}_S$

(2) L_i must be a power of 2 ($L_i = 2^{l_i}$ for $l_i \in \mathbb{N}_0$)

(3) The i-th segment must span $x \in [aL_i, aL_i + (L_i - 1)]$ for some $a \in \mathbb{N}_0$

Condition (1) is inherited from QSVT, whereas conditions (2) and (3) are necessary for the pieces to be described by ignoring bits of x which is key to allowing an efficient piecewise algorithm.

2.1 Piecewise exact linear block encoding

The singular values of our diagonal block encoding will become the x values we wish to fit our polynomials at. As such it is desirable that our piecewise polynomials are each applied to singular values that span the whole range of possible values from 1 to -1. This makes it easier to ensure that the fitted polynomials satisfy condition (1) without rescaling. It also enables us to outperform the coherent inequality test based piecewise polynomial technique suggested in [29] which uses polynomials over disjoint domains between 0 and 1. In this case the polynomials may grow large outside the domains they approximate and hence could require large rescaling to satisfy condition (1) on the whole domain $[-1,1]$.

To achieve this we define a piecewise linear diagonal block encoding U such that the x-th entry is given by $1-2\,\frac{x \bmod L_{s_x}}{L_{s_x}}$ where L_{s_x} is the length of the segment containing x:

$$\left(\begin{array}{cccccccc|c}
1 & & & & & & & & \\
& 1-\frac{2}{L_0} & & & & & & & \\
& & \ddots & & & & & & \\
& & & -1+\frac{2}{L_0} & & & & & \\
& & & & \ddots & & & & \\
& & & & & 1 & & & \\
& & & & & & 1-\frac{2}{L_{S-1}} & & \\
& & & & & & & \ddots & \\
& & & & & & & & -1+\frac{2}{L_{S-1}} \\
\hline
\multicolumn{8}{c|}{j_2} & j_3
\end{array}\right)$$

In this block encoding our desired matrix occupies the top-left block which corresponds to the flag qubits mapping from $|\mathbf{0}\rangle$ to $|\mathbf{0}\rangle$. All other initial or final values for the flag qubits correspond to the junk blocks j_1, j_2 or j_3.

Fig. 1 shows the circuit that efficiently performs a controlled version of such a block encoding. The "compute carry" computes the carry bit from adding the first l bits of x ($x_l = x \bmod 2^l$) and k ($k_l = k \bmod 2^l$) where the value of l is given by l_{s_x} of the current segment. Fig. 2 demonstrates how this can

be performed for the example $l_{max} = 4$ using variable unary iteration and a quantum adder. First the carry bits are computed for all l via a slightly modified version of the adder circuit for adding classical bits to quantum registers described in [33] (and illustrated in Fig. 3); we have removed the final addition layer and made the uncompute symmetrical to ensure the block encoding is Hermitian. Then variable unary iteration [33] controls a copy from the carry bits into the flag register, so the correct carry bit is copied depending on the size of the segment l_{s_x}.

Variable unary iteration is essential to the efficiency of our algorithm, as it only requires $S-2$ compute/uncompute Toffoli pairs to produce S different segments, compared to the $2^n - 2$ pairs required by standard unary iteration. This is accomplished by simply ignoring some of the least significant bits during the unary iteration, hence the segments must all be a power of 2 in length and begin at an integer divisible by that power of 2 (Conditions (2) and (3)).

It is simple to prove that Fig. 1 provides a block encoding of the desired matrix:

$$
\begin{aligned}
&(\langle 0| \otimes \langle 0|^{\otimes l_{max}} \otimes \langle x|) U (|0\rangle \otimes |0\rangle^{\otimes l_{max}} \otimes |x\rangle) \\
&= \sum_{a,b=0}^{1} \sum_{k,k'=0}^{L_{max}-1} \frac{(-1)^b \langle b| \langle k'|}{\sqrt{2L_{max}}} \\
&\qquad \frac{(-1)^a |a \oplus ((k_{l_{s_x}} + x_{l_{s_x}}) \geq 2^{l_{s_x}}))\rangle |k\rangle}{\sqrt{2L_{max}}} \\
&= \frac{1}{2L_{max}} \sum_{a=0}^{1} \sum_{k=0}^{L_{max}-1} (-1)^{a \oplus ((k_{l_{s_x}} + x_{l_{s_x}}) \geq 2^{l_{s_x}}))} (-1)^a \\
&= \frac{1}{L_{max}} \sum_{k=0}^{L_{max}-1} (-1)^{(k_{l_{s_x}} + x_{l_{s_x}}) \geq 2^{l_{s_x}}} \\
&= \frac{1}{2^{l_{s_x}}} \sum_{k=0}^{2^{l_{s_x}}-1} (-1)^{(k + x_{l_{s_x}}) \geq 2^{l_{s_x}}} \\
&= \sum_{k=0}^{2^{l_{s_x}} - x_{l_{s_x}} - 1} \frac{1}{2^{l_{s_x}}} - \sum_{k=2^{l_{s_x}} - x_{l_{s_x}}}^{2^{l_{s_x}}-1} \frac{1}{2^{l_{s_x}}} = \frac{2^{l_{s_x}} - 2x_{l_{s_x}}}{2^{l_{s_x}}} \\
&= 1 - 2\,\frac{x \bmod 2^{l_{s_x}}}{2^{l_{s_x}}}
\end{aligned}
$$

The adder costs l_{max} compute/uncompute Toffoli pairs, the controlled variable unary iteration costs $S-1$ compute/uncompute Toffoli pairs, and the copy operations cost S Toffolis. Therefore, the total cost for an l_{max}-compute carry, and hence the whole block encoding, is $l_{max} + 2S - 1$ Toffolis. As usual, we don't count uncomputation Toffolis, for which measurement based uncomputation can be used.

2.2 Piecewise QSVT circuit

Standard QSVT applies a single polynomial to all the singular values of a block encoding. Here we wish to instead apply a different polynomial to each segment of our block encoding:

$$
\left(\begin{array}{cccc|c}
p_0(1) & & & & \\
& \ddots & & & \\
& & p_0\left(-1+\frac{2}{L_0}\right) & & \\
& & & \ddots & \\
& & & & p_{S-1}(1) \\
& & & & & \ddots \\
& & & & & & p_{S-1}\left(-1+\frac{2}{L_{S-1}}\right) \\
\hline
j_2 & & & & & & & j_3
\end{array}\right) \qquad (2)
$$

In QSVT the polynomial transformation performed is controlled by the choice of phase factor rotations. Hence by performing phase factor rotations conditionally depending on the values of the block registers we can apply different polynomial transformations to different segments of computational basis elements. As our block encoding is diagonal in the computational basis this allows us to apply different polynomials to different segments of our singular values.

Like the compute carry routine (Fig. 2 and Section 2.1), we utilise variable unary iteration [33] to efficiently apply different phase factor rotations for each section of computational basis elements. We achieve this by digitally pre-loading the phase factors into ancilla registers and then performing the rotations by controlled addition into a pre-prepared phase gradient register [35] as demonstrated by Fig. 4. Rather than inserting Fig. 4 into the QSVT circuit for each phase factor rotation, we can pre-load all of the phase factors in a single variable unary iteration, at the expense of more ancillas: Each set of controlled rotations requires their own set of $\log \frac{1}{\epsilon}$ ancillas (where ϵ is the accuracy to which one wishes to perform the rotations).

