Nearly Optimal Quantum Algorithm for Generating the Ground State of a Free Quantum Field Theory

Mohsen Bagherimehrab,[1,2,3,*] Yuval R. Sanders,[4,5] Dominic W. Berry,[4] Gavin K. Brennen,[4,6] and Barry C. Sanders[1]

[1] *Institute for Quantum Science and Technology, University of Calgary, Alberta, Canada*
[2] *Chemical Physics Theory Group, Department of Chemistry, University of Toronto, Ontario, Canada*
[3] *Department of Computer Science, University of Toronto, Ontario, Canada*
[4] *Department of Physics and Astronomy, Macquarie University, New South Wales, Australia*
[5] *Centre for Quantum Software and Information, University of Technology Sydney, New South Wales, Australia*
[6] *Centre of Excellence in Engineered Quantum Systems, Macquarie University, New South Wales, Australia*

(Received 30 December 2021; accepted 26 May 2022; published 28 June 2022)

We devise a quasilinear quantum algorithm for generating an approximation for the ground state of a quantum field theory (QFT). Our quantum algorithm delivers a superquadratic speedup over the state-of-the-art quantum algorithm for ground-state generation, overcomes the ground-state-generation bottleneck of the prior approach and is optimal up to a polylogarithmic factor. Specifically, we establish two quantum algorithms—Fourier-based and wavelet-based—to generate the ground state of a free massive scalar bosonic QFT with gate complexity quasilinear in the number of discretized QFT modes. The Fourier-based algorithm is limited to translationally invariant QFTs. Numerical simulations show that the wavelet-based algorithm successfully yields the ground state for a QFT with broken translational invariance. Furthermore, the cost of preparing particle excitations in the wavelet approach is independent of the energy scale. Our algorithms require a routine for generating one-dimensional Gaussian (1DG) states. We replace the standard method for 1DG-state generation, which requires the quantum computer to perform lots of costly arithmetic, with a novel method based on inequality testing that significantly reduces the need for arithmetic. Our method for 1DG-state generation is generic and could be extended to preparing states whose amplitudes can be computed on the fly by a quantum computer.

DOI: 10.1103/PRXQuantum.3.020364

I. INTRODUCTION

Quantum algorithms for simulating a quantum field theory (QFT) comprise three main steps: generating an initial state, simulating time evolution, and measuring observables, with the initial-state generation being the most expensive step [1,2]. The conventional approach to generating the initial state for simulating a QFT, particularly for simulating particle scattering, is first to generate the ground state of the free (i.e., noninteracting) field theory. Then prepare free wavepackets, which refers to spatially localized noninteracting particles and, finally, turn on the field interaction adiabatically [1–3]. An alternative approach is to avoid adiabatic evolution in initial-state preparation, which is used in Ref. [4] to remove the state preparation as the bottleneck of simulating fermionic QFTs.

Several works have also developed variational [5,6] and stochastic [7,8] methods for simulating QFTs that avoid the need to prepare the full quantum state. These methods, however, are not purely quantum but rather are hybrid quantum classical, which are more suitable for simulation on near-term quantum computers. Despite the advances in simulating QFTs and development of recent algorithmic techniques for preparing quantum states [9–11], state preparation remains the computationally expensive step for simulating a certain class of bosonic QFTs called the massive scalar bosonic QFTs. In particular, preparing the free-field ground state is the most expensive part in one or two spatial dimensions, and is the second-most-expensive part for simulating these theories in three spatial dimensions [1,2].

In this paper, we establish a quasilinear quantum algorithm for generating an approximation for the ground

*mohsen.bagherimehrab@utoronto.ca

state of a massive scalar-bosonic free QFT. Our algorithm is optimal, up to a polylogarithmic factor, provides a superquadratic speedup over the best prior quantum algorithm for ground-state generation and overcomes the ground-state-generation bottleneck.

Specifically, we develop two quantum algorithms, one Fourier-based and the other wavelet-based, to generate the free-field ground state with gate complexity quasilinear in the number of discretized QFT modes. For the case of broken translational invariance, e.g., due to mass defects [12], the Fourier-based algorithm is inappropriate. We show, by numerical simulation, that the wavelet-based algorithm successfully yields an approximation for the free-field ground state with a quasilinear gate complexity. Our algorithms require a routine for preparing one-dimensional Gaussian (1DG) states, which are required for preparing multidimensional Gaussian states. The standard method for generating a 1DG state is based on Zalka-Grover-Rudolph state preparation [13,14], which requires the quantum computer to perform costly arithmetic operations. We replace this method with a novel method based on inequality testing [15] that significantly reduces the need for arithmetic.

A. Nontechnical background

To simulate a QFT, we first need to discretize it [2]. The discretization is needed for regulating infinite-dimensional Hilbert spaces involved in the continuum field theory. Once discretized, the QFT becomes a many-body quantum system whose time evolution can be efficiently simulated on a quantum computer [2,16,17]. The two other steps of a full quantum simulation, namely initial-state generation and measurement, strongly depend on which QFT is being simulated and must be analyzed on a case-by-case basis [2, p. 1017]. Two approaches are used to discretize a continuum massive scalar-bosonic QFT: lattice-based and wavelet-based approaches. The conventional lattice-based approach is used in the seminal quantum algorithm [1,2], and the wavelet-based approach is used in the subsequent quantum algorithm [18]. By discretizing the QFT, the free-field ground state becomes an N-dimensional Gaussian (NDG) state, where N is the number of modes in the discretized QFT.

A method proposed in Ref. [1] to generate a multidimensional Gaussian state is using the Kitaev-Webb method [19]. This method has three main steps to generate an NDG state on a quantum computer. The first step is to compute the LDL matrix decomposition of the Gaussian state's inverse-covariance matrix (ICM) by a classical computation; note that any Gaussian state is fully described by a covariance matrix or its inverse. The second step is to prepare N different 1DG states where the standard deviation of each 1DG state is obtained from a diagonal element of the diagonal matrix in the LDL decomposition. The last step is to perform a basis transformation based on the LDL matrix decomposition that maps the previously generated state, a Gaussian state with a diagonal ICM, to a Gaussian state with the desired ICM.

The Kitaev-Webb method [19] for preparing a 1DG state is an application of the standard state-preparation method by Zalka [13], Grover and Rudolph [14]. In this method for preparing a 1DG state on a quantum register, the continuous state is approximated by a discrete 1DG state over a one-dimensional lattice with unit spacing. Then a recursive description of the approximated 1DG state is used to generate the approximate state that requires the quantum computer to perform a controlled rotation for each qubit of the quantum register. For each controlled rotation, the quantum computer needs to coherently compute the rotation angle by a large amount of coherent arithmetic on the quantum computer.

The space and time complexities of the Kitaev-Webb method, both for 1DG- and NDG-state generation, were not analyzed in the original paper [19]. However, authors of Ref. [1] state that the method's time complexity for NDG-state generation is dominated by the classical complexity of the LDL matrix decomposition, which is $\widetilde{\mathcal{O}}(N^{2.373})$ [20] if we use Coppersmith-Winograd-style matrix multiplication; here $\widetilde{\mathcal{O}}$ suppresses logarithmic factors. There is also a quantum complexity of $\widetilde{\mathcal{O}}(N^2)$ to perform the basis transformation needed to yield the Gaussian state with the desired ICM. This quantum complexity is effectively the cost of performing an in-place matrix-vector multiplication.

B. Overview of methods and results

Our overall approach to reducing the time complexity for ground-state generation is to exploit known sparsity properties of the matrices involved. This approach allows us to reduce both classical and quantum time complexities by replacing dense matrix operations with sparse ones. In our wavelet-based approach, for example, we reduce the classical cost of the LDL matrix decomposition by approximating the ground-state ICM with a matrix containing $\mathcal{O}(N \log N)$ nonzero entries. Similarly, we reduce the quantum cost by replacing the general approach for in-place matrix-vector multiplication with a sparse approach. In our Fourier-based approach, we avoid even these sparse-approach optimizations by exploiting the translational invariance presumed by Jordan-Lee-Preskill [1,2] to eliminate the need for classical LDL decomposition. We replace the quantum in-place matrix-vector multiplication with a coherent fast Fourier transform.

Specifically, in our Fourier-based algorithm, we exploit the translational invariance of the free QFT to reduce the cost of ground-state generation from $\widetilde{\mathcal{O}}(N^{2.373})$ down to $\widetilde{\mathcal{O}}(N)$. We discretize the continuum QFT in a fixed-scale basis and utilize the circulant structure of the ground state's

ICM to compute its eigenvalues by a discrete Fourier transform (DFT). We then generate a NDG with a diagonal ICM whose diagonals are the eigenvalues by preparing N different 1DG states. Finally, we transform this state into the ground state by a basis transformation.

A choice for basis transformation is to execute a DFT on a quantum computer by reversible arithmetic operations along the lines of Ref. [21]. The problem with this basis transformation is that the resulting representation for the Gaussian state requires us to have complex-valued coordinates. We avoid the complex numbers required by the DFT and instead use a discrete Hartley transform (DHT) that involves only real numbers [22, Theorem 1]. The real, symmetric, and circulant properties of the ground-state ICM guarantees that by performing a DHT, we obtain the desired Gaussian state. Analogous to the quantum fast Fourier transform in Ref. [21], we construct a quantum fast Hartley transform (QFHT) algorithm with a quasilinear gate complexity.

In the wavelet-based algorithm, we discretize the continuum free QFT in a multiscale wavelet basis. The ICM of the ground state in this basis has many elements that have an exponentially close-to-zero value. We truncate this matrix by replacing the near-zero elements with exactly zero. This truncation introduces a systematic error that distorts the ground state of the discretized theory. We show that not only the truncated ICM remains a positive-definite matrix, which is required for the ground state to be a Gaussian state, but also the infidelity between the Gaussian state with the truncated ICM and the free-field ground state is within the prespecified error tolerance for preparing the ground state.

By the truncation, the ICM becomes a sparse matrix with a particular structure known as the "fingerlike" sparsity structure [23]; see Fig. 1. A matrix with this structure has a number of nonzero elements that is quasilinear in the dimension of the matrix. In order to exploit the fingerlike sparse structure, we replace the LDL decomposition in the Kitaev-Webb method [19] with the UDU decomposition. By exploiting the sparsity structure, we perform the UDU matrix decomposition of the truncated ICM in a quasilinear time.

The ground state in the wavelet-based approach is generated as follows. First we prepare an NDG state whose ICM is the diagonal matrix in the UDU decomposition. We then transform this state into the ground state by performing a quantum shear transform (QST) on a quantum computer. The shear-transform matrix has the same sparse structure as the truncated ICM in our UDU decomposition. We exploit this sparse structure and perform the needed QST with a quasilinear gate complexity.

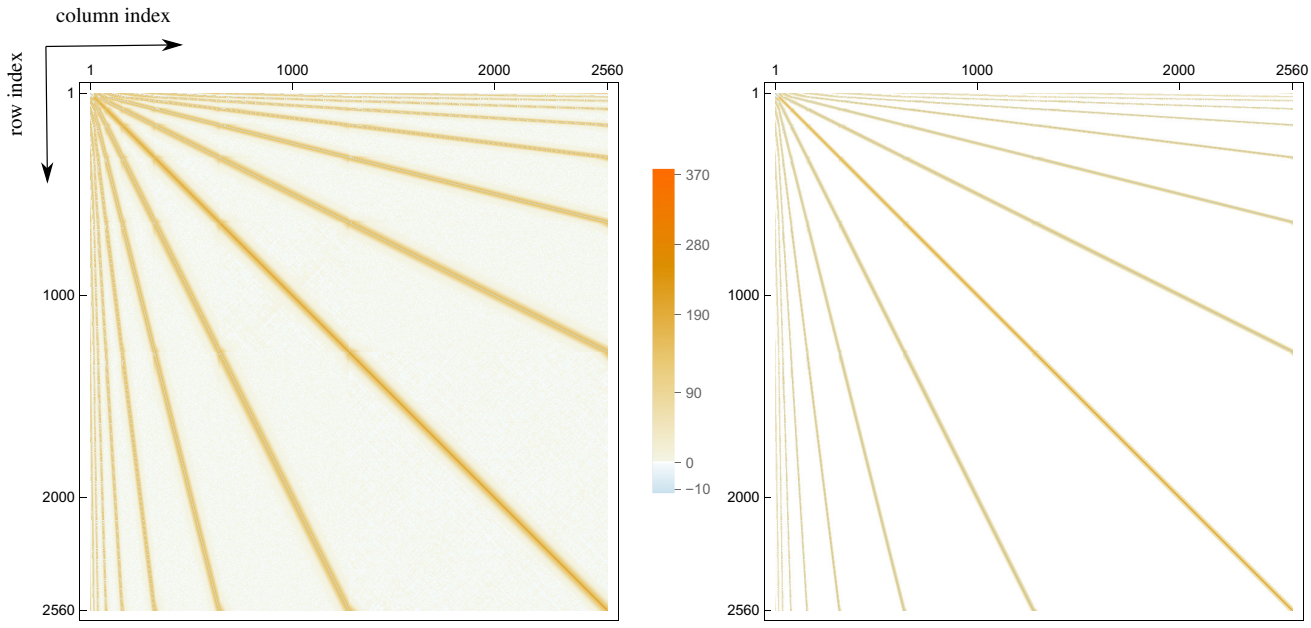

FIG. 1. Visual representation of the exact (left) and approximate ICM (right) for the ground state of the discretized free QFT in a multiscale wavelet basis. The column (row) index of the visualized matrix is ordered from left (top) to right (bottom), just as the typical format of a matrix. The values of matrix elements are represented by different colors with the shown color scale. The number of modes for the discretized QFT is $N = 2560$, free mass is $m_0 = 1$, and the Daubechies wavelet with index $\mathcal{K} = 3$ is used. The exact ICM is a quasisparse matrix, meaning most of its elements are nearly zero. Elements of the exact matrix with a magnitude less than 10^{-8} are replaced with zero to obtain the approximate ICM. The approximate ICM has a specific sparsity structure known as the "fingerlike" sparsity structure.

Our algorithms require a routine for generating an approximation to a 1DG state, and we devise two methods to perform this task. Our first method is based on the Kitaev-Webb method [19]. In our method, we approximate the 1DG state over a finite lattice with a nonunit lattice spacing. We choose the lattice spacing and the number of lattice points in terms of the 1DG-state standard deviation and an error tolerance for the generated state. The Kitaev-Webb method is restricted to 1DG states with extremely large standard deviations. Our method does not have this restriction and can generate a 1DG state with any standard deviation. We show that space and time complexities for preparing an approximation to each 1DG state required for generating the ground state are both logarithmic in the number of modes in the discretized QFT.

A key contributor to gate complexity for the Kitaev-Webb method is the need to perform a large amount of coherent arithmetic. Their method requires performing a sequence of controlled rotations for every qubit of the quantum register used for preparing a Gaussian state. For each controlled rotation, the quantum computer needs to coherently compute the third Jacobi θ function twice, perform a division, and compute one square root and one arccosine function. In contrast, we provide a significantly improved state-preparation technique based on inequality testing [15], where the most complicated computation needed is a single exponential. We go beyond the inequality-testing method in Ref. [15] by showing how to prepare a state with mostly small amplitudes except for a peak in a known location. We exploit the known location in our method and perform a single step of amplitude amplification [24] rather than multiple steps of amplitude amplification required for preparing a general state by inequality testing [15].