Utilising variable unary iteration to control off sections of computational basis elements makes our algorithm more efficient than applying different polynomials based on an inequality test as proposed in [29] which would cost Sn Toffoli gates compared to our $S-2$ Toffoli gates.

We have based our circuit off an indefinite parity version of QSVT [37] which requires $2d$ queries to the block encoding and $2d+1$ rotations to implement a d-degree indefinite parity polynomial. We present this variant as our later applications (Section 3) require indefinite parity. It would be equally possible to simply control the rotations of a standard QSVT circuit and restrict to definite parity polynomials [a]. Fig. 5

[a] For definite parity polynomials it would be necessary to modify the piecewise exact linear block encoding to range from 0 to 1 rather than -1 to 1. This can be simply achieved by removing both Hadamard gates and the leftmost X gate from the flag qubit in Fig. 1. Without this modification it would not be possible to approximate arbitrary states as the polynomial will be necessarily symmetric or anti-symmetric around 0.

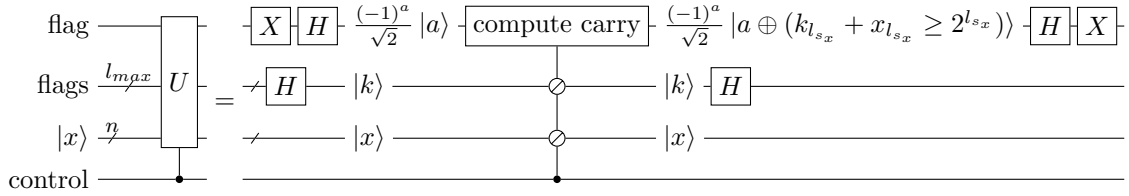

Figure 1: Circuit implementing controlled version of piecewise exact linear block encoding where $L_{s_x} = 2^{l_{s_x}}$ and $l_{max} = \max_i(l_i)$. The slash circle control $\oslash$ is used to denote multiplexed control [34] (e.g. in this case it means that the action of the compute carry depends on the values of k and x in the corresponding registers). The mid circuit kets indicate the basis states per register with the full state being a summation over a and k; after the application of compute carry these basis states are entangled.

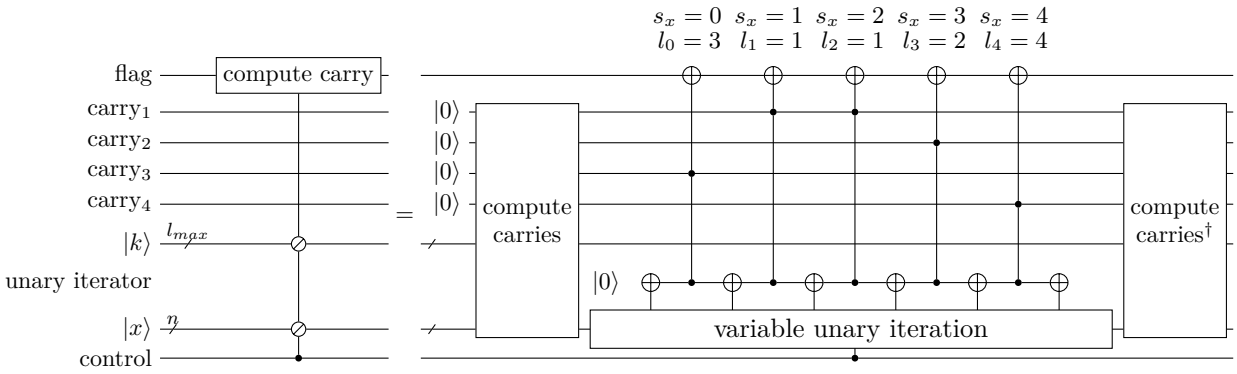

Figure 2: Example of compute carry in Fig. 1 for $l_0 = 3$, $l_1 = 1$, $l_2 = 1$, $l_3 = 2$ and $l_4 = 4$. The unary iteration over these segments to copy the correct carry bit into the flag register is performed by controlled variable unary iteration [33]. Compared to Fig. 1, ancillas carry_1 to carry_4, and an ancilla unary iterator qubit are shown. The variable unary iteration contains further internal ancilla qubits.

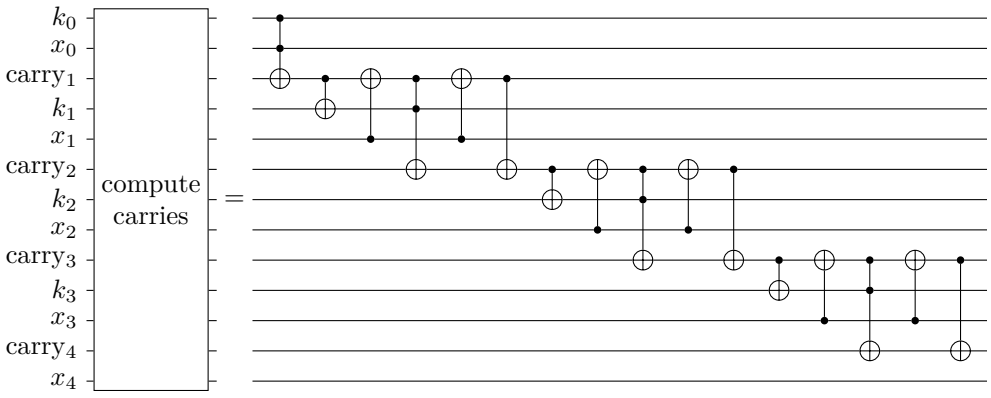

Figure 3: Example of adder in Fig. 2 for $L_{max} = 16$, $N = 32$. It is important to note that the construction of the adder leaves the x registers unaffected, so the variable unary iteration can operate directly on the x register next. x_4 is not involved in this operation as $l_{max} = 4$ is less than $n = 5$ and it is not necessary to compute carries for $l > l_{max}$. The qubits have been ordered differently to Fig. 2 to make the ladder structure more apparent.

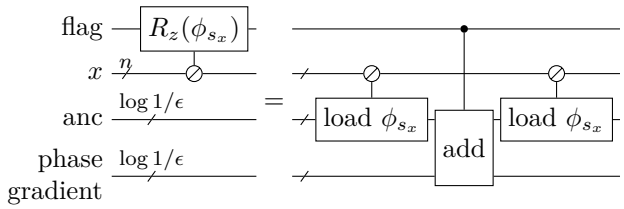

Figure 4: Circuit demonstrating how the variable unary iteration controlled rotations can be implemented by digitally loading the phase factors and adding into a phase gradient register. This technique avoids Clifford+T compilation of rotations[36], which would lead to a multiplicative factor $\log 1/\epsilon$ in T complexity.

demonstrates a piecewise indefinite parity QSVT circuit with 4 iterations to implement S different polynomials of degree 2 on S different segments of length $2^{l_{s_x}}$.

Each step of the circuit requires $\log \frac{1}{\epsilon} - 1$ compute/uncompute Toffoli pairs in order to perform the controlled rotation (via addition into a phase gradient register). The ancillas storing the phase factors must also be loaded/unloaded via variable unary iteration costing a total of $2(S-2)$ compute/uncompute Toffoli pairs. The remaining cost (not attributed to the inner block encoding) is due to the $l_{max}+1$-controlled Z gate in each step, costing a total of $2d(l_{max}+1)$ compute/uncompute Toffoli pairs.

2.3 State preparation

Implementing the piecewise QSVT circuit produces the block encoding described by (2). The state produced by applying this block encoding (piecewise QSVT circuit) to the uniform superposition is:

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} p_{s_x} \left(1 - 2\,\frac{x \bmod 2^{l_{s_x}}}{2^{l_{s_x}}}\right) |x\rangle\, |0\rangle^{\otimes l_{max}+1} + |\text{junk}\rangle \tag{3}$$

Here $|\text{junk}\rangle$ corresponds to all the terms where the flag qubits are not in the all zero state. Therefore, we can use amplitude amplification (reflecting around the all zero state for the flag qubits) to recover the desired state (1). The number of amplitude amplification steps required to achieve this is given by:

$$O\left(\sqrt{\frac{N}{\sum_{x=0}^{N-1}\left[p_{s_x}\left(1 - 2\frac{x \bmod 2^{l_{s_x}}}{2^{l_{s_x}}}\right)\right]^2}}\right) \tag{4}$$

We always want to use polynomials that are as large as possible (under the restraint of condition (1)) to minimise the number of amplitude amplification steps, hence we fit polynomials $\tilde{p}_i$ to the normalised amplitudes of the desired state but actually implement polynomials $p_i = \frac{\tilde{p}_i}{\tilde{p}_{max}}$ (where $\tilde{p}_{max}$ is the maximum value of all the polynomials over the continuous region $[-1, 1]$). As $\sum_{x=0}^{N-1}\left[\tilde{p}_{s_x}\left(1 - 2\,\frac{x \bmod 2^{l_{s_x}}}{2^{l_{s_x}}}\right)\right]^2 \approx 1$ due to normalisation of the desired state, the number of amplitude amplification steps is given by: $O(\tilde{p}_{max}\sqrt{N})$.

For desired states with amplitudes given by samples from fixed functions, increasing N corresponds to taking finer grained sampling of the function. As such the normalisation of such amplitudes will increase with $O(\sqrt{N})$. Hence if the maximum of the unnormalised polynomial approximation remains approximately the same with the finer sampling, the normalised maximum $\tilde{p}_{max}$ will scale with $O(\frac{1}{\sqrt{N}})$, and hence the number of amplitude amplification steps will scale with $O(1)$. This conclusion can similarly be reached by considering $\tilde{p}_{max}\sqrt{N}$ as the inverse "discretized L2-norm filling fraction" of $\tilde{p}_i$.

One caveat is that as $\tilde{p}_{max}$ is taken over the continuous region it may be larger than 1 even though all the normalised amplitudes are always less than 1. So for polynomials that oscillate wildly away from the sampled amplitudes this process is inefficient. It should be noted that fitting piecewise polynomials reduces the impact of the Runge Phenomena and hence polynomials produced by our approximations are less prone to this wild oscillation than in prior work.

2.4 Resource costs

The total cost of running a piecewise QSVT circuit to implement S indefinite parity polynomials of degree d with maximum segment size $2^{l_{max}}$ is $(2d+1)(\log \frac{1}{\epsilon} - 1) + 2(S-2) + 2d(l_{max}+1) + 2d(l_{max}+2S-1)$ Toffolis.

When performing amplitude amplification with A steps we repeat this circuit $2A$ times along with $2A$ single qubit rotations, A times $(l_{max}+2)$-controlled Z gates and A times $(n+l_{max}+2)$-controlled Z gates. Hence the cost of preparing a given n-qubit state is $O\left(\tilde{p}_{max}\sqrt{N}\max(n, dl_{max}, d\log\frac{1}{\epsilon}, dS)\right)$ Toffolis. As discussed in Section 2.3, if the state we wish to load has amplitudes sampled from some function (such that increasing N corresponds to taking a finer graining of the function), then the cost of preparing the n-qubit state is:

$$O\left(\max\left(n, dl_{max}, d\log\frac{1}{\epsilon}, dS\right)\right) \text{ Toffolis} \tag{5}$$

In the case described in Section 3.1 we have $S = O(n)$, and hence this can be simplified to just:

$$O\left(d\max\left(n, \log\frac{1}{\epsilon}\right)\right) \text{ Toffolis} \tag{6}$$

The algorithm acts on: n qubits to store the desired state, $l_{max}+1$ block encoding flag qubits, $(2d+1)\log\frac{1}{\epsilon}$ ancillas for digitally loading phase rotations, a QSVT flag qubit and an amplitude amplification ancilla qubit. Additional qubits are temporarily required: $n + l_{max}$ ancillas to perform the compute carry, n ancillas to perform the variable unary iteration on

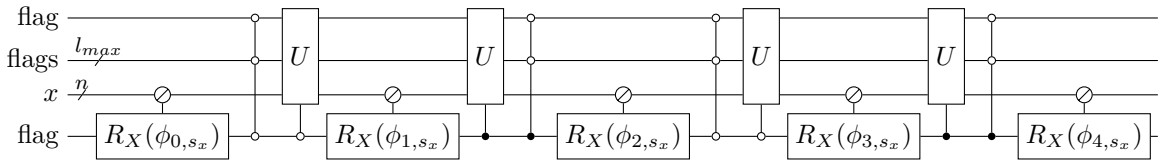

Figure 5: Piecewise indefinite parity QSVT circuit with 4 iterations to implement S different polynomials of degree 2 on S different segments of length $2^{l_{s_x}}$. The multiplexed rotations are implemented efficiently using the variable unary iteration subroutine described in Section 2.1.

n qubits and $n + l_{max} + 2$ ancillas to perform the $n + l_{max} + 2$-controlled Z gate required in amplitude amplification. After accounting for the reuse of temporary ancillas, $2n + 2l_{max} + (2d + 1)\log\frac{1}{\epsilon} + 5$ total qubits are required by this algorithm.

Above we assumed the use of clean ancillas to minimise the Toffoli count; space-time tradeoffs using conditionally clean or dirty qubits are possible [38, 39].

3 Applications

Our algorithm allows more states to be prepared efficiently. Existing QSVT state preparation techniques required the desired state to be approximated by a single global polynomial. Our approach relaxes this restriction to allow approximation by piecewise polynomials. In Section 3.1 we will explore amplitudes sampled from functions which are hard to approximate with the former, but efficiently approximated by the later. In Section 3.2 we discuss loading the piecewise polynomial B-spline window function and the exponential performance boost it lends to Quantum Phase Estimation. Then in Section 3.3 we illustrate a classical algorithm for finding the optimal piecewise polynomial to approximately load a given generic state.

3.1 Approximating nearly-analytic functions

It is well known that analytic functions admit a d-degree polynomial approximation with approximation error $O(\exp(-d))$ [40]. Hence for such functions a single global polynomial will be sufficiently efficient. However, for functions with singularities on the interval we wish to approximate (e.g. x^α on $x, \alpha \in (0, 1)$) it is not possible to find a single global polynomial that efficiently approximates the function. However, by utilising our technique with piecewise polynomials we can efficiently approximate some such functions with approximation error $O(\exp(-d))$ as demonstrated in Sections 3.1.1 and 3.1.2.

Our technique (similar to Section II E [33]) for dealing with such situations uses a cascade of exponentially smaller segments as we approach the singularity (e.g. $[2^{-i-1}, 2^{-i}]$ for a singularity at zero). This allows us to resolve the tricky behaviour up to a desired finite resolution 2^{-n} whilst only paying a logarithmic cost in number of segments $S = O(n)$. This technique will also be useful for many other states which exhibit large discontinuities, singularities or sharp cusps which would otherwise require a very high degree single polynomial to approximate.