We perform a numerical study to justify why the wavelet-based approach could be preferred over the Fourier-based approach for QFTs with broken translational invariance. We consider a simple case where the translational invariance is broken due to a mass defect [12]. The Fourier-based approach is not applicable in this case because the ground-state ICM cannot be diagonalized by a discrete Fourier transform. However, our numerical experiment demonstrates that the wavelet-based approach accommodates such QFT with broken translational invariance. Specifically, the fingerlike sparse structure of the ground-state ICM is not affected by the mass defect, thereby the wavelet-based algorithm successfully yields an approximate ground state with a quasilinear gate complexity.

We go beyond ground-state generation and construct procedures for preparing free-field particle states in the Fourier- and wavelet-based approaches. We show that, unlike the Fourier approach, the wavelet approach enables preparing particle states at different energy scales without an additional cost required for the Fourier approach. Specifically, we show that preparing a free-field single-particle state at a given scale is more expensive than preparing the same state in the wavelet approach.

C. Organization

Our paper is organized as follows. We begin, in Sec. II, by elaborating the key background pertinent to subsequent sections. In this section, we review wavelet bases, various methods for discretizing a continuum QFT, a standard approach for generating a Gaussian state and a computation model used for analyzing an algorithm's time complexity. Next we describe our approach for generating the free-field ground state in Sec. III where we discuss our model for describing a scalar QFT, a metric that we use to analyze our algorithms' time complexity and our methods for generating the free-field ground state.

We then present our results in Sec. IV. In this section, we construct our ground-state-generation algorithms, analyze their classical and quantum time complexities, and show that our algorithms are optimal up to polylogarithmic factors. Furthermore, we determine the space required to represent the free-field ground state and compare the Fourier- and wavelet-based algorithms for a QFT with broken translational invariance and for generating states beyond the free-field ground state. We finally discuss our results in Sec. V and conclude in Sec. VI.

II. BACKGROUND

This section covers the key background pertinent to subsequent sections. We begin by describing wavelet bases in Sec. II A. Next, in Sec. II B, we discuss different approaches for discretizing a continuum QFT. Then we review the Kitaev-Webb method for generating a Gaussian state in Sec. II C. Finally, in Sec. II D, we review the quantum random-access machine (QRAM) model for computation (not to be confused with quantum random-access memory [25]).

A. Wavelet bases

Here we briefly review wavelet bases and define terms frequently used in this paper. For a detailed description, we refer to Ref. [26, Sec. 2.1] and Ref. [27,28]. We begin by explaining salient features of wavelet bases. Then we describe a fixed-scale wavelet basis followed by a multiscale wavelet basis.

The wavelet bases constitute an orthonormal basis for the Hilbert space $\mathscr{L}^2(\mathbb{R})$ of square-integrable functions on $\mathbb{R}$. A wavelet basis is defined in terms of two functions: *scaling* $s(x)$ and *wavelet* $w(x)$ functions. Here we focus on Daubechies $\mathcal{K}$ wavelets and refer to them as db$\mathcal{K}$ wavelets. The index $\mathcal{K} \in \mathbb{Z}^+$ specifies the number of vanishing moments of $w(x)$. The scaling and wavelet functions become smoother and less localized by increasing $\mathcal{K}$; see Fig. 2.

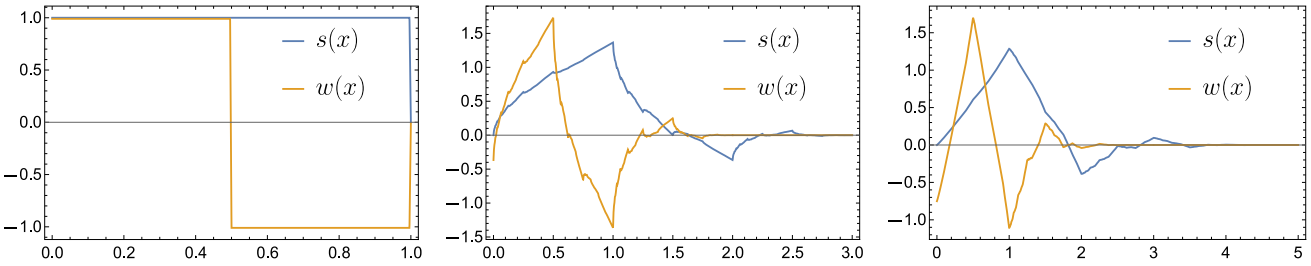

FIG. 2. From left to right: scaling $s(x)$ and wavelet $w(x)$ functions for the db$\mathcal{K}$ wavelet with index $\mathcal{K} = 1, 2, 3$; db1 wavelet is identical to Haar wavelet. The support of these functions is $[0, 2\mathcal{K} - 1]$. The $s(x)$ and $w(x)$ for db3 wavelet have continuous first derivatives.

The db$\mathcal{K}$ scaling function is the solution of the linear equation

$$s(x) = \sqrt{2} \sum_{\ell=0}^{2\mathcal{K}-1} h_\ell \, s(2x - \ell), \tag{1}$$

where, for a unique solution, $s(x)$ is normalized to have a unit area. The $2\mathcal{K}$ real coefficients $\{h_0, \ldots, h_{2\mathcal{K}-1}\}$ are called the *low-pass filter* coefficients. These coefficients uniquely determine a wavelet basis, and several methods are available to compute their numerical values; see Appendix B. Given $s(x)$, the wavelet function is

$$\begin{aligned} w(x) &:= \sqrt{2} \sum_{\ell=0}^{2\mathcal{K}-1} g_\ell \, s(2x - \ell), \\ g_\ell &:= (-)^\ell h_{2\mathcal{K}-1-\ell}, \end{aligned} \tag{2}$$

and $\{g_0, \ldots, g_{2\mathcal{K}-1}\}$ are called the *high-pass filter* coefficients. The scaling and wavelet functions at scale $k \in \mathbb{Z}$ are defined as

$$\begin{aligned} s_\ell^{(k)}(x) &:= \sqrt{2^k} \, s\left(2^k x - \ell\right), \\ w_\ell^{(k)}(x) &:= \sqrt{2^k} \, w\left(2^k x - \ell\right). \end{aligned} \tag{3}$$

These functions are orthonormal and have support on $\left[\ell/2^k, (\ell + 2\mathcal{K} - 1)/2^k\right]$. For convenience, we henceforth use the term "scale" to refer to the parameter k, so higher (lower) scale means larger (smaller) k.

The scaling function at a fixed scale $k \in \mathbb{Z}$ and its integer translation span a subspace $\mathcal{S}_k$ of $\mathscr{L}^2(\mathbb{R})$. We refer to this subspace as the *scale subspace* of $\mathscr{L}^2(\mathbb{R})$. Note that $\mathscr{L}^2(\mathbb{R}) \cong \lim_{k\to\infty} \mathcal{S}_k$. A fixed-scale wavelet basis is an orthonormal basis whose basis vectors are comprised of a scaling function and its integer translations at a fixed scale.

The wavelet function at a fixed scale k and its integer translations span a subspace $\mathcal{W}_k$ of $\mathscr{L}^2(\mathbb{R})$, which we call the *wavelet subspace*. The wavelet subspace $\mathcal{W}_k$ is the orthogonal complement of $\mathcal{S}_k$ in $\mathcal{S}_{k+1}$, i.e., $\mathcal{S}_{k+1} \cong \mathcal{S}_k \oplus \mathcal{W}_k$. This property of the scale and wavelet subspaces leads to a multiscale decomposition of $\mathscr{L}^2(\mathbb{R})$ through the relation $\mathscr{L}^2(\mathbb{R}) \cong \mathcal{S}_{s_0} \bigoplus_{r=s_0}^{\infty} \mathcal{W}_r$ [18], where s_0 is coarsest scale. The scaling function and its integer translations at a fixed scale along with the wavelet function and its integer translations at all finer scales construct an orthonormal basis for $\mathscr{L}^2(\mathbb{R})$, which we call the multiscale wavelet basis.

B. Discretization of a continuum QFT

In this subsection, we discuss different approaches for discretizing a continuum QFT. First we explain the common approach of discretizing space as a regular lattice. Then we describe the alternative wavelet approach for discretizing a QFT. For simplicity, we consider quantum fields in one spatial dimension. These discretizations can be extended to higher dimensions using known techniques [18,29].

We begin by describing a massive scalar bosonic free QFT using a Hamiltonian formalism, which is typical for quantum simulation of a QFT [2,30]. The time-independent Hamiltonian for a massive scalar bosonic free QFT with bare mass m_0, confined to one spatial dimension, is

$$\hat{H}_{\text{free}} := \frac{1}{2} \int_{\mathbb{R}} dx \left[\hat{\Pi}^2(x) + \left(\nabla \hat{\Phi}(x)\right)^2 + m_0^2 \hat{\Phi}^2(x) \right], \tag{4}$$

with x a position and with field operator $\hat{\Phi}(x)$ and conjugate momentum $\hat{\Pi}(x)$ satisfying

$$\begin{aligned} \left[\hat{\Phi}(x), \hat{\Pi}(y)\right] &= \mathrm{i}\delta(x - y)\mathbb{1}, \\ \left[\hat{\Phi}(x), \hat{\Phi}(y)\right] &= \left[\hat{\Pi}(x), \hat{\Pi}(y)\right] = 0. \end{aligned} \tag{5}$$

By QFT conventions, we fix constants $c \equiv 1 \equiv \hbar$. Discretizing QFT is achieved by replacing the spatial continuum with a hypercubic lattice of "locations" with finite lattice spacing $a \in \mathbb{R}^+$. For a 1D scalar field over $\mathbb{R}$, the discrete domain is $\Omega := a\mathbb{Z}$, and discrete field operators over the lattice satisfy $\left[\hat{\Phi}(x), \hat{\Pi}(y)\right] = \mathrm{i}\delta_{x,y}\mathbb{1}/a$, and all other commutators are zero.

For lattice QFT [31], the Laplacian in Eq. (4) and integration measure are

$$\nabla^2\hat{\Phi}(x) \mapsto \nabla_a^2\hat{\Phi}(x) := \frac{1}{a^2}[\hat{\Phi}(x+a)+\hat{\Phi}(x-a)-2\hat{\Phi}(x)], \qquad \int_{\mathbb{R}} dx \mapsto \sum_{x\in\Omega} a, \tag{6}$$

respectively. Defining $\hat{\Phi}_\ell := \hat{\Phi}(\ell a)$ and $\hat{\Pi}_\ell := \hat{\Pi}(\ell a)$ for each integer ℓ, the discretized version of the free QFT, Eq. (4), is [32]

$$\begin{aligned}\hat{H}_a &:= \frac{a}{2}\sum_{x\in\Omega}\left[\hat{\Pi}^2(x)-\hat{\Phi}(x)\nabla_a^2\hat{\Phi}(x)+m_0^2\hat{\Phi}^2(x)\right]\\ &= \frac{a}{2}\sum_{\ell\in\mathbb{Z}}\hat{\Pi}_\ell^2+\frac{a}{2}\sum_{\ell,\ell'\in\mathbb{Z}}\hat{\Phi}_\ell K_{\ell\ell'}\hat{\Phi}_{\ell'},\end{aligned} \tag{7}$$

where

$$K_{\ell\ell'} := m_0^2\delta_{\ell\ell'}-\left(\delta_{\ell,\ell'+1}+\delta_{\ell,\ell'-1}-2\delta_{\ell\ell'}\right)/a^2, \tag{8}$$

is the coupling between the two localized ℓ and ℓ' modes.

The field $\hat{\Phi}(x)$ and conjugate $\hat{\Pi}(x)$ operators can be expressed in a wavelet basis by projections onto the scaling and wavelet functions as [18,29]

$$\begin{aligned}\hat{\Phi}^{(k)}_{s;\,\ell} &:= \int_{\mathbb{R}} dx\, s_\ell^{(k)}(x)\hat{\Phi}(x),\\ \hat{\Pi}^{(k)}_{s;\,\ell} &:= \int_{\mathbb{R}} dx\, s_\ell^{(k)}(x)\hat{\Pi}(x),\end{aligned} \tag{9}$$

$$\begin{aligned}\hat{\Phi}^{(r)}_{w;\,\ell} &:= \int_{\mathbb{R}} dx\, w_\ell^{(r)}(x)\hat{\Phi}(x),\\ \hat{\Pi}^{(r)}_{w;\,\ell} &:= \int_{\mathbb{R}} dx\, w_\ell^{(r)}(x)\hat{\Pi}(x) \quad \forall\, r\geq k.\end{aligned} \tag{10}$$

We refer to field operators with subscript "s" and "w" as scale-field and wavelet-field operators, respectively. These operators have a compact support, determined by the support of their associated wavelet or scaling functions, and satisfy a set of commutation relations analogous to those of Eq. (5), but with the Dirac δ replaced with the Kronecker δ [18, p. 3].

To discretize the free QFT in a fixed-scale basis, we project the continuum Hamiltonian (4) onto a scale subspace $\mathcal{S}_k$ of $\mathscr{L}^2(\mathbb{R})$ for some $k\in\mathbb{Z}^+$. By Eq. (9), we first project $\hat{\Phi}(x)$ and $\hat{\Pi}(x)$ onto this subspace. Substituting the projected field and momentum operators into Eq. (4), we obtain the expression

$$\hat{H}_s^{(k)} := \frac{1}{2}\sum_{\ell}\hat{\Pi}^{(k)}_{s;\ell}\hat{\Pi}^{(k)}_{s;\ell}+\frac{1}{2}\sum_{\ell\ell'}\hat{\Phi}^{(k)}_{s;\ell}K^{(k)}_{ss;\ell\ell'}\hat{\Phi}^{(k)}_{s;\ell'}, \tag{11}$$

for projected Hamiltonian onto a scale subspace $\mathcal{S}_k$. Here

$$K^{(k)}_{ss;\ell\ell'} := m_0^2\delta_{\ell\ell'}-4^k\Delta^{(2)}_{\ell'-\ell}, \tag{12}$$

are the coupling between different modes and $\Delta^{(2)}_{\ell'-\ell}$ are the coefficients

$$\Delta^{(n)}_{\ell-\ell'} := \int dx\, s_{\ell-\ell'}(x)\frac{d^n}{dx^n}xs(x) \quad \forall\, n\geq 1, \tag{13}$$

with $n=2$. We refer to these coefficients as the n-order *derivative overlaps* and use $\Delta_\ell := \Delta^{(2)}_\ell$ for simplicity.

The multiscale wavelet discretization of the free QFT is obtained by projecting $\hat{\Phi}(x)$ and $\hat{\Pi}(x)$ onto a multiscale basis and substituting the projected operators into Eq. (4). In this case, the projected Hamiltonian is

$$\begin{aligned}\hat{H}_w := \hat{H}_s^{(s_0)} &+ \frac{1}{2}\sum_{\ell,r\geq s_0}\hat{\Pi}^{(r)}_{w;\ell}\hat{\Pi}^{(r)}_{w;\ell}+\frac{1}{2}\sum_{\ell\ell',rr'}\hat{\Phi}^{(r)}_{w;\ell}K^{(r,r')}_{ww;\ell\ell'}\hat{\Phi}^{(r')}_{w;\ell'}\\ &+\frac{1}{2}\sum_{\ell\ell',r\geq s_0}\hat{\Phi}^{(s_0)}_{s;\ell}K^{(s_0,r)}_{sw;\ell\ell'}\hat{\Phi}^{(r)}_{w;\ell'},\end{aligned} \tag{14}$$

where $K^{(r,r')}_{ww;\ell\ell'}$ are coupling between the wavelet fields at scales r and r', and $K^{(s_0,r)}_{sw;\ell\ell'}$ are coupling between the scale fields at scale s_0 and wavelet fields at scale r. These couplings are systematically computed from the derivative overlaps, Eq. (13) [29,33].