In our examples we consider only the case with a single singularity at 0 when attempting to approximate the range $[2^{-n}, 1]$. This can be easily extended without loss of generality to any number of singularities within the interval of interest by dividing the interval up either side of each singularity at the closest multiples of 2^{-n}. On an n-qubit state only a single amplitude can lie in each of the remaining 2^{-n} sized intervals, so they can be covered by single 0-degree polynomial segments.

We now illustrate 2 explicit examples of how our technique produces exponential speed ups in loading common functions compared to naive state preparation and single polynomial QSVT. We also show an improvement over black box techniques for quantum state preparation [33, 41]. In general, the implementation of the black box oracle in each round of amplitude amplification can be costly: with quantum arithmetic evaluation of a piecewise polynomial this cost is $\Omega(n^2 d)$ [28], making our approach more efficient. Whilst we are unaware of specific quantum algorithms requiring these states, the examples effectively illustrate the increased capability of our approach compared to employing a single polynomial approximation.

3.1.1 x^α

In prior work [29] on QSVT state preparation it was explicitly stated that "one current drawback of our method is that it cannot efficiently prepare the state representing $\sqrt{x}$ for $x \in [0, 1]$". Here we demonstrate that we can even approximately load the state $\frac{1}{\mathcal{N}}\sum_{x=0}^{2^n-1}(2^{-n}x)^\alpha |x\rangle$ for $\alpha \in (0, 1)$ using our technique.

Theorem 8.1 in [40] states that any function f analytic and bounded by M on an open Bernstein ellipse E_ρ with semi-major axis $\frac{\rho+\rho^{-1}}{2}$ has a d-degree Chebyshev approximation f_d satisfying:

$$|f - f_d| \leq \frac{2M}{\rho - 1}\rho^{-d} \tag{7}$$

Considering the $S = n$ segments $[2^{-i-1}, 2^{-i}]$ separately for $f = x^\alpha$, then the maximal open Bernstein

ellipse we can draw on each segment has $\rho = 3 + \sqrt{8}$. Therefore, we get the following bound on the error in each segment:

$$|f - f_{d,i}| \leq \frac{2 * 2^{-\alpha i}}{\mathcal{N}(2+\sqrt{8})}(3+\sqrt{8})^{-d} \tag{8}$$

Therefore, the d-degree n-piecewise approximation f_d satisfies the following bound for $[2^{-n}, 1]$ as $\frac{1}{\mathcal{N}} < 1$ for $n > 1$.

$$|f - f_d| \leq \frac{2}{2+\sqrt{8}}(3+\sqrt{8})^{-d} \tag{9}$$

Further, we can utilise a constant approximation of 0 in the remaining segment $[0, 2^{-n})$.

The normalisation $\mathcal{N}$ is given by:

$$\sqrt{\sum_{x=0}^{2^n-1}(2^{-n}x)^{2\alpha}} = O\left(\sqrt{\frac{N}{2\alpha+1}}\right) \tag{10}$$

Hence, $\tilde{p}_{max} = O\left(\sqrt{\frac{2\alpha+1}{N}}\right)$. Therefore, our technique can efficiently prepare x^α for $\alpha \in (0,1)$ on n-qubits with accuracy ϵ in $O(\sqrt{2\alpha}\log\frac{1}{\epsilon}\max(n, \log\frac{1}{\epsilon}))$ Toffolis — providing exponential speed-up on naive state preparation. We note that a faster technique [42] exists for preparing the special case $\sqrt{x}$, however, that approach does not generalise efficiently to x^α.

3.1.2 $\log x$

To approximately load the state $\frac{1}{\mathcal{N}}\sum_{x=1}^{2^n-1}\log 2^{-n}x\,|x\rangle$, we need to approximate the function $f(x) = \frac{1}{\mathcal{N}}\log 2^{-n}x$ for $x \in [1, 2^n - 1]$, or equivalently $\tilde{f}(x) = \frac{1}{\mathcal{N}}\log x$ for $x \in [2^{-n}, 1 - 2^{-n}]$.

Here, we will use a Taylor rather than Chebyshev approximation. The Taylor series expansion of $\tilde{f}$ around $x = a$ is given by:

$$\tilde{f}(x) = \sum_{k=0}^{d}\frac{\tilde{f}^{(k)}(a)}{k!}(x-a)^k + R_d(x,a) \tag{11}$$

where the Taylor remainder is

$$\begin{aligned} R_d(x,a) &= \frac{\tilde{f}^{(d+1)}(x_*)}{(d+1)!}(x-a)^{d+1} \\ &= \frac{1}{\mathcal{N}}\frac{(-1)^{d-1}}{d+1}x_*^{-d-1}(x-a)^{d+1} \end{aligned} \tag{12}$$

and $x_* \in (a, x)$.

We can split the range $[2^{-n+1}, 1]$ into segments $[2^{-i-1}, 3*2^{-i-2}]$ and $[3*2^{-i-2}, 2^{-i}]$ for $i \in \mathbb{Z}_{n-1}$, and then take a Taylor series expansion about 2^{-i-1} in the former segments and 2^{-i} in the latter. The maximal error R_d occurs at $x_* = 2^{-i-1}$ and $x = 3*2^{-i-2}$ for the former segments, whereas it occurs at $x_* = 3*2^{-i-2}$ and $x = 3*2^{-i-2}$ for the latter segments. Therefore, the error in each segment is bounded by:

$$\begin{aligned} |R_{i,d}(x,a)| &\leq \frac{1}{\mathcal{N}(d+1)}2^{(i+1)(d+1)-(i+2)(d+1)} \\ &= \frac{1}{\mathcal{N}}\frac{2^{-d-1}}{d+1} \end{aligned} \tag{13}$$

Therefore, combining these into a single $2(n-1)$-piecewise Taylor series approximation f_d achieves error bounded by:

$$|\tilde{f} - f_d| \leq \frac{1}{\mathcal{N}}\frac{2^{-d-1}}{d+1} \tag{14}$$

As $\frac{|\log 2^{-n}|}{\mathcal{N}} < 1$, we have

$$|\tilde{f} - f_d| \leq \frac{1}{n(d+1)}2^{-d-2}. \tag{15}$$

The above does not cover the range $[0, 2^{-n+1})$, however, only two amplitudes lie in this range so they may be exactly approximated by a 1-degree polynomial, giving $S = 2n - 1$ total segments.

The normalisation $\mathcal{N}$ is given by:

$$\sqrt{\sum_{x=0}^{2^n-1}\log^2 2^{-n}x} = O\left(\sqrt{N}\right) \tag{16}$$

Hence, $\tilde{p}_{max} = O\left(\frac{n}{\sqrt{N}}\right)$. Therefore, preparing $\log x$ on n-qubits with accuracy ϵ costs $O(n\log\frac{1}{\epsilon}\max(n, \log\frac{1}{\epsilon}))$ Toffolis.

3.2 B-spline window

The B-spline window function [43] is a family of classical windowing functions constructed out of piecewise polynomials. The m-th B-spline $w_m(x)$ is obtained by m-fold self-convolution of the rectangular function:

$$w_m(x) = \int_{-\infty}^{\infty} w_1(mz)w_{m-1}\left(\frac{m}{m-1}(x-z)\right)dz \tag{17}$$

$$w_1(x) = \begin{cases} 1 & |x| < 2^{n-1} \\ 0 & |x| \geq 2^{n-1} \end{cases} \tag{18}$$

Hence the m-th B-spline is an m-piecewise polynomial of degree $m-1$. As such it is immediately obvious that our technique can prepare the following state (with accuracy ϵ) in $O(m^{\frac{5}{4}}\max(n, m, \log\frac{1}{\epsilon}))$ Toffolis:

$$\frac{1}{\tilde{\mathcal{N}}}\sum_{x=2^{n-1}}^{2^{n-1}-1}w_m(x)\,|x\rangle \approx \frac{m^{m+1}}{2^{n(m-\frac{1}{2})}}\left(\frac{\pi}{3m}\right)^{\frac{1}{4}}\sum_{x=2^{n-1}}^{2^{n-1}-1}\sum_{p=0}^{m}\frac{(-1)^p\max\left(0, x-\left(p-\frac{m}{2}\right)\frac{2^n}{m}\right)^{m-1}}{p!(m-p)!}|x\rangle \tag{19}$$

Crucially, the m-fold self-convolution (17) corresponds to self-multiplying in the Fourier domain, and hence the Fourier transform of the m-th B-spline window function is:

$$\left(\frac{\pi}{3m}\right)^{\frac{1}{4}}\left(\frac{\sin\frac{x\pi}{m}}{\frac{x\pi}{m}}\right)^m \tag{20}$$

which has the useful property of being sharply peaked near $x = 0$.

Efficiently preparing the B-spline window function will be immensely useful for improving quantum algorithms. One of the main subroutines of quantum algorithms is Quantum Phase Estimation (QPE) which is used for finding the eigenvalues of a given unitary. The biggest problem with QPE is its fat tails; the probability of measuring an inaccurate eigenvalue is non-negligible if the eigenvalue is not guaranteed to be exactly specified in binary at the resolution used. This is typically countered through preparing a window function in the ancillas before applying QPE and increasing the resolution of the phase estimation beyond the precision desired. The window function has a narrow profile in the Fourier domain and hence narrows the profile of the probability distribution output by the QPE algorithm. One method for measuring the effectiveness of the window functions is by how much the resolution of the phase estimation must be increased by (or equivalently the number of extra ancillas required) to reduce the probability of measuring an energy outside the confidence interval to below a given value δ [44]. Extra ancilla qubits are included in the final prepared state, but unlike the base ancillas—which increase the resolution of the QPE—they serve instead to reduce the value of δ. The current state of the art is the Kaiser window function [45] which reduces the number of extra ancillas required to $O(\log\log\frac{1}{\delta})$. Thereom 1 (proof in Appendix A) demonstrates that the B-spline window achieves the same asymptotic complexity and hence provides a viable alternative to the Kaiser window:

Theorem 1. *To achieve a given probability δ of measuring energy outside of a given confidence interval ϵ (when using $O(\log\frac{1}{\delta})$-th B-spline boosted QPE), one must use $O(\log\frac{1}{\epsilon}+\log\log\frac{1}{\delta})$ ancillary qubits.*

Fig. 6 numerically demonstrates this doubly exponential decay. The exponent for the Kaiser window is larger than the B-Spline window, however, even with a very small number of extra ancillas both windows achieve such a small δ that it will likely be outweighed by other considerations such as logical error floor [46].

We compare the cost of preparing the B-spline window using our technique (with the improvements outlined in Section 4.1) to the cost of preparing the Kaiser window using the best technique known to us [29, 44]. Fig. 7 demonstrates that our technique provides a 50 fold decrease in preparation cost when using 4 extra ancillas. The code for producing both these figures can be found here: [47]. Therefore, in the early fault tolerant regime (where improvements in δ beyond experimental constraints cannot be realised) utilising the B-Spline window via our technique provides a significant reduction in cost for performing Quantum Phase Estimation.

It should be noted that in order to successfully prepare the B-spline window using our technique m must be a power of 2, which happily corresponds to including an integer number of extra ancilla qubits.

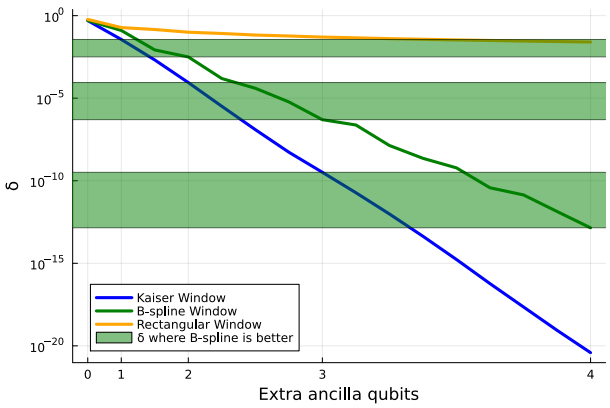

Figure 6: Numerical results demonstrating $O(\log\log\frac{1}{\delta})$ dependence of extra ancilla qubits for QPE when using both the best Kaiser Window and the best B-Spline window [b]. Comparison with the rectangular window demonstrates the improvement over "unboosted" QPE. Both the Kaiser Window and B-Spline window demonstrate an exponential improvement, though the Kaiser window has a slightly larger exponent. The green regions indicate target error rates of δ for which the same number of extra ancillas are required for both the Kaiser window and the B-Spline window. In these regions it is optimal to use the B-Spline window due to its lower preparation cost. This figure includes calculations for fractional qubit counts as it is possible to calculate δ as if we used a confidence interval not equal to a power of 2.

The B-Spline function is an example of finding a piecewise polynomial that produces a desired property (being sharply peaked in the Fourier domain). Therefore, our technique offers a general framework by which one can efficiently prepare states with desired properties through careful choice of piecewise polynomials. This is potentially more powerful than simply approximating existing states known to have the desired property.

Function	Toffoli cost
$\left(\frac{x}{N}\right)^{\alpha}, \alpha\in(0,1)$	$O\left(\sqrt{2\alpha}\log\frac{1}{\epsilon}\max(n,\log\frac{1}{\epsilon})\right)$
$\log\frac{x}{N}$	$O\left(n\log\frac{1}{\epsilon}\max(n,\log\frac{1}{\epsilon})\right)$
m-th B-spline window	$O\left(m^{\frac{5}{4}}\max\left(n,\log\frac{1}{\epsilon}\right)\right)$

Table 1: Asymptotic Toffoli costs for loading these functions into the amplitudes of states on n qubits with an error tolerance of ϵ. The tail probability δ when utilising the m-th B-spline window in QPE is given by $O(\exp(-m))$ as shown by Lemma 1.2.

[b] The tail probability depends only very weakly on the number of base qubits which correspond to how finely the window function is sampled. Hence, this figure is expected to be very similar for all choices of number of base qubits. This particular figure was generated for 10 base qubits.

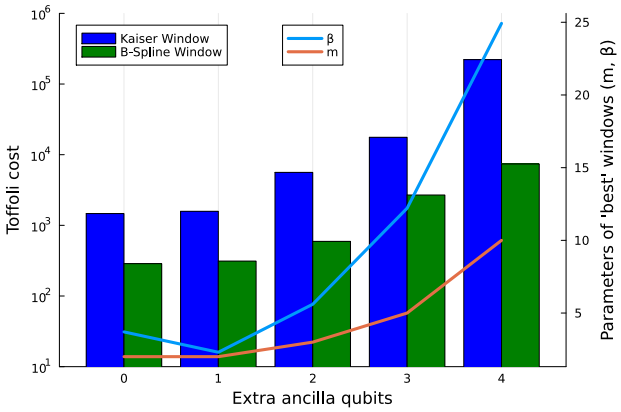

Figure 7: Numerical results demonstrating Toffoli cost of preparing the "best" β-Kaiser and m-th B-Spline windows for a given number of extra ancilla qubits. The parameters (β and m) of the windows are chosen to minimise the probability of measuring the wrong energy when performing QPE with 25 base qubits. Kaiser window is costed using the state-of-the-art approach from Appendix E of [44] (derived from [29]), whereas the B-spline window is costed using our piecewise QSVT according the formulas given in 2.4 incorporating the improvements from 4.1. These results demonstrate up to 50x lower cost for the B-spline window.