Wavelet discretization for the momentum operator $\hat{P} := -\int_{\mathbb{R}} dx\,\hat{\Pi}(x)\nabla\hat{\Phi}(x)$ of the free QFT [29, p. 6] is similar to wavelet discretization for Hamiltonian. Therefore, we consider only fixed-scale discretization of $\hat{P}$. In this discretization, $\hat{P}$ is projected onto a scale subspace $\mathcal{S}_k$. The projected momentum operator is

$$\hat{P}_s^{(k)} := -\sum_{\ell,\ell'}\hat{\Pi}^{(k)}_{s;\ell}P^{(k)}_{\ell\ell'}\hat{\Phi}^{(k)}_{s;\ell'}, \quad P^{(k)}_{\ell\ell'} := 2^k\Delta^{(1)}_{\ell'-\ell}, \tag{15}$$

where $\Delta^{(1)}_{\ell'-\ell}$ are the derivative overlaps in Eq. (13) with $n=1$. We use this expression in Sec. III B 1 to obtain the number of modes of the discretized free QFT in a fixed-scale basis.

C. Kitaev-Webb method for Gaussian-state generation

Here we review the Kitaev-Webb method for generating an approximation of a multidimensional continuous Gaussian pure state on a quantum register [19]. First we define a continuous Gaussian pure state and set our notations for particular Gaussian states. Then we explain the main idea of the Kitaev-Webb method and proceed with describing details of the method.

We begin by defining a continuous Gaussian pure state in Definition 1. Throughout this paper, we use this definition when we refer to a continuous Gaussian pure state.

Definition 1: (Continuous Gaussian pure state.) Let N be a positive integer, $\mathbf{A}$ be a real-valued N-by-N symmetric

positive-definite matrix and

$$p_N(\mathbf{A};\mathbf{x}) := \sqrt{\frac{\det \mathbf{A}}{(2\pi)^N}}\, e^{-\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}/2}, \tag{16}$$

be the probability density function of a continuous N-dimensional Gaussian (NDG) distribution, with the ICM $\mathbf{A}$, for a random variable $\mathbf{x} := (x_0, x_1, \ldots, x_{N-1}) \in \mathbb{R}^N$. We define the pure state

$$|\mathrm{G}_N(\mathbf{A})\rangle := \int_{\mathbb{R}^N} d^N\mathbf{x} \sqrt{p_N(\mathbf{A};\mathbf{x})}\, |\mathbf{x}\rangle\,, \tag{17}$$

where

$$|\mathbf{x}\rangle := |x_0\rangle \otimes \cdots \otimes |x_{N-1}\rangle\,, \tag{18}$$

is a vector of distributions, as the continuous NDG pure state with the ICM $\mathbf{A}$.

We use a particular notation for 1DG pure states. If the Gaussian distribution in Definition 1 is one dimensional with the standard deviation $\sigma \in \mathbb{R}^+$ and the mean value $\mu \in \mathbb{R}^+$, then we refer to the state

$$\begin{aligned} |G(\sigma,\mu)\rangle &:= \frac{1}{\mathcal{N}} \int_{\mathbb{R}} dx\, e^{-\frac{(x-\mu)^2}{4\sigma^2}}\, |x\rangle\,, \\ \mathcal{N}^2 &:= \int_{\mathbb{R}} dx\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \sigma\sqrt{2\pi}, \end{aligned} \tag{19}$$

for $|x\rangle$ a continuously parameterized position state, as the continuous 1DG pure state with standard deviation σ and mean value μ. For simplicity, we denote $|G(\sigma,0)\rangle$ by $|G(\sigma)\rangle$.

We now present a high-level description of Kitaev's and Webb's method for generating a multidimensional Gaussian state. Their method's main idea is first to prepare a set of independent 1DG states and then perform a basis transformation to produce the desired Gaussian state. The parameters needed for preparing the 1DG states and the basis transformation are outputs of a classical algorithm that computes the LDL decomposition of the Gaussian state's ICM $\mathbf{A}$. Specifically, the classical algorithm returns a diagonal matrix $\mathbf{D}$ and a lower unit-triangular matrix $\mathbf{L}$ such that $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}$. Diagonal of $\mathbf{D}$ are parameters needed for preparing the 1DG states, and off diagonals of $\mathbf{L}$ are parameters needed for the basis transformation.

Their method for generating an NDG state with ICM $\mathbf{A}$ can be described by three main steps: (1) classically compute $\mathbf{L}$ and $\mathbf{D}$ in the LDL decomposition of the ICM $\mathbf{A}$; (2) generate an approximation for $|G_N(\mathbf{D})\rangle$; and (3) implement the basis transformation $|\mathbf{x}\rangle \mapsto |\mathbf{S}\mathbf{x}\rangle$, where $\mathbf{S}$ is inverse transpose of $\mathbf{L}$ and $|\mathbf{x}\rangle$, Eq. (18), is the basis state. The basis transformation in the last step is implemented by storing off-diagonal elements of $\mathbf{S}$ on ancillary quantum registers and performing reversible operations on a quantum computer. The state $|G_N(\mathbf{D})\rangle$ in the second step is generated by preparing N independent 1DG states with standard deviations $\sigma_\ell := 1/\sqrt{D_{\ell\ell}}$ for $\ell \in \{0,\ldots,N-1\}$.

We now describe the Kitaev-Webb method for generating 1DG states. To generate the continuous 1DG state $|G(\sigma,\mu)\rangle$, Eq. (19), on a quantum register, it is first approximated by the discrete 1DG state

$$\begin{aligned} \left|\tilde{\mathrm{G}}_{\mathrm{KW}}(\sigma,\mu)\right\rangle &:= \sum_{i\in\mathbb{Z}} \tilde{G}_{\mathrm{KW}}(\sigma,\mu;\ \mathrm{i})\, |i\rangle\,, \\ \tilde{G}_{\mathrm{KW}}(\sigma,\mu;\ i) &:= \frac{1}{\sqrt{f_{\mathrm{KW}}(\sigma,\mu)}} e^{-\frac{(i-\mu)^2}{4\sigma^2}}, \\ f_{\mathrm{KW}}(\sigma,\mu) &:= \sum_{i\in\mathbb{Z}} e^{-\frac{(i-\mu)^2}{2\sigma^2}}, \end{aligned} \tag{20}$$

over the 1D infinite lattice with unit lattice spacing. This discrete 1DG state is again approximated by the state

$$\begin{aligned} |\xi(\sigma,\mu,m)\rangle &:= \sum_{i=0}^{2^m-1} \xi(\sigma,\mu,m;\ i)\, |i\rangle\,, \\ \xi^2(\sigma,\mu,m;\ i) &:= \sum_{j\in\mathbb{Z}} \tilde{G}^2_{\mathrm{KW}}(\sigma,\mu;\ i+j2^m). \end{aligned} \tag{21}$$

This quantum state is used as an approximation for $|G(\sigma,\mu)\rangle$, Eq. (19), in the Kitaev-Webb method to be generated on a quantum register. The key point of this method is to employ the recursive decomposition

$$\begin{aligned} |\xi(\sigma,\mu,m)\rangle &= \left|\xi\left(\frac{\sigma}{2},\frac{\mu}{2},m-1\right)\right\rangle \otimes \cos\theta\, |0\rangle \\ &\quad + \left|\xi\left(\frac{\sigma}{2},\frac{\mu-1}{2},m-1\right)\right\rangle \otimes \sin\theta\, |1\rangle\,, \\ \theta &:= \arccos\sqrt{\frac{f_{\mathrm{KW}}(\sigma/2,\mu/2)}{f_{\mathrm{KW}}(\sigma,\mu)}}, \end{aligned} \tag{22}$$

for generating the approximate 1DG state $|\xi(\sigma,\mu,m)\rangle$, Eq. (21).

By the recursive formula (22) and classical inputs $\sigma_0 := \sigma, \mu_0 := \mu$ and m, following recursive procedure is used to generate the approximate 1DG state: (1) compute $\theta/2\pi$, Eq. (22), and store it on an ancillary quantum register; (2) perform the single-qubit rotation $R(\theta) := \exp(-\mathrm{i}\theta Y)$ on the rightmost qubit, where Y is the Pauli-Y gate; (3) uncompute $\theta/2\pi$; (4) compute $\sigma_1 := \sigma_0/2$ and $\mu_1 := (\mu_0 - q_0)/2$, where $q_0 = 0$ if the state of the rotated qubit is $|0\rangle$ and $q_0 = 0$ if it is $|1\rangle$; and (5) generate the state $|\xi(\sigma_1,\mu_1,m-1)\rangle$, Eq. (22), on the remaining $m-1$ qubits.

The Kitaev-Webb method is restricted to 1DG states that possess an extremely large standard deviation. In Sec. III C 1 and Sec. IV D 1, we describe our methods for generating an approximation of a continuous 1DG state for any standard deviation.

D. Quantum random-access machine model for computation

An algorithm's time complexity depends on the model for computation. Here we review the quantum random-access machine model for computation introduced by Knill [34]. We use a variant of this model for analyzing our algorithms' time complexity. First we describe the architecture of the QRAM model. Then we discuss "primitive" operations for this model. Finally, we describe how an algorithm's time complexity is assessed in the QRAM model and discuss the computational complexity for computing some elementary functions.

The QRAM model is an extension of the classical RAM model that allows classical and quantum computations. We describe a simplified architecture of the QRAM model. For more details, we refer to Refs. [35–37]. In this model, as schematically illustrated in Fig. 3, a computer has a classical and a quantum processor that are, respectively, connected to classical and quantum registers. These processors work in a master-slave fashion, where the classical processor is the master that controls the quantum processor. A hybrid quantum-classical code is first compiled into the classical processor. The compiled code contains both classical and quantum instructions. The classical processor performs the classical instructions on classical registers and sends the quantum instructions to be performed by the quantum processor on quantum registers. The measurement results are sent back to the classical processor by the quantum processor. This process could be repeated multiple times depending on the code.

The classical and quantum processors in the QRAM model can perform only a restricted set of operations on their associated registers. These operations are called primitive operations, and a unit cost is assigned to each primitive in this model. As QRAM is an extension of the classical RAM, the QRAM's classical primitives are considered to be the same as the primitives in the classical RAM model, which are the following [38]: (1) basic arithmetic operations, i.e., addition, subtraction, multiplication, and division; (2) data-movement operations such as writing data from memory to classical registers and reading data from classical registers to memory; (3) Boolean logic operations such as AND and OR; and (4) flow-control operations such as calling a function or returning from a function. The classical primitives, except Boolean logic operations, are high-level operations. In practice, implementing high-level operations does not have the same cost in terms of low-level (i.e., bit-wise) operations.

Unlike classical primitives, which are high-level operations, quantum primitives in the QRAM model are low-level operations [34]. Specifically, quantum primitives in the QRAM model are quantum gates from a universal set of gates. In Sec. III A 3, we describe our alternative for quantum primitives that are high-level operations, similar to classical primitives.

The time complexity for a classical algorithm is determined by counting the number of classical primitives that need to be executed in the algorithm. Similarly, the common approach to analyzing a quantum algorithm's time complexity in the QRAM model is counting the number of quantum primitives. As quantum primitives in QRAM are quantum gates, the algorithm's gate complexity is typically used as a standard metric to cost out a quantum algorithm.

Finally, we state the time complexity for computing four elementary functions with respect to the classical primitives in the QRAM model: logarithm, square-root, inverse-square-root, and trigonometric functions. These functions are used in various subroutines of our ground-state-generation algorithms described in Sec. IV A. The time complexity for computing each of these functions was analyzed in terms of the time complexity for performing a multiplication in Ref. [39]. Multiplication is a primitive operation in the QRAM model and has a unit cost. Therefore, we list only time complexities for the elementary functions with respect to classical primitives in the QRAM model. The time complexity for computing square root or inverse square root of a number in this model is

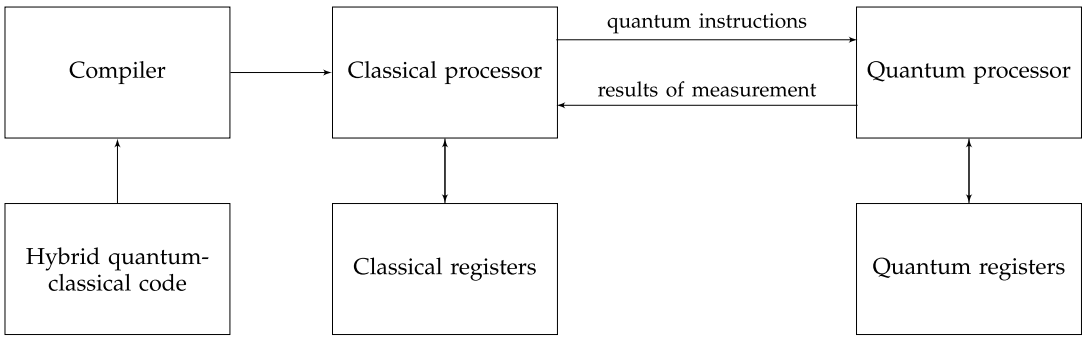

FIG. 3. Schematic description of the QRAM model. A hybrid code is compiled into the classical processor. The compiled code provides instructions to be performed by classical and quantum processors on their associated registers. The quantum processor sends the measurement results back to the classical processor.

$\mathcal{O}(1)$ [39, pp. 3–4], and the time complexity for computing a logarithm or trigonometric functions to precision p is $\mathcal{O}(\log p)$ [39, pp. 11, 14]. We use these complexities in Sec. IV E, where we analyze our algorithms' time complexity.

III. APPROACH

In this section, we present our approach for generating an approximate quantum-register representation for the ground state of a free massive scalar bosonic QFT, such that the quantum algorithm succeeds deterministically in a quasilinear time with respect to the number of modes for the discretized QFT. We begin in Sec. III A by discussing our model for describing a free massive scalar bosonic QFT. We introduce a client-server framework for simulating a QFT and discuss a metric to measure our algorithm's time complexity. Next we describe in Sec. III B the mathematical approach inherent in generating the free-field ground state. We explain how the mass, momentum cutoff, wavelet index, and error tolerance are used to discretize the continuum free-field Hamiltonian (4) in fixed- and multiscale wavelet bases from which the covariance-matrix description of the discretized QFT ground state is obtained. Finally, in Sec. III C, we describe our Fourier and wavelet-based method for ground-state generation and also our methods for generating one-dimensional Gaussian states.

A. Model

In this subsection, we discuss our model for describing a massive scalar bosonic QFT. We also introduce a client-server framework for simulating a QFT on a quantum computer and discuss the metric we use to measure our algorithms' time complexity. We begin by explaining our model for describing the QFT in Sec. III A 1. We then explain the framework in Sec. III A 2 followed by the metric for measuring an algorithm's runtime in Sec. III A 3.

1. Discretized quantum field theory

Here we explain our model for describing a massive scalar bosonic QFT. We describe our approach for discretizing a one-dimensional scalar bosonic field in fixed- and multiscale wavelet bases. Furthermore, we compare these discretizations with the Jordan-Lee-Preskill discretization, which is based on the conventional lattice approach [1].

In contrast to the usual approach for discretizing a QFT over a noncompact domain, such as the infinite real line for a one-dimensional theory, we discretize for the field on a finite interval of the real line with periodic boundary conditions. Mathematically, the finite interval with periodic boundary conditions can be treated as a circle domain. We denote the bare mass of the field theory by m_0, as in Eq. (4). We consider an ultraviolet momentum cutoff Λ for the field theory, meaning we ignore all field configurations with momentum higher than Λ. For convenience, we work in the normalized scale where the compact domain becomes the unit interval. In this case, the theory is on the unit interval with periodic boundaries, and all involved parameters such as mass and momentum cutoff become dimensionless parameters.

In fixed-scale discretization, we partition the unit interval into 2^k subintervals of length $1/2^k$. We choose the positive integer k such that $2^k \geq 2(2\mathcal{K}-1)$, where $\mathcal{K}$ is the wavelet index (Sec. II A). This choice is made because the smallest size admissible with periodic boundaries is $2(2\mathcal{K}-1)$ to ensure the scale function and its translations are orthogonal [33, p. 4]. We assign a discrete scale field, Eq. (9), to each of the $2\mathcal{K}-1$ cyclically consecutive subintervals according to the following averaging rule. Each discrete field is an average of the continuous field over an interval of length $(2\mathcal{K}-1)/2^k$ weighted by the db$\mathcal{K}$ scaling function at scale k [29, p. 7]. The discrete fields and their conjugate momenta satisfy the commutation relations analogous to those of Eq. (5) but with the Dirac δ replaced by the Kronecker δ. The momentum cutoff Λ in this discretization is proportional to the inverse of the subinterval's length.

In multiscale wavelet discretization, we partition the unit interval into 2^s subintervals of length $1/2^s$ at each scale, where $s \in \{s_0, s_0+1, \ldots, k-1\}$. Here k is an integer such that $2^k \geq 2(2\mathcal{K}-1)$ and s_0 is the smallest integer that satisfies this inequality. At the smallest scale s_0, we assign a scale field and a wavelet field, Eq. (9), to each of the $2\mathcal{K}-1$ cyclically consecutive subintervals according to the following averaging rule. Each scale (wavelet) field is an average of the continuous field over an interval of length $(2\mathcal{K}-1)/2^{s_0}$ weighted by the db$\mathcal{K}$ scaling (wavelet) function at scale s_0. At each other scale $s > s_0$, we assign a wavelet field to each of the $2\mathcal{K}-1$ cyclically consecutive subintervals. Each wavelet field is an average of the continuous field over an interval of length $(2\mathcal{K}-1)/2^s$ weighted by the db$\mathcal{K}$ wavelet function at scale s. The discrete fields and their conjugate momenta in a multiscale wavelet basis also satisfy the commutation relations analogous to those of Eq. (5) but with the Dirac δ replaced by a Kronecker δ [18, p. 3]. The momentum cutoff Λ in this discretization is proportional to the inverse of the subinterval's length at scale with the largest scale index s.

In the Jordan-Lee-Preskill approach [1], the conventional lattice discretization is used to discretize the scalar bosonic quantum field. The unit interval is approximated by a one-dimensional finite lattice with 2^k points and lattice spacing $1/2^k$ for some positive integer k. A discrete field is then assigned to each lattice point, where the discrete fields are samples of the continuous field at lattice points. In contrast, the discrete fields in the wavelet approach are an average of the continuous field over subintervals of the unit interval. These discrete fields have overlapping domains,

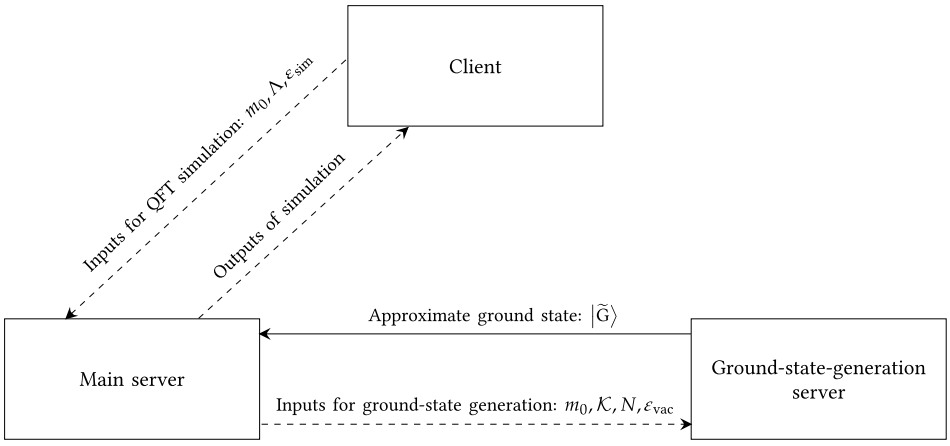

FIG. 4. A client-server framework for simulating a QFT. Dashed lines represent classical communication, and solid lines represent quantum communication. The classical inputs to the main and ground-state-generation servers are specified in Tables I and II, respectively, for simulating a massive scalar bosonic QFT. Output from the ground-state-generation server is an approximation for the free-field ground state.

whereas the domains of the discrete fields in the lattice approach, i.e., the lattice points, do not overlap.

2. Client-server framework for simulating a QFT

We now describe a framework for simulating a QFT. To make clear ground-state generation versus other aspects of a QFT simulation, we employ a framework comprising three components: a client, a main server, and a ground-state-generation server; see Fig. 4 for a schematic representation of the framework. We define each component of the framework and elucidate its relative task in simulating a QFT. Finally, we describe the information flow between the client and the two servers for simulating a massive scalar bosonic QFT.

A client is an agent who supplies the needed parameters for solving a computational problem to a server and accepts the solution. The client in our framework communicates only with the main server. She provides the required inputs for simulating a QFT to this server and accepts the solution, which is the simulation's outputs.

A server is a computer that provides a function or service to one or many clients; a server could also be a client to another server. The main server's task in our framework is to simulate the QFT specified by the client and deliver the simulation's output to the client. The main server chooses a particular basis for simulation to accomplish this task and delegates the ground-state-generation part of the simulation to the ground-state-generation server. The main server is, therefore, a client to the ground-state-generation server in our framework. The main server supplies the input parameters needed for generating the ground state of the QFT and accepts a quantum state, which is an approximation for the free-field ground state.

The ground-state-generation server in our framework is an auxiliary server whose task is to generate an approximation for the QFT ground state, which is represented in a particular basis by the main server, on a quantum register and deliver the generated state to the main server. We consider this auxiliary server in our framework to elucidate the input parameters needed for generating the ground state and separate the ground-state generation part of a QFT simulation from other parts of the full quantum simulation as ground-state generation is a bottleneck for the entire simulation. The main server then performs the simulation using the generated state by the ground-state-generation server. The main server may wish to perform the simulation on a different basis. In this case, the main server first executes a basis transformation on the generated state by the ground-state-generation server and then performs the simulation on a new basis.

We now discuss the information flow between the client and the two servers in the QFT-simulation framework. The client in this framework supplies the input parameters for simulating a QFT to the main server. The required input parameters are those that specify the Hamiltonian (or Lagrangian) describing the QFT, an error tolerance for output of simulation and a parameter specifying the energy at which the simulation is performed. The free mass m_0 is the only parameter that specifies the Hamiltonian of a free massive scalar bosonic QFT; see Eq. (4). We use an ultraviolet cutoff on the momentum of the free QFT particles as the parameter specifying the simulation energy. Table I specifies the required inputs that need to be supplied by the client to the main server for simulating a massive scalar bosonic free QFT.

The main server chooses a wavelet basis by selecting a wavelet index $\mathcal{K} \in \mathbb{Z}_{\geq 3}$ to perform the simulation. This server approximates the continuum field theory by a finite-mode discretized QFT in the db$\mathcal{K}$ wavelet basis and calculates the sufficient number of modes for the

TABLE I. Inputs from client to the main server.

Parameter	Type	Description
m_0	$\mathbb{R}^+$	Free QFT mass
Λ	$\mathbb{R}^+$	Ultraviolet cutoff on momentum of the free QFT particles
ε_{sim}	$(0, 1)$	Error tolerance for output of simulation

TABLE II. Inputs from the main server to the ground-state-generation server.

Parameter	Type	Description
m_0	$\mathbb{R}^+$	Free QFT mass
$\mathcal{K}$	$\mathbb{Z}_{\geq 3}$	Wavelet index
N	$\mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$	Number of modes in the discretized QFT
ε_{vac}	$(0, 1)$	Error tolerance for generating the discretized QFT ground state

discretized theory using the client's inputs. The main server then supplies the inputs specified in Table II to the ground-state-generation server.

The ground-state-generation server uses the inputs supplied by the main server to generate an approximation for the ground state of the discretized free QFT. This server then delivers the generated state to the main server for QFT simulation. Finally, the simulation outputs are provided to the client by the main server. The flow of information between the client and two servers is shown in Fig. 4.

3. Measure for time complexity

Here we describe how we assess an algorithm's time complexity. We begin by stating the metric that we use for the time complexity of an algorithm. Then we specify primitive operations for classical and quantum processors of the QRAM model described in Sec. II D. The chosen primitives are high-level operations, and we explain how to relate these primitives to low-level primitives. Finally, we discuss how our metric differs from common metrics for analyzing an algorithm's time complexity.

We use the number of primitive operations in an algorithm as a metric to quantify the algorithm's time complexity. That is to say, the number of classical primitives in an algorithm determines its classical complexity, and the number of quantum primitives determines the algorithm's quantum complexity. By this metric, the time complexity depends on the classical and quantum primitives; changing the set of primitives yields a different time complexity. Hence the set of primitive operations needs to be specified.

We now specify the classical and quantum primitives. We choose the classical primitives to be the same as the classical primitive operations in the QRAM model; see Sec. II D. Except for the flow-control operations (Sec. II D), we take the quantum primitives to be a quantum version of the classical primitives. We exclude the flow-control operations as quantum primitives because the quantum processor in the QRAM model is controlled by the classical processor. Specifically, we choose the following operations as quantum primitives in our time-complexity analysis for a quantum algorithm: (1) basic arithmetic operations on quantum registers; (2) data-movement operations on quantum registers, such as writing (preparing) classical data from memory into quantum registers and reading (measuring) data from quantum registers to memory; and (3) quantum logic gates such as the Hadamard, controlled NOT (CNOT), and Toffoli gates.

With our chosen quantum primitives, similar to the classical RAM model, the QRAM model encapsulates computers' core functionality, not their exact functionalities. For instance, addition and multiplication operations are each considered a single primitive operation for each processor in this model. In practice, however, a processor needs to execute more low-level operations—bit-wise operations for the classical processor and qubit-wise operations for quantum processor—to perform multiplication versus addition. By analyzing the cost of performing high-level primitives in terms of low-level operations, one can obtain an algorithm's time complexity with respect to low-level operations.

We comment that our approach for analyzing a quantum algorithm's time complexity is not common in the literature. As described in Sec. II D, the common approach to cost out a quantum algorithm is to count the number of low-level operations, i.e., quantum gates, in the algorithm [34,40]. In Sec. V, we discuss our algorithms' time complexity with respect to low-level operations.

B. Mathematics

This subsection describes the mathematical approach for generating the ground state of a massive scalar bosonic QFT. We begin in Sec. III B 1 by describing a procedure for approximating the ground state of the continuum QFT, Eq. (4), in a fixed-scale basis. Then we proceed with approximating the ground state in a multiscale wavelet basis in Sec. III B 2. Lastly, we explain our procedure for discretizing a continuous Gaussian pure state in Sec. III B 3.

1. Free-field ground state in a fixed-scale basis

Here we represent the ground state of the continuum free QFT, Eq. (4), in a fixed-scale wavelet basis. To this end, we discretize the continuum theory by projecting its Hamiltonian onto a fixed-scale subspace of $\mathscr{L}^2(\$)$. The ground state of the discretized QFT represents the free-field ground state in a fixed-scale basis. We explain how to select a sufficient number of modes N for the discretized QFT using the client inputs in Table II. Specifically, we establish sufficiency for N in terms of the momentum cutoff Λ, supplied by the client, such that the magnitude of mean momentum (expectation value of the momentum operator) for a single-particle state in the discretized QFT is no greater than Λ.

To represent the ground state of the free theory, Eq. (4), over the unit interval with periodic boundaries in a fixed-scale wavelet basis, we project the continuum Hamiltonian onto a scale subspace $\mathcal{S}_k$ of $\mathscr{L}^2(\$)$ for some integer k

such that $2^k \geq 2(2\mathcal{K}-1)$; see Sec. III A 1. The projected Hamiltonian has the form of the discrete Hamiltonian in Eq. (11) but with the coupling matrix

$$K^{(k)}_{ss;\,\ell\ell'} := m_0^2\delta_{\ell\ell'} - N^2\left(\Delta^{(2)}_{\ell'-\ell} + \Delta^{(2)}_{\ell'-(\ell+N)}\right), \quad N := 2^k, \tag{23}$$

which are matrix elements of $\mathbf{K}^{(k)}_{ss}$. Here N is the number of modes for the discretized QFT and $\Delta^{(2)}_{\ell'-\ell}$, Eq. (13), are the second-order derivative overlaps; the second term inside the parentheses comes from the periodic boundary condition. The projected Hamiltonian is quadratic in the field operators and their conjugate momenta akin to the discrete Hamiltonian in Eq. (11) and, therefore, its ground state is a Gaussian state. Specifically, the ground state of the projected Hamiltonian is

$$\left|G^{(k)}_{\text{scale}}\right\rangle := \left(\frac{\det \mathbf{A}^{(k)}_{ss}}{(2\pi)^N}\right)^{1/4} \int_{\mathbb{R}^N} d^N\boldsymbol{\phi}\, e^{-\frac{1}{4}\boldsymbol{\phi}^{\mathsf{T}}\mathbf{A}^{(k)}_{ss}\boldsymbol{\phi}}\,|\boldsymbol{\phi}\rangle\,,$$
$$\mathbf{A}^{(k)}_{ss} := \sqrt{\mathbf{K}^{(k)}_{ss}}, \tag{24}$$

where $\mathbf{A}^{(k)}_{ss}$ is the ground state's ICM as per Definition 1.

We now establish sufficiency for the number of modes N using the supplied momentum cutoff Λ by the client to calculate the sufficient N for the discretized QFT. We first project the momentum operator (Sec. II B) of the continuum QFT to the same scale subspace $\mathcal{S}_k$ that the continuum Hamiltonian is projected. Then we write an expression for a single-particle state whose mean momentum has the maximum magnitude $\bar{P}_{\max}$. Next we bound $\bar{P}_{\max}$ from above by Λ. The sufficient N saturates this bound. Proposition 2 provides the established sufficiency for N with respect to Λ and the largest first-order derivative overlap $\Delta^{(1)}_{\max} := \max_\ell|\Delta^{(1)}_\ell|$, for $\Delta^{(1)}_\ell$, Eq. (13), the first-order derivative overlaps.