3.3 Generic optimal algorithm

We present Algorithm 1 for finding the optimal splitting of the amplitudes into segments to be approximated that runs in $O(N \log_2 N)$ time. It takes a fixed degree d and maximum error tolerance ϵ and greedily finds the largest segment that a polynomial can approximate sequentially. This produces the smallest possible number of segments that satisfy the constraints, as for any given data point all the possible larger segments it lies in will have been considered and rejected by the algorithm.

Section 3.1 demonstrated upper bounds on the cost of approximating certain states by piecewise polynomials. In reality the chosen segment sizes are suboptimal and we should instead run the algorithm presented here upon these states to find the optimal divisions into segments. The resulting approximation will improve upon the results demonstrated in Section 3.1.

Our algorithm utilises an ErrPolyApprox subroutine which finds a good polynomial approximation to the given set of amplitudes and returns the L^∞ norm error of the approximation. We utilised least squares fitting to achieve this goal, but other approaches are feasible (e.g. polynomial frame approximations or regularised least squares). If it is known that the amplitudes are sampled from a particular function then other methods such as the Remez algorithm might be suitable.

Algorithm 1: Performs at most $N \log_2 N$ queries to ErrPolyApprox and returns the optimal cuts between the pieces of the polynomial

Input: d, ϵ (target error), N, amplitudes
Output: cuts between polynomial segments

cuts $\leftarrow$ [ ];
$RHS \leftarrow N$;
while $RHS > 0$ **do**
  $l \leftarrow \max\left(l \text{ s.t. } 2^l \mid RHS\right)$;
  $LHS \leftarrow \max(RHS - 2^l, 0)$;
  while ErrPolyApprox$(amplitudes[LHS{:}RHS]) > \epsilon$ **do**
    $LHS \leftarrow \frac{LHS+RHS}{2}$;
  cuts.append(LHS);
  $RHS \leftarrow LHS$;
return cuts;

4 Further improvements

4.1 Block Encoding

We can make our piecewise linear block encoding more efficient in some scenarios depending upon the sizes of segments we desire. If all the segments have the same size then no variable unary iteration is needed and a single Toffoli from the relevant carry to the flag qubit will suffice. This is also true if there is only one segment in which case our technique reproduces the prior work [29] with a new inner block encoding. In both these cases the cost of our inner block encoding is just $l_{max}+1$ Tofollis. This is significantly cheaper than inner block encodings introduced by previous work such as the sin [29] and approximate linear [30] blocking encodings as it avoids any compilation into Clifford+T of rotation gates[c]. Our block encoding does require more ancillas than the sin block encoding, but these ancillas will likely be required anyway by the subsequent algorithm so are well worth the ≈ 15 times reduction in Toffolis.

Another efficiency saving can be made to the block encoding if there are only a small k number of unique segment sizes. Here we could utilise variable unary iteration to load a label for each segment size into an ancilla register before performing QSVT and then control off this register (instead of the block register) when copying the carries in compute carry. This would reduce the cost of the block encoding in each step of QSVT to $l_{max} + 2k - 1$.

[c]To compare like for like we must slightly modify our block encoding to range from 0 to 1. This can be simply achieved by removing both Hadamard gates and the leftmost X gate from the flag qubit in Fig. 1.

4.2 Priors

We can reduce the number of amplitude amplification steps by applying our block encoding to an efficiently loaded prior distribution as in [48, 49]. By applying our QSVT circuit to the uniform superposition we are implicitly choosing a uniform prior. If we instead applied our circuit to the state $\sum_x c_x |x\rangle$, then we would use polynomial approximations to a_x/c_x (where a_x are the amplitudes of the state we wish to approximate). If c_x is a good approximation then a_x/c_x will be closer to the uniform distribution than a_x, so $\tilde{p}_{max}$ will be smaller and hence the number of amplitude amplification rounds will be reduced. This allows us to incorporate all the improvements demonstrated in [48] achieved by utilising efficiently preparable reference states as priors.

5 Conclusion

We have presented a new algorithm for preparing states that, in some explicit instances, demonstrates an exponential speedup over naive state preparation. In particular, our approach can handle nearly-analytic functions with cusps and singularities. Furthermore, we have demonstrated that the B-spline window function provides a viable alternative to the state-of-the-art Kaiser window for boosting QPE. We also present numerical results demonstrating that the B-spline function (implemented via our algorithm) is 50 times cheaper to prepare than the Kaiser window (implemented via existing state-of-the-art methods).

We expect that our algorithm in combination with the B-spline window will be adopted by early fault tolerant quantum phase estimation protocols when it will be crucial to reduce the Toffoli cost of circuits. QPE is a prolific component of numerous quantum algorithms and hence the impact of this improvement will be very widespread.

Further, our algorithm will prove useful for loading states with amplitudes drawn from smooth functions with sharp discontinuities. For example, loading a 2D or 3D grid of smooth data into the amplitudes of the 1D computational basis. This is a problem commonly encountered in Quantum Computational Fluid Dynamics [10].

In classical computer science, splines (piecewise polynomials) are an incredibly popular and powerful approximation technique and this is likely to be reflected in quantum computing. Hence, our algorithm which allows the preparation of spline states will likely find broad spread application.

We have only demonstrated our approach to loading states with real valued amplitudes. However, it is simple to extend our results to states with complex amplitudes. The only change we need to make is that now we approximate these states using piecewise polynomials with complex coefficients.

The downside of our approach is that our block encoding requires more flag and ancilla qubits than the prior work. We require $2n + 2l_{max} + 5$ qubits compared to $n+4$ required by [29]. We argue that this extra qubit cost is acceptable for state preparation as it is highly likely that any subsequent algorithm will reuse these ancillas plus more so it will not increase the cost of the total quantum algorithm.

Additionally, it is relevant to note that in the regime of small quantum states (fewer than 9 base qubits), the method of Low, Kliuchnikov, and Schaeffer (LKS) for arbitrary state preparation [34] would be more efficient than our approach and that of McArdle et al. [29]. However, the cost of the LKS method increases rapidly with system size, making it less efficient than QSVT-based techniques even for moderately large states.

It is important to note that we do not apply QSVT piecewise to different segments of eigenvalues, but rather to different segments of Hilbert space enumerated by the computational basis elements. Here the later acts as the former because the eigenspaces of our block encoding align with the computational basis and the relationship between the eigenvalues and computational basis is known. In general this is not true and to perform different QSVT polynomial transformations to different sections of eigenvalues of an arbitrary block encoding would be much more costly and rely on some eigenvalue thresholding or windowing technique.

Acknowledgments

We are grateful to Bjorn Berntson for very helpful discussions regarding polynomial approximations and to Vlad Gheorghiu and his team for comments on the manuscript. This work was partially funded by softwareQ Inc.

References

[1] Alán Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. "Simulated quantum computation of molecular energies". Science **309**, 1704–1707 (2005).

[2] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. "Quantum computational chemistry". Rev. Mod. Phys. **92**, 015003 (2020).