Proposition 2: *For Λ the momentum cutoff and $\Delta^{(1)}_{\max}$ the largest first-order derivative overlap,*

$$N = \left\lfloor \frac{2\Lambda}{\Delta^{(1)}_{\max}} \right\rfloor, \tag{25}$$

modes suffices to guarantee that the mean momentum of a single-particle state in the discretized QFT is bounded from above by Λ.

Proof. Let $a^{(k)\dagger}_\ell$ be the creation operator constructed from the ℓ^{th} scale-field operator (9) and its conjugate momentum [41]. Acting this operator on the ground state in Eq. (24) creates a single-particle state with zero-mean momentum whose wave function is localized in a compact space of size equal to the support of $s^{(k)}_\ell$ [18]. Single-particle states with finite mean momentum can be created from a superposition of two zero-mean-momentum single-particle states as

$$\left|\Psi^{(k)}_{\ell\ell'}\right\rangle := \left(\alpha a^{(k)\dagger}_\ell + \beta a^{(k)\dagger}_{\ell'}\right)\left|G^{(k)}_{\text{scale}}\right\rangle, \tag{26}$$

with $\alpha,\beta\in\mathbb{C}$ such that $|\alpha|^2+|\beta|^2=1$. The expectation value of the projected momentum operator (15), i.e., the mean momentum, for this state is

$$\bar{P} := \langle\Psi^{(k)}_{\ell\ell'}|\hat{P}^{(k)}|\Psi^{(k)}_{\ell\ell'}\rangle = P^{(k)}_{\ell\ell'}\,\text{Im}\left(\alpha\beta^*\right). \tag{27}$$

The magnitude of this expression is maximized for $\alpha = \pm\mathrm{i}\beta = 1/\sqrt{2}$. By this equation and the projected momentum operator (15), the maximum magnitude of the mean momentum for a single-particle state is

$$\bar{P}_{\max} = \frac{2^k}{2}\max_\ell|\Delta^{(1)}_\ell| = \frac{2^k}{2}\Delta^{(1)}_{\max}. \tag{28}$$

By bounding this expression from above by the momentum cutoff Λ, we obtain

$$k = \left\lfloor \log_2\left(\frac{2\Lambda}{\Delta^{(1)}_{\max}}\right)\right\rfloor, \tag{29}$$

which, by $N=2^k$, Eq. (23), yields the sufficient number of modes in Eq. (25). ■

The established sufficiency, Eq. (25), for N is used by the main server to calculate the number of modes for the discretized QFT.

2. *Free-field ground state in a multiscale wavelet basis*

We now represent the ground state of the continuum theory, Eq. (4), in a multiscale wavelet basis. In this case, we project the continuum Hamiltonian onto a subspace of $\mathscr{L}^2$ (\$) that is a multiscale decomposition of the scale subspace $\mathcal{S}_k$ (Sec. II A). The projected Hamiltonian (14) onto a multiscale subspace is quadratic, similar to the fixed-scale Hamiltonian (11), but involves both scale- and wavelet-field operators (9).

The coupling matrix $\mathbf{K}^{(k)}$ in the multiscale Hamiltonian (14) is obtained by a multilevel wavelet transform from the fixed-scale coupling matrix $\mathbf{K}^{(k)}_{ss}$, Eq. (23). Let $s_0 < k$ be the scale index for the lowest scale, then $\mathbf{K}^{(k)}$ has the block-matrix structure

$$\mathbf{K}^{(k)} := \begin{bmatrix} \mathbf{K}^{(s_0)}_{ss} & \mathbf{K}^{(s_0,\,s_0)}_{sw} & \cdots & \mathbf{K}^{(s_0,\,k-1)}_{sw} \\ \mathbf{K}^{(s_0,\,s_0)\mathsf{T}}_{sw} & \mathbf{K}^{(s_0,\,s_0)}_{ww} & \cdots & \mathbf{K}^{(s_0,\,k-1)}_{ww} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}^{(s_0,\,k-1)\mathsf{T}}_{sw} & \mathbf{K}^{(s_0,\,k-1)\mathsf{T}}_{ww} & \cdots & \mathbf{K}^{(k-1,\,k-1)}_{ww} \end{bmatrix}, \tag{30}$$

imposed by the wavelet transform [18]. We select the level of wavelet transform so that the number of modes 2^{s_0} in

the lowest scale s_0, i.e., the number of rows or columns of the top-left block $\mathbf{K}_{ss}^{(s_0)}$, is at least $2(2\mathcal{K}-1)$ as per Sec. III A 1. The ground state of the projected Hamiltonian onto a multiscale wavelet basis is

$$\left|G_{\text{wavelet}}^{(k)}\right\rangle := \left(\frac{\det \mathbf{A}^{(k)}}{(2\pi)^N}\right)^{1/4} \int_{\mathbb{R}^N} d^N\boldsymbol{\phi}\, e^{-\frac{1}{4}\boldsymbol{\phi}^{\mathsf{T}}\mathbf{A}^{(k)}\boldsymbol{\phi}}\,|\boldsymbol{\phi}\rangle\,, \tag{31}$$

which is a continuous Gaussian pure state akin to the state in Eq. (24) but with the ICM

$$\mathbf{A}^{(k)} := \begin{bmatrix} \mathbf{A}_{ss}^{(s_0)} & \mathbf{A}_{sw}^{(s_0,\,s_0)} & \cdots & \mathbf{A}_{sw}^{(s_0,\,k-1)} \\ \mathbf{A}_{sw}^{(s_0,\,s_0)\mathsf{T}} & \mathbf{A}_{ww}^{(s_0,\,s_0)} & \cdots & \mathbf{A}_{ww}^{(s_0,\,k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{sw}^{(s_0,\,k-1)\mathsf{T}} & \mathbf{A}_{ww}^{(s_0,\,k-1)\mathsf{T}} & \cdots & \mathbf{A}_{ww}^{(k-1,\,k-1)} \end{bmatrix} = \sqrt{\mathbf{K}^{(k)}}, \tag{32}$$

which has the same block-matrix structure as the coupling matrix in Eq. (30). For convenience, we henceforth refer to the block with subscript "*ss*" as the *ss* block. Similarly, we refer to the blocks with subscript "*sw*" as the *sw* blocks and those with subscript "*ww*" as the *ww* blocks.

3. Discretization of continuous Gaussian pure states

Discretization is essential in obtaining a qubit representation for a continuous quantum state. Here we explain how we discretize a multidimensional continuous Gaussian pure state. We use the described method to discretize the ground state of the free field theory represented in both fixed- and multiscale wavelet bases to generate the ground state on a quantum register. First we define a discrete 1DG pure state over a lattice in Definition 3. Then we explain how to discretize a multidimensional Gaussian pure state.

Definition 3: (Discrete 1DG pure state over a lattice.) For $\sigma, \delta \in \mathbb{R}^+$ and $m \in \mathbb{Z}^+$, let $\mathbb{L} := \{j\,\delta \mid j \in [-2^{m-1}, 2^{m-1}) \cap \mathbb{Z}\}$ be a one-dimensional lattice with 2^m points and lattice spacing δ, and let $\tilde{\sigma} := \sigma/\delta$. We define the pure state

$$|G_{\text{lattice}}(\tilde{\sigma}, \delta, m)\rangle := \frac{1}{\tilde{\mathcal{N}}} \sum_{j=-2^{m-1}}^{2^{m-1}-1} \delta\, e^{-\frac{j^2}{4\tilde{\sigma}^2}}\,|j\,\delta\rangle\,,$$
$$\tilde{\mathcal{N}}^2 := \delta^2 \sum_{j=-2^{m-1}}^{2^{m-1}-1} e^{-j^2/2\tilde{\sigma}^2}, \tag{33}$$

for $|j\,\delta\rangle$ equally spaced lattice states in one dimension, as the discrete 1DG pure state with standard deviation $\tilde{\sigma}$ over lattice $\mathbb{L}$.

We use the discrete 1DG state in Eq. (33) as a discrete approximation for the continuous 1DG state, Eq. (19), with the standard deviation σ. The lattice parameters, i.e., the lattice spacing and the number of lattice points, are chosen based on two given inputs: the standard deviation and an error tolerance on the infidelity between the discrete and continuous 1DG states. In Sec. IV B 1, we describe how these two inputs are used to calculate the lattice parameters.

To discretize a continuous multidimensional Gaussian pure state, Eq. (17), first we decompose the state into a tensor product of several continuous 1DG pure states by a basis transformation. Then we discretize each continuous 1DG pure by a discrete 1DG pure state over a lattice as per Definition 3. Note that a continuous N-dimensional Gaussian pure state $|G_N(\mathbf{A})\rangle$, Eq. (17), with the ICM $\mathbf{A}$ is a linear combination of basis states $|\mathbf{x}\rangle := |x_0\rangle \otimes \cdots \otimes |x_{N-1}\rangle$, where $\mathbf{x}$ is a vector of real numbers. Let $\mathbf{O}$ be a matrix such that $\mathbf{O}^{\mathsf{T}}\mathbf{A}\mathbf{O}$ is a diagonal matrix $\mathbf{D}$. Then the basis transformation $|\mathbf{x}\rangle \mapsto |\mathbf{O}^{-1}\mathbf{x}\rangle$ yields the continuous Gaussian state with the diagonal ICM $\mathbf{D}$, which can be decomposed into a tensor product of N continuous 1DG states; see Sec. IV B 2.

C. Methods

In this subsection, we present our Fourier- and wavelet-based methods for generating an approximation for the free-field ground state on a quantum register. Both methods are based on a method for generating 1DG states. We begin in Sec. III C 1 by explaining our method for generating a discrete approximation for a continuous 1DG state. Then we describe the Fourier-based method in Sec. III C 2 and the wavelet-based method in Sec. III C 3.

1. One-dimensional Gaussian-state generation

Here we present two methods for generating a discrete approximation for a continuous 1DG state on a quantum register. First we specify the inputs along with the task in generating a 1DG state. Next we describe our discrete approximation for a 1DG state. Then we explain our first method for a 1DG-state generation. Our first method is similar to the Kitaev-Webb method [19]. However, in contrast to the Kitaev-Webb method, which is restricted to 1DG states with an extremely large standard deviation, our method generates 1DG states with any standard deviation. We finally describe our second method for generating a 1DG state, which is based on a method for performing inequality testing [15] on a quantum computer.

We begin by specifying the task in a 1DG-state generation. To generate a continuous 1DG state $|G(\sigma)\rangle$, Eq. (19), we are given two inputs: (1) the standard deviation $\sigma \in \mathbb{R}^+$ of the 1DG state and (2) an error tolerance $\varepsilon_{1\text{DG}} \in (0, 1)$. The task is to generate an approximate 1DG state $|\tilde{G}(\sigma)\rangle$

such that the infidelity [42]

$$\text{infid}\left(|G(\sigma)\rangle, \left|\tilde{G}(\sigma)\right\rangle\right) := 1 - \left\langle G(\sigma)\middle|\tilde{G}(\sigma)\right\rangle \in [0,1), \tag{34}$$

between the approximate and exact states is no greater than $\varepsilon_{1\text{DG}}$. We consider only continuous 1DG states with means of zero ($\mu = 0$) as we need only to prepare these states in order to generate an approximation for the ground state of the free QFT, Eq. (4).

We approximate $|G(\sigma)\rangle$ by a discrete 1DG pure state $|G_{\text{lattice}}(\tilde{\sigma}, \delta, m)\rangle$, Eq. (33), over a lattice with 2^m points and lattice spacing δ as per Definition 3. We select m and δ based on σ and $\varepsilon_{1\text{DG}}$ such that the infidelity between the continuous and discrete 1DG states is at most $\varepsilon_{1\text{DG}}$. Our approximate 1DG state is different from that of the Kitaev-Webb method. A continuous 1DG in their method is first approximated by a discrete 1DG state over an infinite lattice with unit spacing as in Eq. (20). The discrete 1DG state is then again approximated by the state in Eq. (21) to be generated on a quantum register. Our approximate 1DG state, however, is a discrete 1DG state over a finite lattice with a nonunit lattice spacing.

We now explain our strategy for generating $|G_{\text{lattice}}(\tilde{\sigma}, \delta, m)\rangle$, Eq. (33), on a quantum register. Our strategy comprises two steps. In the first step, we generate the state $|G_{\text{lattice}}(\tilde{\sigma}, 1, m)\rangle$. That is to say, we first prepare a discrete 1DG state with the same standard deviation but over a lattice with unit spacing. This state is a linear combination of basis states $|j\rangle$, where j is an integer; see Eq. (33) with $\delta = 1$. In the second step, we transform $|G_{\text{lattice}}(\tilde{\sigma}, 1, m)\rangle$ to $|G_{\text{lattice}}(\tilde{\sigma}, \delta, m)\rangle$ by performing the unitary map for which $|j\rangle \mapsto |j\,\delta\rangle$ for all j. Our method for generating $|G_{\text{lattice}}(\tilde{\sigma}, 1, m)\rangle$ is similar to the Kitaev-Webb method for 1DG-state generation. We write a recursive decomposition for $|G_{\text{lattice}}(\tilde{\sigma}, 1, m)\rangle$ analogous to Eq. (22) and employ the recursive decomposition to design an iterative algorithm for generating this state. See Sec. IV D 1 for a detailed description of the algorithm.

We now describe our inequality-testing-based method for generating the state $|G_{\text{lattice}}(\tilde{\sigma}, 1, m)\rangle$, Eq. (33). To elucidate the method, we write this state as $\sum_j g(j)\,|j\rangle$ with unnormalized amplitude distribution $g(j) := \exp\left(-j^2/4\sigma^2\right)$. To generate this state by inequality testing, first we prepare a quantum state with amplitude according to the value of j rounded down to the nearest power of 2. Specifically, we first prepare the state $\sum_j g_{\text{round}}(j)\,|j\rangle$ with unnormalized amplitude distribution $g_{\text{round}}(j) := \exp\left(2^{2\lfloor \log_2 j \rfloor}/4\sigma^2\right)$ for all $j \neq 0$ and $g_{\text{round}}(0) := 1$. The amplitude distributions $g(j)$ and $g_{\text{round}}(j)$ are shown in Fig. 5 by blue and orange points, respectively; these distributions are shown only for nonnegative j for simplicity.

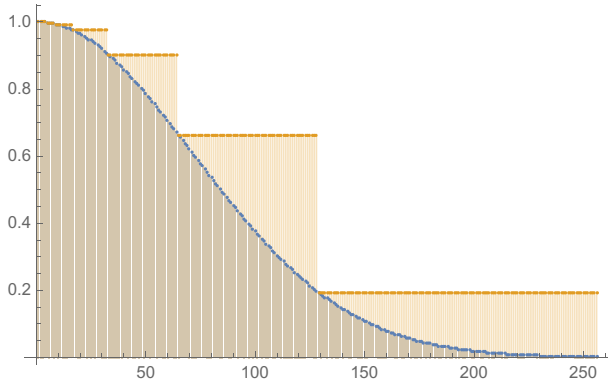

FIG. 5. Illustration of state-generation steps before performing an inequality test. First we prepare a state with amplitudes according to the orange points. Then we test an inequality against the blue points. The success probability is at least about 70%.