[3] Yuan Su, Dominic W. Berry, Nathan Wiebe, Nicholas Rubin, and Ryan Babbush. "Fault-tolerant quantum simulations of chemistry in first quantization". PRX Quantum **2**, 040332 (2021).

[4] William J. Huggins, Oskar Leimkuhler, Torin F. Stetina, and K. Birgitta Whaley. "Efficient

state preparation for the quantum simulation of molecules in first quantization" (2024). arXiv:2407.00249.

[5] Scott Aaronson. "Read the fine print". Nature Phys. **11**, 291–293 (2015).

[6] Shouvanik Chakrabarti, Rajiv Krishnakumar, Guglielmo Mazzola, Nikitas Stamatopoulos, Stefan Woerner, and William J. Zeng. "A Threshold for Quantum Advantage in Derivative Pricing". Quantum **5**, 463 (2021).

[7] Javier Gonzalez-Conde, Ángel Rodríguez-Rozas, Enrique Solano, and Mikel Sanz. "Efficient Hamiltonian simulation for solving option price dynamics". Phys. Rev. Res. **5**, 043220 (2023). arXiv:2101.04023.

[8] Steven Herbert. "No quantum speedup with grover-rudolph state preparation for quantum monte carlo integration". Phys. Rev. E **103**, 063302 (2021).

[9] Alexander M. Dalzell et al. "End-To-End Resource Analysis for Quantum Interior-Point Methods and Portfolio Optimization". PRX Quantum **4**, 040325 (2023). arXiv:2211.12489.

[10] Leigh Lapworth. "A Hybrid Quantum-Classical CFD Methodology with Benchmark HHL Solutions" (2022). arXiv:2206.00419.

[11] Shi-Ju Ran. "Encoding of matrix product states into quantum circuits of one- and two-qubit gates". Physical Review A**101** (2020).

[12] A. Holmes and A. Y. Matsuura. "Efficient quantum circuits for accurate state preparation of smooth, differentiable functions". In 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). Pages 169–179. Los Alamitos, CA, USA (2020). IEEE Computer Society.

[13] Ar A Melnikov, A A Termanova, S V Dolgov, F Neukart, and M R Perelshtein. "Quantum state preparation using tensor networks". Quantum Science and Technology **8**, 035027 (2023).

[14] Michael Lubasch, Jaewoo Joo, Pierre Moinier, Martin Kiffner, and Dieter Jaksch. "Variational quantum algorithms for nonlinear problems". Phys. Rev. A **101**, 010301 (2020).

[15] Rohit Dilip, Yu-Jie Liu, Adam Smith, and Frank Pollmann. "Data compression for quantum machine learning". Phys. Rev. Res. **4**, 043007 (2022).

[16] Jason Iaconis, Sonika Johri, and Elton Yechao Zhu. "Quantum state preparation of normal distributions using matrix product states". npj Quantum Inf. **10**, 15 (2024). arXiv:2303.01562.

[17] Daniel Malz, Georgios Styliaris, Zhi-Yuan Wei, and J. Ignacio Cirac. "Preparation of matrix product states with log-depth quantum circuits". Phys. Rev. Lett. **132**, 040404 (2024).

[18] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. "Quantum state preparation with optimal circuit depth: Implementations and applications". Phys. Rev. Lett. **129**, 230504 (2022).

[19] Jonathan Welch, Daniel Greenbaum, Sarah Mostame, and Alan Aspuru-Guzik. "Efficient quantum circuits for diagonal unitaries without ancillas". New Journal of Physics **16**, 033040 (2014).

[20] Julien Zylberman and Fabrice Debbasch. "Efficient quantum state preparation with Walsh series". Phys. Rev. A **109**, 042401 (2024). arXiv:2307.08384.

[21] Mudassir Moosa, Thomas W. Watts, Yiyou Chen, Abhijat Sarma, and Peter L. McMahon. "Linear-depth quantum circuits for loading Fourier approximations of arbitrary functions". Quantum Sci. Technol. **9**, 015002 (2024). arXiv:2302.03888.

[22] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum generative adversarial networks for learning and loading random distributions". npj Quantum Information**5** (2019).

[23] Kouhei Nakaji, Shumpei Uno, Yohichi Suzuki, Rudy Raymond, Tamiya Onodera, Tomoki Tanaka, Hiroyuki Tezuka, Naoki Mitsuda, and Naoki Yamamoto. "Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators". Phys. Rev. Res. **4**, 023136 (2022). arXiv:2103.13211.

[24] Gabriel Marin-Sanchez, Javier Gonzalez-Conde, and Mikel Sanz. "Quantum algorithms for approximate function loading". Phys. Rev. Res. **5**, 033114 (2023). arXiv:2111.07933.

[25] Christoph Sünderhauf, Earl Campbell, and Joan Camps. "Block-encoding structured matrices for data input in quantum computing". Quantum **8**, 1226 (2024). arXiv:2302.10949.

[26] Aleksei V. Ivanov, Christoph Sünderhauf, Nicole Holzmann, Tom Ellaby, Rachel N. Kerber, Glenn Jones, and Joan Camps. "Quantum computation for periodic solids in second quantization". Phys. Rev. Res. **5**, 013200 (2023).

[27] Mihir K. Bhaskar, Stuart Hadfield, Anargyros Papageorgiou, and Iasonas Petras. "Quantum algorithms and circuits for scientific computing". Quant. Inf. Comput. **16**, 0197–0236 (2016). arXiv:1511.08253.

[28] Thomas Häner, Martin Roetteler, and Krysta M. Svore. "Optimizing Quantum Circuits for Arithmetic" (2018). arXiv:1805.12445.

[29] Sam McArdle, András Gilyén, and Mario Berta. "Quantum state preparation without coherent arithmetic" (2022). arxiv:2210.14892.

[30] Javier Gonzalez-Conde, Thomas W. Watts, Pablo Rodriguez-Grasa, and Mikel Sanz. "Efficient quantum amplitude encoding of polynomial functions". Quantum **8**, 1297 (2024). arXiv:2307.10917.

[31] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics". In 51st Annual ACM SIGACT Symposium on Theory of Computing. (2018). arXiv:1806.01838.

[32] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. "Grand Unification of Quantum Algorithms". PRX Quantum **2**, 040203 (2021). arXiv:2105.02859.

[33] Yuval R. Sanders, Dominic W. Berry, Pedro C.S. Costa, Louis W. Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush. "Compilation of fault-tolerant quantum heuristics for combinatorial optimization". PRX Quantum**1** (2020).

[34] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. "Trading T-gates for dirty qubits in state preparation and unitary synthesis" (2018). arxiv:1812.00954.

[35] Craig Gidney. "Halving the cost of quantum addition". Quantum **2**, 74 (2018).

[36] Alex Bocharov, Martin Roetteler, and Krysta M. Svore. "Efficient synthesis of universal repeat-until-success quantum circuits". Physical Review Letters**114** (2015).

[37] Christoph Sünderhauf. "Generalized quantum singular value transformation" (2023). arXiv:2312.00723.

[38] Tanuj Khattar and Craig Gidney. "Rise of conditionally clean ancillae for optimizing quantum circuits" (2024). arXiv:2407.17966.

[39] Junhong Nie, Wei Zi, and Xiaoming Sun. "Quantum circuit for multi-qubit Toffoli gate with optimal resource" (2024). arXiv:2402.05053.