Once the state $\sum_j g_{\text{round}}(j)\,|j\rangle$ is generated, we then coherently compute an approximation for the ratio of the amplitudes $r_j := g_{\text{round}}(j)/g(j)$ into a scratch register and prepare a reference quantum register in uniform superposition. Next we perform an inequality test between the value encoded in the scratch register and the value encoded in the reference register, and write the result into a flag qubit. We then erase the reference register and measure the flag qubit. If the postmeasurement state of the flag qubit is $|0\rangle$, then the state generated on the first register is the desired 1DG state. See Sec. IV D 2 for a detailed description of the inequality-testing-based algorithm for generating a 1DG state.

2. Fourier-based method for ground-state generation

We now present our Fourier-based method for generating a discrete approximation for the free-field ground state on a quantum register. First we explain the rationale for specifying the task in this method and describe the task. Then we explain our strategy for generating the approximate ground in the Fourier-based method.

In the Fourier-based method, we discretize the continuum QFT in a fixed-scale wavelet basis and use the ground state of the discretized QFT as an approximation for that of the continuum theory. The discretized QFT ground state in this method is a continuous Gaussian state whose ICM $\mathbf{A}_{ss}$, Eq. (24), is a circulant matrix. The ICM, which fully describes the discretized QFT ground state, is specified by three parameters: wavelet index $\mathcal{K} \in \mathbb{Z}^+$, free QFT mass $m_0 \in \mathbb{R}^+$, and the number of modes $N \in \mathbb{Z}^+$ of the discretized QFT.

For the Fourier-based method to generate the ground state, we are given an error tolerance $\varepsilon_{\text{vac}} \in (0,1)$ along with the parameters that specify the ground-state ICM. The task is to generate an approximation for the ground state

$|G_{\rm scale}\rangle$, Eq. (24), of the discretized QFT in a fixed-scale basis on a quantum register. The infidelity between the approximate and exact states is required to be no greater than the error tolerance $\varepsilon_{\rm vac}$.

Our strategy for generating an approximate ground state for the discretized QFT is as follows. First we construct a classical algorithm for computing the eigenvalues $\boldsymbol{\lambda} := (\lambda_0, \ldots, \lambda_{N-1})$ of the ground-state ICM; eigenvalues are diagonals of the diagonal matrix $\boldsymbol{\Lambda}$ in the spectral decomposition of the ICM. In this algorithm, we exploit the circulant structure of the ICM and compute $\boldsymbol{\lambda}$ by a discrete Fourier transform [43, p. 100]. Next we use $\boldsymbol{\lambda}$ as a classical input to design a quantum circuit for generating an approximation for the state $|G_N(\boldsymbol{\Lambda})\rangle$, Eq. (17), i.e., the continuous Gaussian state whose ICM is the diagonal matrix $\boldsymbol{\Lambda}$. Finally, we perform a basis transformation by a quantum fast Fourier transform (QFFT) to map the state $|G_N(\boldsymbol{\Lambda})\rangle$ to the ground state $|G_N(\mathbf{A}_{ss})\rangle$. The state $|G_N(\boldsymbol{\Lambda})\rangle$ is a linear combination of basis states $|\mathbf{x}\rangle := |x_1\rangle \otimes \cdots \otimes |x_N\rangle$, where $\mathbf{x} = (x_1, \ldots, x_N)$ is a vector of real numbers; see Eq. (17). The QFFT implements the map $|\mathbf{x}\rangle \mapsto |\mathbf{F}\mathbf{x}\rangle$ on a quantum computer, where $\mathbf{F}$ is the transformation matrix for the discrete Fourier transform.

To design a quantum circuit for preparing an approximation for a continuous NDG state $|G_N(\mathbf{D})\rangle$ with a diagonal inverse-covariance matrix $\mathbf{D} := \mathrm{diag}(d_0, \ldots, d_{N-1})$, first we decompose the state as $|G(\sigma_0)\rangle \otimes \cdots \otimes |G(\sigma_{N-1})\rangle$, where $|G(\sigma_\ell)\rangle$, Eq. (19), is a continuous 1DG state with the standard deviation $\sigma_\ell := 1/\sqrt{d_\ell}$. By the method described in Sec. III C 1, we then design a quantum circuit for generating a discrete approximation for each 1DG state. The combined output of all these quantum circuits is an approximation for the continuous Gaussian state with the diagonal ICM $\mathbf{D}$.

3. Wavelet-based method for ground-state generation

Here we present our wavelet-based method for generating a discrete approximation for the free-field ground state represented in a multiscale wavelet basis. Similar to the Fourier-based method, we first provide the rationale for specifying the task in the wavelet-based method and describe the task. Then we explain the strategy for generating the approximate ground state.

In the wavelet-based method, we discretize the continuum QFT in a multiscale wavelet basis and use the ground state of the discretized QFT as an approximation for that of the continuum theory. The ground state, Eq. (31), of the discretized QFT in this method is also a continuous Gaussian state. The same parameters specify the ground-state ICM here as in the Fourier-based method. For the wavelet-based method, we are given the same classical inputs as the Fourier-based method. The task, however, is to generate an approximation for the ground state $|G_{\rm wavelet}\rangle$, Eq. (31), of the discretized QFT in a multiscale wavelet basis such that the infidelity between the approximate and exact states is no greater than the error tolerance $\varepsilon_{\rm vac}$.

The strategy for generating an approximate ground state in the wavelet-based method is as follows. The ground-state ICM in a multiscale wavelet basis has many near-zero elements. We approximate this matrix by replacing its near-zero elements with exactly zero. Specifically, we replace all matrix elements whose magnitudes are less than the threshold value $\varepsilon_{\rm th} = m_0 \varepsilon_{\rm vac} N^{-3/2}$ with exactly zero. This approximation enables a fingerlike sparse structure [44] for the ground-state ICM with a quasilinear number of nonzero elements; see Fig. 1.

We exploit the fingerlike structure and perform the UDU matrix decomposition of the approximate ICM in a quasilinear time. In the UDU decomposition, we decompose the fingerlike sparse matrix $\tilde{\mathbf{A}}$ as the product of an upper unit-triangular matrix $\mathbf{U}$, a diagonal matrix $\mathbf{D}$ and transpose of $\mathbf{U}$. We compute diagonals of $\mathbf{D}$ and shear elements, i.e., nonzero off-diagonal elements, of $\mathbf{U}$ by a classical algorithm, and use them as classical inputs to construct a quantum circuit for generating an approximation for the free-field ground state.

To generate an approximate ground state, first we construct a quantum circuit for preparing an approximation for the NDG state $|G_N(\mathbf{D})\rangle$ whose ICM is the diagonal matrix $\mathbf{D}$ in the UDU decomposition. The diagonals of $\mathbf{D}$ are used as classical inputs, and the quantum circuit is constructed by the method described in Sec. III C 2. Then we transform $|G_N(\mathbf{D})\rangle$ into the ground state by performing a quantum shear transform (QST) on a quantum computer; the shear elements of $\mathbf{U}$ are classical inputs for the QST. Akin to the QFFT in the Fourier-based method, the QST executes a basis transformation. Specifically, the QST implements the map $|\mathbf{x}\rangle \mapsto |\mathbf{S}\mathbf{x}\rangle$ on a quantum computer, where the shear-transform matrix $\mathbf{S}$ is the inverse transpose of $\mathbf{U}$.

IV. RESULTS

In this section, we present our main results. We first construct a high-level description of our ground-state-generation algorithms in Sec. IV A. Next we discuss the number of required qubits for representing an approximation for the discretized QFT ground state in the Fourier- and wavelet-based methods in Sec. IV B. Our algorithms have classical preprocessing and quantum routine. We present the classical preprocessing of our algorithms in Sec. IV C and the quantum routine in Sec. IV D. Then we analyze the runtime of our algorithms in Sec. IV E. In Sec. IV F, we establish a lower bound on the gate complexity for ground-state generation. Finally, in Sec. IV G, we compare the Fourier versus wavelet approach for ground-state generation.

A. High-level description of our algorithms for ground-state generation

We begin with a high-level description of our two algorithms for ground-state generation. The first of these algorithms, described in Sec. IV A 1, is based on the use of a discrete Fourier transform, and we refer to it as the Fourier-based algorithm. The second algorithm, described in Sec. IV A 2, is based on the use of a wavelet transform, and we call it the wavelet-based algorithm.

Both algorithms have a similar structure that we now explain. The algorithms proceed in two stages: (1) prepare several independent one-dimensional Gaussian states, and (2) perform a collection of arithmetic operations on those Gaussian states. Both stages require a certain amount of classical information, much of which is not provided directly by the main server (Sec. III A 2) but requires a nonnegligible amount of computation to produce. We therefore must analyze not only the quantum complexity but also the classical complexity of our algorithms in order to ensure that the resulting procedures are indeed quasilinear in the number of modes of the discretized QFT. We refer to the classical part of our algorithm as the "classical preprocessing" step, as it must be carried out prior to the execution of our quantum algorithms. In our descriptions of the Fourier-based and wavelet-based algorithms, we therefore distinguish between the classical preprocessing procedure and the quantum algorithm itself.

1. High-level description of Fourier-based algorithm

Here we construct a high-level description of the Fourier-based algorithm for ground-state generation. We begin by explaining this algorithm's classical preprocessing and then describe the quantum routine. Finally, we present the algorithm by pseudocode to elucidate the inputs, output, and procedure of the Fourier-based algorithm.

The Fourier-based algorithm generates an approximation for the free-field ground state, Eq. (24), represented in a fixed-scale basis. To generate this state, first we generate N discrete 1DG states over a lattice with spacing δ. The task in the classical preprocessing of the Fourier-based algorithm is to compute the standard deviations $\tilde{\boldsymbol{\sigma}} := (\tilde{\sigma}_0, \ldots, \tilde{\sigma}_{N-1})$ for the discrete 1DG states as per Definition 3, and the lattice spacing δ; these are the needed parameters for the quantum routine in the Fourier-based algorithm. To compute $\tilde{\boldsymbol{\sigma}}$ and δ, first we compute the derivative overlaps, Eq. (13), for the second-order derivative, i.e., Laplace, operator. We then use these derivative overlaps and the bare mass m_0 to compute the eigenvalues $\boldsymbol{\lambda} := (\lambda_0, \ldots, \lambda_{N-1})$ of the ground-state ICM. Having $\boldsymbol{\lambda}$, we compute the lattice spacing as $\delta = 1/\sqrt{\lambda_{\max}}$ and the 1DG standard deviations as $\tilde{\boldsymbol{\sigma}} = 1/(\delta\sqrt{\boldsymbol{\lambda}})$. Figure 6 (top) shows a schematic description of the classical preprocessing in the Fourier-based algorithm.

In the quantum routine, we use outputs of the classical preprocessing to generate an approximation for the ground state. For each component of $\tilde{\boldsymbol{\sigma}}$ and the lattice spacing δ, we generate a discrete 1DG state, Eq. (33), corresponding to these inputs on a quantum register. Then we execute a quantum fast Fourier transform. The QFFT performs a discrete Fourier transform by a collection of arithmetic operations on the 1DG states. The resulting state is an approximation for the ground state, Eq. (24), represented in a fixed-scale basis. Figure 6 (bottom) shows a schematic description of the quantum routine in the Fourier-based algorithm. For clarity, we present the inputs, outputs, and procedure of this algorithm as pseudocode in Algorithm 1.

2. High-level description of wavelet-based algorithm

We now present a high-level description of the wavelet-based algorithm for ground-state generation. This algorithm, similar to the Fourier-based algorithm, has classical preprocessing and quantum routine. We begin by describing the classical preprocessing and proceed with explaining the quantum routine. We finally present the wavelet-based algorithm as pseudocode to specify the algorithm's inputs, output, and procedure.

In classical preprocessing of the wavelet-based algorithm, we compute the required inputs for the quantum routine. These inputs are the lattice spacing δ, the vector $\tilde{\boldsymbol{\sigma}}$ of standard deviations for the discrete 1DG states, and shear elements of the upper unit-triangular matrix $\mathbf{U}$ in the UDU decomposition of the approximate ICM (Sec. III C 3) for the free-field ground state, Eq. (31), represented in a multiscale wavelet basis. The first two inputs are needed to generate the discrete 1DG states, and the last input is needed to perform the basis transformation.

The needed classical inputs for quantum routine of the wavelet-based algorithm are computed as follows. First we compute the second-order derivative overlaps in Eq. (13). These derivative overlaps are then used to compute the unique matrix elements of the ground-state ICM. The ICM in Eq. (32) is a block matrix, and each block is a circulant matrix. The unique matrix elements are, therefore, the circuit row of each block in the block matrix. Next we use the circulant rows, denoted by $\mathbf{a}$, to compute the vector $\mathbf{d}$ of diagonals in the diagonal matrix $\mathbf{D}$ and the shear elements $\mathbf{S}$ of the upper unit-triangular matrix $\mathbf{U}$ in the UDU decomposition of the approximate ICM. Having $\mathbf{d}$, we compute the lattice spacing as $\delta = 1/\sqrt{d_{\max}}$ and the vector of standard deviations as $\tilde{\boldsymbol{\sigma}} = 1/(\delta\sqrt{\mathbf{d}})$, where $d_{\max}$ is the largest element of $\mathbf{d}$. Figure 7 (top) shows a schematic description of the classical preprocessing in the wavelet-based algorithm.