[40] Lloyd N. Trefethen. "Approximation theory and approximation practice, extended edition". Society for Industrial and Applied Mathematics. Philadelphia, PA (2019).

[41] Johannes Bausch. "Fast Black-Box Quantum State Preparation". Quantum **6**, 773 (2022). arXiv:2009.10709.

[42] Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. "Black-box quantum state preparation without arithmetic". Phys. Rev. Lett. **122**, 020502 (2019).

[43] K. Toraichi, M. Kamada, S. Itahashi, and R. Mori. "Window functions represented by b-spline functions". IEEE Transactions on Acoustics, Speech, and Signal Processing **37**, 145–147 (1989).

[44] Sean Greenaway, William Pol, and Sukin Sim. "A case study against QSVT: assessment of quantum phase estimation improved by signal processing techniques" (2024). arXiv:2404.01396.

[45] Dominic W. Berry, Yuan Su, Casper Gyurik, Robbie King, Joao Basso, Alexander Del Toro Barba, Abhishek Rajput, Nathan Wiebe, Vedran Dunjko, and Ryan Babbush. "Analyzing Prospects for Quantum Advantage in Topological Data Analysis". PRX Quantum **5**, 010319 (2024). arXiv:2209.13581.

[46] Rajeev Acharya et al. "Quantum error correction below the surface code threshold" (2024). arXiv:2408.13687.

[47] Oliver O'Brien and Christoph Sünderhauf (2025). code: https://doi.org/10.5281/zenodo.14794185.

[48] Jessica Lemieux, Matteo Lostaglio, Sam Pallister, William Pol, Karthik Seetharam, Sukin Sim, and Burak Şahinoğlu. "Quantum sampling algorithms for quantum state preparation and matrix block-encoding" (2024). arXiv:2405.11436.

[49] Ryan Babbush, Dominic W. Berry, Jarrod R. McClean, and Hartmut Neven. "Quantum simulation of chemistry with sublinear scaling in basis size". npj Quantum Information**5** (2019).

A B-Spline Boosted Quantum Phase Estimation

Here we present a more detailed explanation of how the B-spline window function can be utilised to improve Quantum Phase Estimation.

Given an initial state $|\psi_i\rangle$ that is an eigenstate of U with eigenvalue E_i. Standard QPE progresses by applying U to the initial state conditional upon the value of l ancilla qubits initialised in a uniform superposition. If instead the l ancilla qubits are initialised with the B-spline window function then after this step we would have:

$$\frac{1}{\tilde{\mathcal{N}}} \sum_{x=-2^{l-1}+1}^{2^{l-1}} w_m(x) \, |x\rangle \, e^{iE_i x} \, |\psi_i\rangle \tag{21}$$

The next step of QPE is to apply the inverse Quantum Fourier Transform. The effect of this can be easily evaluated as multiplying by $e^{iE_i x}$ before a Fourier transform is equivalent to shifting by E_i after a Fourier transform. As the Fourier transform of the m-th B-spline w_m is given by [43], we have:

$$\frac{1}{\mathcal{N}} \sum_{k=-2^{l-1}+1}^{2^{l-1}} \left(\frac{\sin \frac{(k-E_i)\pi}{m}}{\frac{(k-E_i)\pi}{m}} \right)^m |k\rangle \, |\psi_i\rangle \tag{22}$$

As this distribution is sharply spiked near $k = E_i$, if we measure the ancilla register we are highly likely to get a value close to E_i.

We can prove this by bounding the tail probabilities, which first requires that we approximate $\mathcal{N}$.

Lemma 1.1. *Let $\mathcal{N}$ by defined as:*

$$\mathcal{N} := \sqrt{\sum_{k=-2^{l-1}+1}^{2^{l-1}} \left| \frac{\sin \frac{(k-E_i)\pi}{m}}{\frac{(k-E_i)\pi}{m}} \right|^{2m}} \tag{23}$$

Then,

$$\mathcal{N} \approx \left(\frac{3m}{\pi}\right)^{\frac{1}{4}} \tag{24}$$

Proof. We utilise the same technique as [45] used to approximate the normalisation of the Kaiser window, whereby we approximate the centre of the distribution by a Gaussian. This Gaussian is found by Taylor expanding the logarithm of the distribution:

$$\log\left(\frac{\sin\frac{(k-E_i)\pi}{m}}{\frac{(k-E_i)\pi}{m}}\right)^m = m\log\frac{\sin\frac{(k-E_i)\pi}{m}}{\frac{(k-E_i)\pi}{m}} \tag{25}$$

$$= -\frac{(k-E_i)^2\pi^2}{6m} + O\left((k-E_i)^4\right) \tag{26}$$

Therefore, we can approximate the centre of the distribution by $e^{-\frac{x^2\pi^2}{6m}}$ and so we can utilise the normalisation for the Gaussian $\int_{-\infty}^{\infty}\left|e^{-\frac{x^2\pi^2}{6m}}\right|^2 dx = \sqrt{\frac{3m}{\pi}}$ to approximate the normalisation for our distribution. □

Now we must prove that the tail probability decreases exponentially as we increase m.

Lemma 1.2. *Let δ be the probability of measuring an energy x outside of the confidence interval of $E_i \pm m$ when performing m-th B-spline boosted QPE. Then,*

$$\delta = O\left(\exp(-m)\right) \tag{27}$$

Proof. We define δ to be:

$$\delta := \sum_{|x-E_i|>m}\left|\frac{1}{\mathcal{N}}\left(\frac{\sin\frac{(x-E_i)\pi}{m}}{\frac{(x-E_i)\pi}{m}}\right)^m\right|^2 \tag{28}$$

Therefore, we can bound δ as follows:

$$\delta = \sum_{|x-E_i|>m}\left|\frac{1}{\mathcal{N}}\left(\frac{\sin\frac{(x-E_i)\pi}{m}}{\frac{(x-E_i)\pi}{m}}\right)^m\right|^2 \tag{29}$$

$$\leq \sum_{|x-E_i|>m}\frac{1}{\mathcal{N}^2}\left(\frac{1}{\frac{(x-E_i)\pi}{m}}\right)^{2m} \tag{30}$$

$$\leq \frac{2}{\mathcal{N}^2}\int_m^\infty\left(\frac{1}{\frac{x\pi}{m}}\right)^{2m} dx \tag{31}$$

$$\leq \frac{2}{\mathcal{N}^2}\frac{m}{2m-1}\pi^{-2m} \tag{32}$$

$$\tag{33}$$

Therefore, using Lemma 1.1 we get:

$$\delta = O(\exp(-m)) \tag{34}$$

□

Finally, we are ready to prove that the B-spline window function can boost QPE with the same asymptotic number of extra ancilla qubits as the Kaiser window function: $O\left(\log\log\frac{1}{\delta}\right)$.

Theorem 1. *To achieve a given probability δ of measuring energy outside of a given confidence interval ϵ (when using $O(\log\frac{1}{\delta})$-th B-spline boosted QPE), one must use $O(\log\frac{1}{\epsilon} + \log\log\frac{1}{\delta})$ ancillary qubits.*

Proof. Let l be the number of ancilla qubits used by B-spline boosted QPE. Using Lemma 1.2, we can achieve a normalised confidence interval of $\frac{2m}{2^l}$ where $m = O(\log\frac{1}{\delta})$. Hence, $\epsilon = \frac{O(\log\frac{1}{\delta})}{2^{l-1}}$. Therefore,

$$l = O\left(\log\left(\frac{1}{\epsilon}\log\frac{1}{\delta}\right)\right)$$

□