The quantum routine of the wavelet-based algorithm proceeds as follows. Similar to the quantum routine of the Fourier-based algorithm, first we generate a discrete 1DG state, Eq. (33), for each component of $\tilde{\boldsymbol{\sigma}}$ and the

Input:
$\mathcal{K} \in \mathbb{Z}_{\geq 3}$ ▷ wavelet index
$m_0 \in \mathbb{R}^+$ ▷ free-QFT mass
$N \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ ▷ number of modes
$\varepsilon_{\text{vac}} \in (0,1)$ ▷ error tolerance for output state
Output:
$|\tilde{\text{G}}\rangle \in \mathscr{H}_2^{N\times\lceil\log_2(N/\sqrt{m_0\varepsilon_{\text{vac}}})\rceil}$ ▷ $\left(N \times \lceil \log_2\left(N/\sqrt{m_0\varepsilon_{\text{vac}}}\right)\rceil\right)$-qubit approximate ground state
function FOURIERBASEDGSG($\mathcal{K}, m_0, N, \varepsilon_{\text{vac}}$)
Classical preprocessing
$\mathbb{Z}^+ \ni p \leftarrow \lceil \log_2\left(N/\sqrt{m_0\varepsilon_{\text{vac}}}\right)\rceil$ ▷ computes working precision p
$\mathbb{R}^{2\mathcal{K}-1} \ni \boldsymbol{\Delta} \leftarrow$ DERIVATIVEOVERLAPS($\mathcal{K}, p$) ▷ computes derivative overlaps (13) for Laplace operator in index-$\mathcal{K}$ wavelet basis
$\mathbb{R}^N \ni \boldsymbol{\lambda} \leftarrow$ INVCOVEIGENS($\mathcal{K}, m_0, N, \boldsymbol{\Delta}, p$) ▷ computes eigenvalues of the ICM by Algorithm 3
$\mathbb{R}^+ \ni \delta \leftarrow 1/\sqrt{\text{MAX}(\boldsymbol{\lambda})}$ ▷ computes lattice spacing δ
$\mathbb{R}^N \ni \tilde{\boldsymbol{\sigma}} \leftarrow 1/(\delta\sqrt{\boldsymbol{\lambda}})$ ▷ computes standard deviation $\tilde{\boldsymbol{\sigma}}$ of approximate 1DG states
Quantum routine
for $\ell \leftarrow 0$ to $N-1$ **do**
$\mathscr{H}_2^p \ni \text{vac}[\ell] \leftarrow$ ONEDG $(\tilde{\sigma}_\ell, \delta)\,|0^p\rangle$ ▷ generates p-qubit approximate 1DG state with standard deviation $\tilde{\sigma}_\ell$ by Algorithm 6
$\mathscr{H}_2^{N\times p} \ni \text{vac} \leftarrow \text{QFFT}\left(N, \bigotimes_{\ell=0}^{N-1} \text{vac}[\ell]\right)$ ▷ transforms the 1DG states into the approximate ground state by Algorithm 8
yield vac ▷ we use **yield** instead of **return** for quantum algorithms as the output is a state generated on a quantum register

Algorithm 1. Fourier-based algorithm for ground-state generation

lattice spacing δ. Using the shear elements **S** as classical input, we then perform a basis transformation on the discrete 1DG states by executing a quantum shear transform. The QST performs a collection of arithmetic operations to map the discrete 1DG states to an approximation for the ground state, Eq. (31), of the discretized QFT in a multiscale wavelet basis. A schematic description of the quantum routine in the wavelet-based algorithm is shown in Fig. 7 (bottom). The pseudocode in Algorithm 2 specifies the inputs, output, and procedure of the wavelet-based algorithm.

B. Space requirement to represent the ground state

In this subsection, we determine how the number of qubits required to represent an approximation for the free-field ground state in both Fourier- and wavelet-based methods scales with respect to the basis-independent input parameters in Table II specified by the main server for ground-state generation. The wavelet index $\mathcal{K}$ in Table II is basis-dependent and is typically chosen to be a small constant number [18,29,33]. Hence we exclude this parameter in our analysis.

We begin in Sec. IV B 1 by discussing the number of required qubits to represent a discrete approximation for a continuous 1DG state as in Eq. (19). Having determined the space required to represent a 1DG state, we then analyze the space required to represent an approximation for a continuous multidimensional Gaussian state in Sec. IV B 2. Our result on the space requirement for representing a multidimensional Gaussian state allows us to show how the space required to represent the free-field ground state scales in terms of the inputs specified by the main server.

1. Space requirement to represent a one-dimensional Gaussian state

Here we determine the minimal number of qubits required to represent an approximation for a continuous 1DG state, Eq. (19), in terms of its standard deviation and an error tolerance on the infidelity, Eq. (34), between the approximate and continuous states. First we explain how we approximate a continuous 1DG state. Then we establish a bound on the infidelity between our approximate 1DG state and the continuous 1DG state. Having this bound, we then show that the number of qubits needed to represent the approximate 1DG state is logarithmic in the ratio of the standard deviation to the square root of the error tolerance.

We begin by discussing how we approximate a continuous 1DG pure state. We are given the standard deviation $\sigma \in \mathbb{R}^+$ of a continuous 1DG pure state and an error tolerance $\varepsilon_{\text{1DG}} \in (0,1)$. We approximate the continuous 1DG pure state by a discrete 1DG pure state over a one-dimensional lattice with 2^m points and lattice spacing δ as per Definition 3. We determine δ and m in terms of σ and ε_{1DG} such that the infidelity, Eq. (34), between the discrete and continuous 1DG states is at most ε_{1DG}. We show, in Proposition 4, how to determine δ and m in terms of σ and ε_{1DG}.

Proposition 4: *Let* $|G(\sigma)\rangle$*, Eq. (19), be a continuous* 1DG *state with the standard deviation* $\sigma \in \mathbb{R}^+$ *and let* $|G_{lattice}(\tilde{\sigma}, \delta, m)\rangle$ *in Eq. (33), be a discrete* 1DG *state with standard deviation* $\tilde{\sigma} := \sigma/\delta$ *over a one-dimensional lattice with* 2^m *points and lattice spacing* $\delta \in \mathbb{R}^+$*. For* $\varepsilon_{\text{1DG}} \in (0,1)$*, if*

$$\delta \leq \min(1/2, \sigma) \quad \textit{and} \quad 2^m\delta \geq 2\sigma/\sqrt{\varepsilon_{\text{1DG}}}, \tag{35}$$

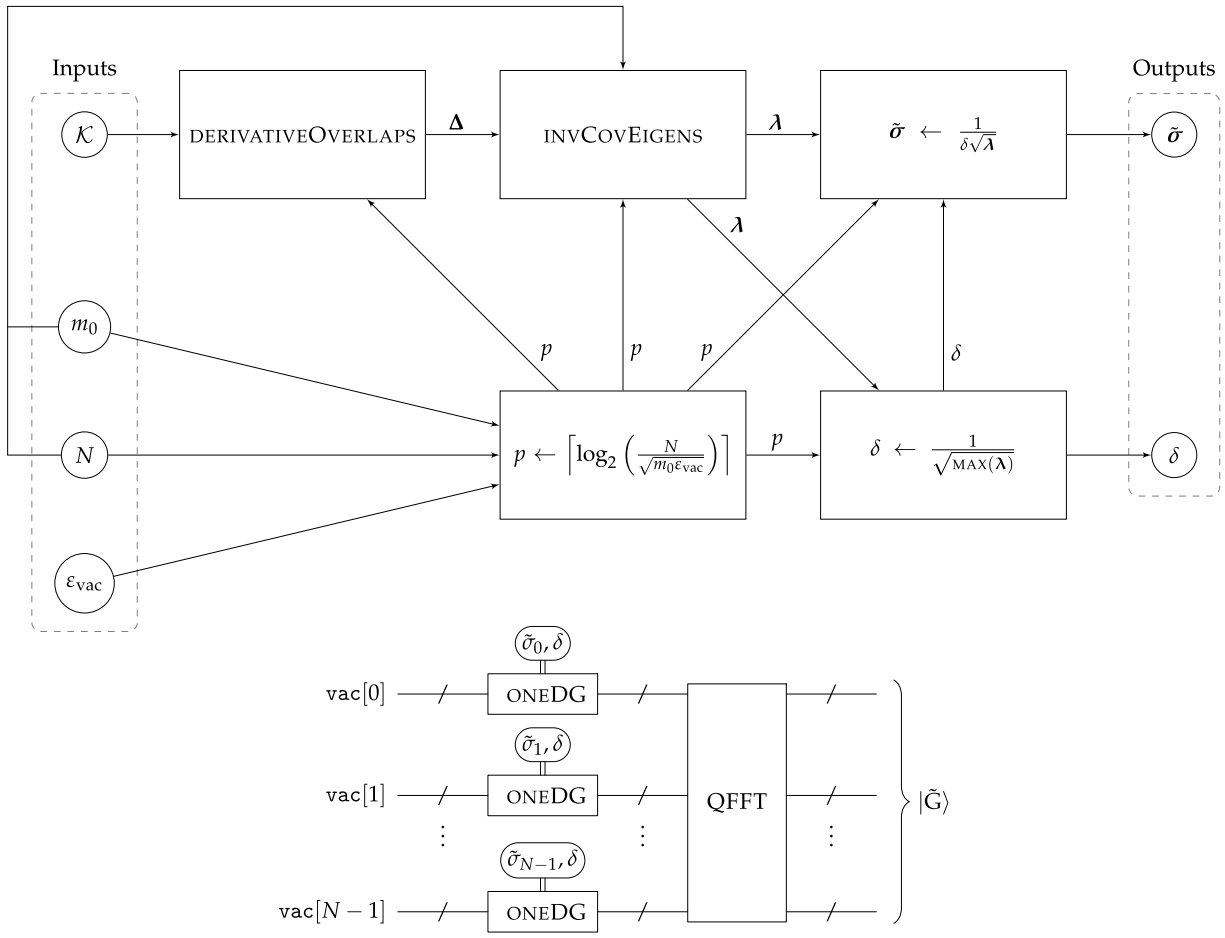

FIG. 6. Description of Fourier-based algorithm for ground-state generation. Top: classical preprocessing. Inputs: wavelet index $\mathcal{K}$, mass m_0, number of modes N in the discretized QFT, and error tolerance ε_{vac} for the output state. Each box represents a process; incoming arrows identify the inputs, and outgoing arrows identify the outputs of the process. Outputs: standard deviation $\tilde{\boldsymbol{\sigma}}$ of the approximate one-dimensional Gaussian states and lattice spacing δ. Outputs of intermediate processes: working precision p, second-order derivative overlaps $\boldsymbol{\Delta}$ and eigenvalues $\boldsymbol{\lambda}$ of the ground-state ICM. Bottom: quantum routine. Double lines indicate classical inputs to the quantum routine. vac is a quantum register with N cells, and each cell comprises p qubits; $\vdash$ represents multiple qubits. Each ONEDG accepts δ and one component of $\tilde{\boldsymbol{\sigma}}$ as classical inputs and generates an approximate 1DG state corresponding to these inputs on once cell of vac. The quantum fast Fourier transform (QFFT) acts collectively on the set of approximate 1DG states and transforms them into the approximate ground state $|\tilde{G}\rangle$.

then the infidelity, Eq. (34), between the discrete and continuous 1DG *states is bounded above by* ε_{1DG}.

Proof. Let $J := 2^m$. Then the fidelity between the continuous, Eq. (19), and discrete, Eq. (33), 1DG states is

$$\begin{aligned} F &:= \langle G_{\text{lattice}}(\tilde{\sigma},\delta,m)|G(\sigma)\rangle \\ &= \frac{1}{\mathcal{N}\tilde{\mathcal{N}}}\sum_{j=-J/2}^{J/2-1}\delta\, e^{-\frac{j^2}{2\tilde{\sigma}^2}} \geq \int_{-J\delta/2}^{J\delta/2} dx\,\rho(x) \\ &= \Pr(|x|\leq J\delta/2)\,, \end{aligned} \tag{36}$$

where the inequality follows from Proposition 14. Here $\rho(x) := \mathcal{N}^{-2}e^{-x^2/(2\sigma^2)}$ is the probability density function of a Gaussian distribution and $\Pr(|x|\leq J\delta/2)$ is the probability that $x\in[-J\delta/2, J\delta/2]$. By Eq. (36), $J\delta/2 \geq \sigma/\sqrt{\varepsilon_{\text{1DG}}}$ in Eq. (35), and $\Pr(|x|\leq J\delta/2) = 1 - \Pr(|x| > J\delta/2)$, we have

$$F \geq 1 - \Pr\big(|x|\geq \sigma/\sqrt{\varepsilon_{\text{1DG}}}\big)\,. \tag{37}$$

Using Chebyshev's inequality [42, p. 609],

$$\Pr\big(|x|\geq \sigma/\sqrt{\varepsilon_{\text{1DG}}}\big) \leq \varepsilon_{\text{1DG}}, \tag{38}$$

which together with Eq. (37) yields $F \geq 1-\varepsilon_{\text{1DG}}$. The infidelity $1-F$ is therefore bounded from above by ε_{1DG}. ■

Input:
$\mathcal{K} \in \mathbb{Z}_{\geq 3}$ ▷ wavelet index
$m_0 \in \mathbb{R}^+$ ▷ free-QFT mass
$N \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ ▷ number of modes
$\varepsilon_{\text{vac}} \in (0,1)$ ▷ error tolerance for output state
Output:
$|\tilde{\text{G}}\rangle \in \mathscr{H}_2^{N\times\lceil\log_2(N/\sqrt{m_0\varepsilon_{\text{vac}}})\rceil}$ ▷ $\left(N \times \lceil \log_2\left(N/\sqrt{m_0\varepsilon_{\text{vac}}}\right)\rceil\right)$-qubit approximate ground state
function WAVELETBASEDGSG($\mathcal{K}, m_0, N, \varepsilon_{\text{vac}}$)
Classical preprocessing
$\mathbb{Z}^+ \ni p \leftarrow \lceil \log_2\left(N/\sqrt{m_0\varepsilon_{\text{vac}}}\right)\rceil$ ▷ computes working precision p
$\mathbb{R}^{2\mathcal{K}-1} \ni \boldsymbol{\Delta} \leftarrow$ DERIVATIVEOVERLAPS($\mathcal{K}, p$) ▷ computes the derivative overlaps (13) for Laplace operator by Algorithm 11
$\mathbf{a} := \left\{ \mathbf{a}_{ss}^{(s_0)} \in \mathbb{R}^{2^{s_0}}, \mathbf{a}_{sw}^{(s_0,c)} \in \mathbb{R}^{2^c}, \mathbf{a}_{ww}^{(r,c)} \in \mathbb{R}^{2^{c-r}} \middle| s_0 \leq r \leq c < k \right\} \leftarrow$ INVCOVCIRCROWS $(\mathcal{K}, m_0, N, \boldsymbol{\Delta}, \varepsilon_{\text{vac}}, p)$ ▷ computes circulant row in main and upper-diagonal blocks of multi-scale ICM (32) by Algorithm 4; here $s_0 := \lceil \log_2(4\mathcal{K}-2)\rceil$ and $k := \log_2 N$
$\left\{\mathbb{R}^N \ni \mathbf{d}, \mathbb{R}^{\widetilde{\mathcal{O}}(N)} \ni \mathbf{S}\right\} \leftarrow$ INVCOVUDU $(m_0, N, \varepsilon_{\text{vac}}, \mathbf{a})$ ▷ computes diagonals $\mathbf{d}$ of $\mathbf{D}$ and shear elements $\mathbf{S}$ of $\mathbf{U}$ in UDU decomposition of approximate ICM by Algorithm 5. Here $\widetilde{\mathcal{O}}(N)$ denotes a quasilinear number; see Sec. IV C 3 and Algorithm 5 for details
$\mathbb{R}^+ \ni \delta \leftarrow 1/\sqrt{\text{MAX}(\mathbf{d})}$ ▷ computes lattice spacing δ
$\mathbb{R}^N \ni \tilde{\boldsymbol{\sigma}} \leftarrow 1/(\delta\sqrt{\mathbf{d}})$ ▷ computes standard deviation $\tilde{\boldsymbol{\sigma}}$ of approximate 1DG states
Quantum routine
for $\ell \leftarrow 0$ to $N-1$ **do**
$\mathscr{H}_2^p \ni \text{vac}[\ell] \leftarrow$ ONEDG $(\tilde{\sigma}_\ell, \delta)\,|0^p\rangle$ ▷ generates p-qubit approximate 1DG state with standard deviation $\tilde{\sigma}_\ell$ by Algorithm 6
$\mathscr{H}_2^{N\times p} \ni \text{vac} \leftarrow \text{QST}\left(N, \mathbf{S}, \bigotimes_{\ell=0}^{N-1} \text{vac}[\ell]\right)$ ▷ transforms the 1DG states into the approximate ground state by Algorithm 9
yield vac

Algorithm 2. Wavelet-based algorithm for ground-state generation

Proposition 4 allows us to obtain the minimal number of qubits to represent the discrete 1DG state in Eq. (33). We provide the result for the minimal number of qubits in the following proposition and proceed with a proof.

Proposition 5: *For $\sigma \in \mathbb{R}^+$ the standard deviation of a continuous* 1DG *state, Eq. (19), and $\varepsilon_{\text{1DG}} \in (0,1)$ an error tolerance, at least*

$$n_{\text{1DG}} = \lceil \log_2\left(\sigma/\sqrt{\varepsilon_{\text{1DG}}}\right)\rceil + \max\left(1,\ \lceil \log_2(1/\sigma)\rceil\right), \tag{39}$$

qubits are required to represent a discrete approximation for the continuous state with infidelity, Eq. (34), no greater than ε_{1DG}.

Proof. The discrete 1DG state, Eq. (33), is a superposition of lattice states $|j\,\delta\rangle$, where $j \in [-2^{m-1}, 2^{m-1}) \cap \mathbb{Z}$ and $j\,\delta$ is a real number. Hence the smallest nonzero value for the real numbers is δ, and the largest value is $2^{m-1}\delta$. By virtue of Proposition 4, taking $\delta = \min(1/2, \sigma)$ and $2^{m-1}\delta = \sigma/\sqrt{\varepsilon_{\text{1DG}}}$ ensures that the infidelity between the approximate and continuous 1DG states is at most ε_{1DG}. As $\delta < 1$, the number of qubits needed to represent the fractional part of the real numbers is $\max\left(1, \lceil \log_2(1/\sigma)\rceil\right)$. The largest value for the real numbers is $\sigma/\sqrt{\varepsilon_{\text{1DG}}}$, so $\lceil \log_2\left(\sigma/\sqrt{\varepsilon_{\text{1DG}}}\right)\rceil$ qubits are needed to represent the integer part of the real numbers. The total number of qubits needed to represent the approximate 1DG state is obtained by adding the number of qubits required to represent the real numbers' integer and fractional parts. ■

Having determined the space requirement for representing a 1DG state, next we establish a bound on the number of qubits needed to represent a multidimensional Gaussian state.

2. Space requirement to represent a multidimensional Gaussian state

We now determine the minimal number of qubits needed to represent an approximation for a continuous N-dimensional Gaussian state in terms of the condition number of its ICM, N and an error tolerance on the infidelity between the approximate and continuous states. We present the result in Theorem 6 and proceed with a proof. Then we invoke this theorem to obtain the space requirement for representing the free-field ground state in both Fourier- and wavelet-based methods in terms of the inputs specified by the main server.

Theorem 6: *Let $\mathbf{A} \in \mathbb{R}^{N\times N}$ be the* ICM *for a continuous N-dimensional Gaussian state $|G_N(\mathbf{A})\rangle$, Eq. (17), and let $\varepsilon_G \in (0,1)$ be an error tolerance. Also let $\kappa \in \mathbb{R}^+$ be the condition number of $\mathbf{A}$, quantified as the ratio of the largest to smallest eigenvalues of $\mathbf{A}$. Then*

$$n \in \Omega\left(N \log_2\left(\frac{N\kappa}{\varepsilon_G}\right)\right), \tag{40}$$

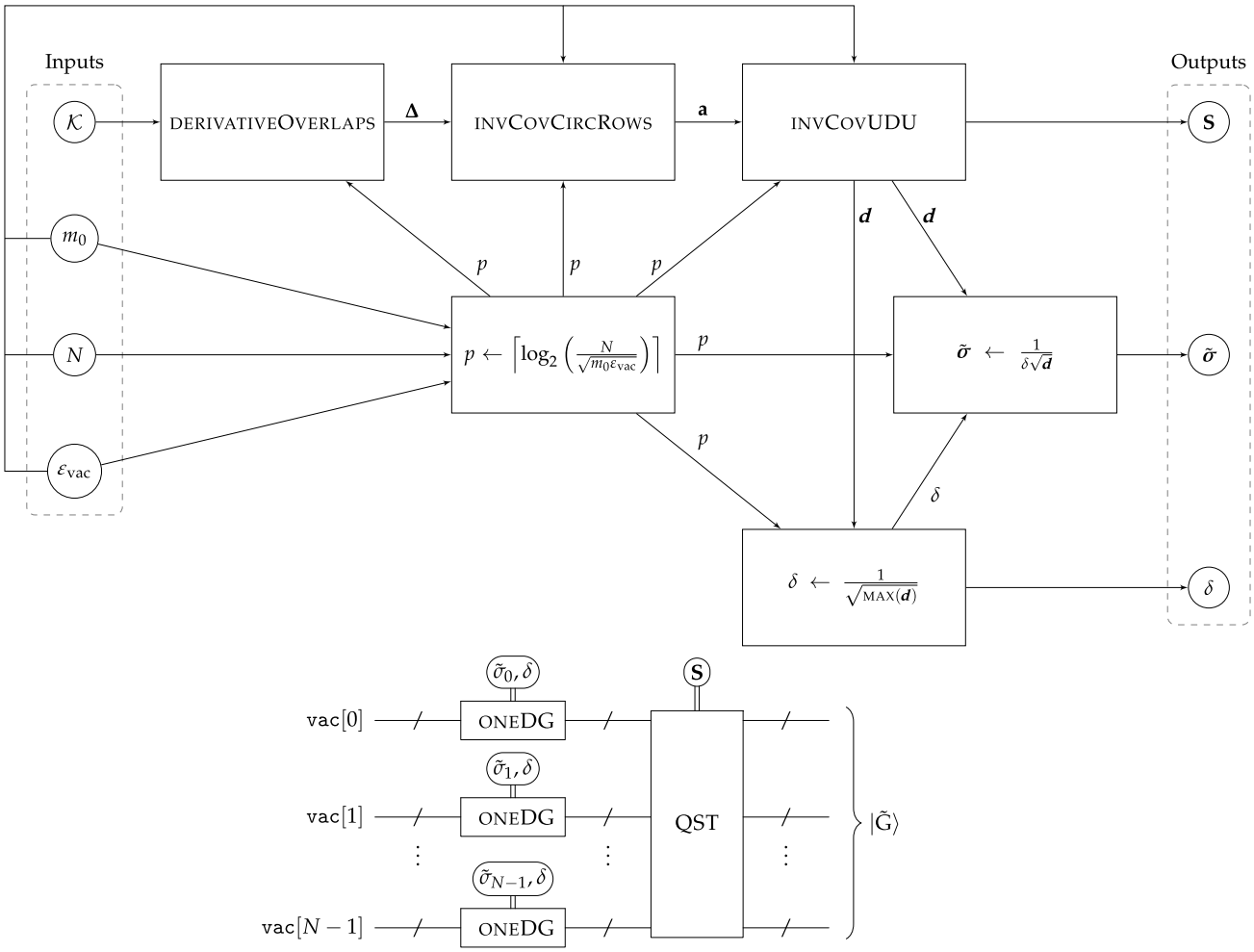

FIG. 7. Description of wavelet-based algorithm for ground-state generation. Top: classical preprocessing. Inputs are the same as the inputs to the Fourier-based algorithm Outputs: shear elements **S** of the upper unit-triangular matrix in the UDU decomposition of the approximate ICM, standard deviation $\tilde{\boldsymbol{\sigma}}$ of the approximate one-dimensional Gaussian states and lattice spacing δ. Outputs of intermediate processes: working precision p, second-order derivative overlaps $\mathbf{\Delta}$, circulant rows **a** of the upper-triangular blocks in the ICM and diagonals **d** of the diagonal matrix in the UDU decomposition. Bottom: quantum routine. Double lines indicate classical inputs to the quantum routine. vac is a quantum register with N cells, and each cell comprises p qubits; $\vdash$ represents multiple qubits. Each ONEDG accepts δ and one component of $\tilde{\boldsymbol{\sigma}}$ as classical inputs and generates an approximate 1DG state corresponding to these inputs on one cell of vac. The quantum shear transform (QST) acts collectively on the set of approximate 1DG states and transforms them into the approximate ground state $\left|\tilde{G}\right\rangle$.

qubits are required to represent a discrete approximation for $|G_N(\mathbf{A})\rangle$ *such that the infidelity between the discrete and continuous states is bounded from above by* ε_G.

Proof. Let **D** be the diagonal matrix in either the spectral or the UDU decomposition of the ICM **A**. Then the space required to represent $|G_N(\mathbf{A})\rangle$ is the same as the space required to represent $|G_N(\mathbf{D})\rangle$ because the former state is obtained from the latter by a basis transformation. Therefore, we determine the required space to represent $|G_N(\mathbf{D})\rangle$. We decompose this state into a tensor product of N continuous 1DG states as

$$|G_N(\mathbf{D})\rangle = \bigotimes_{\ell=0}^{N-1} |G(\sigma_\ell)\rangle\,, \quad \sigma_\ell := 1/\sqrt{D_{\ell\ell}}, \tag{41}$$

where σ_ℓ is the standard deviation of the ℓ^{th} continuous 1DG state $|G(\sigma_\ell)\rangle$, Eq. (19). We approximate each 1DG state by a discrete 1DG state over a lattice with 2^m points and spacing δ as per Definition 3. The approximate

Gaussian state is then

$$\left|\tilde{G}_N(\mathbf{D})\right\rangle := \bigotimes_{\ell=0}^{N-1} |G_{\text{lattice}}(\tilde{\sigma}_\ell, \delta, m)\rangle\,, \tag{42}$$

where $\tilde{\sigma}_\ell := \sigma_\ell/\delta$ is the standard deviation of the ℓ^{th} discrete 1DG state, Eq. (33). By Proposition 4, if

$$\varepsilon_{\text{1DG}} = \varepsilon_G/N, \quad \delta \le \min(1/2, \sigma_{\min}) \quad \text{and} \quad 2^m\delta \ge 2\sigma_{\max}/\sqrt{\varepsilon_{\text{1DG}}}, \tag{43}$$

then for each ℓ

$$\langle G_{\text{lattice}}(\tilde{\sigma}_\ell, \delta, m)|G(\sigma_\ell)\rangle \ge 1 - \varepsilon_G/N. \tag{44}$$

Equations (41)–(44) yield

$$\left\langle \tilde{G}_N(\mathbf{D})\middle|G_N(\mathbf{D})\right\rangle = \prod_{\ell=0}^{N-1} \langle G_{\text{lattice}}(\tilde{\sigma}_\ell, \delta, m)|G(\sigma_\ell)\rangle \ge (1-\varepsilon_G/N)^N \ge 1-\varepsilon_G. \tag{45}$$

Each discrete 1DG state, Eq. (33), is a superposition of lattice states $|j\delta\rangle$, where $j\delta$ is a real number. To ensure that Eq. (44) holds for each 1DG state, we need at least $\lceil \log_2(\sigma_{\max}\sqrt{N/\varepsilon_G})\rceil$ qubits to represent the integer part and at least $\lceil \log_2(1/\sigma_{\min})\rceil$ qubits to represent the fractional part of the real numbers. Thus, the minimal number of qubits to represent each 1DG state scales as

$$n_{\text{1DG}} \in \Omega\left(\log_2\left((\sigma_{\max}/\sigma_{\min})\sqrt{N/\varepsilon_G}\right)\right). \tag{46}$$

Let $d_{\max}$ and $d_{\min}$ be, respectively, the largest and smallest diagonal elements of $\mathbf{D}$, then by Eq. (41)

$$\frac{\sigma_{\max}}{\sigma_{\min}} = \sqrt{\frac{d_{\max}}{d_{\min}}}, \tag{47}$$

and by Proposition 15, $d_{\max}/d_{\min} \in O(\kappa)$. The combination of these equations with Eq. (46) yields $n_{\text{1DG}} \in \Omega[\log_2(N\kappa/\varepsilon_G)]$. Therefore, the total number of qubits to represent an approximation for a N-dimensional Gaussian state scales as Eq. (40). ■

We now determine the space required to represent an approximation for the ground state in both Fourier- and wavelet-based methods. To this end, by Theorem 6, we need only to bound the condition number κ of the ground-state ICM for each method to obtain the space requirement in terms of the parameters specified by the main server. By Proposition 16, the condition number of the ground state's ICM for both methods scales as $\kappa \in \Theta(N/m_0)$. Therefore, by Theorem 6, the number of qubits needed to represent the ground state scales as

$$n \in \Omega\left(N\log_2\left(N/\sqrt{m_0\varepsilon_{\text{vac}}}\right)\right), \tag{48}$$

with respect to the parameters specified by the main server in Table II. Notice that n is quasilinear in the number of modes N. The logarithmic factor here is the number of qubits required to represent each 1DG state for generating the free-field ground state. For simplicity, we use

$$p = \lceil \log_2\left(N/\sqrt{m_0\varepsilon_{\text{vac}}}\right)\rceil, \tag{49}$$

in description of our ground-state-generation algorithms for number of qubits to represent each 1DG state.

C. Classical preprocessing

In this subsection, we construct key subroutines of the classical preprocessing in the Fourier- and wavelet-based algorithms for ground-state generation. We begin, in Sec. IV C 1, by constructing a classical algorithm for computing the eigenvalues of the ground-state ICM in a fixed-scale wavelet basis; this algorithm is used as a subroutine in classical preprocessing of both ground-state-generation algorithms. Then, in Sec. IV C 2, we present a classical algorithm for computing the circulant row in unique blocks of the ground-state ICM in a multiscale wavelet basis. Finally, in Sec. IV C 3, we construct a classical algorithm for computing the UDU decomposition of the ICM in a multiscale wavelet basis. The last two algorithms are subroutines of the classical preprocessing in the wavelet-based algorithm.

1. Eigenvalues of the ground state's inverse-covariance matrix

Here we devise a classical algorithm for computing the eigenvalues of the ground-state ICM $\mathbf{A}_{ss}$, Eq. (24), in a fixed-scale wavelet basis. First we state the properties of this matrix used to compute the eigenvalues and explain the parameters that specify unique elements of the matrix. We then provide the rationale for computing the eigenvalues of $\mathbf{A}_{ss}$ by a DHT and present our algorithm as pseudocode. Finally, we elaborate on the relationship between the eigenvalues of the ground-state ICM in fixed- and multiscale wavelet bases.

We begin by stating properties of the fixed-scale ICM, Eq. (24), used for computing the eigenvalues. This matrix is the principal square root of the fixed-scale coupling matrix $\mathbf{K}_{ss}$, Eq. (23), which itself is a circulant, real, and symmetric matrix. Being the coupling matrix's principal square root, the ICM is also a circulant, real, and symmetric matrix. A circulant matrix is fully specified by its first row, which we call the "circulant row" of the circulant matrix. Four parameters specify the unique elements in the

Input:
$\mathcal{K} \in \mathbb{Z}_{\geq 3}$ ▷ wavelet index
$m_0 \in \mathbb{R}^+$ ▷ mass of free QFT
$N \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ ▷ number of modes
$\boldsymbol{\Delta} \in \mathbb{R}^{2\mathcal{K}-1}$ ▷ derivative overlaps (13) for second-order derivative operator in a wavelet basis with index $\mathcal{K}$
Output:
$\boldsymbol{\lambda} \in \mathbb{R}^N$ ▷ eigenvalues of ground-state ICM $\mathbf{A}_{ss}$ (24)

function INVCOVEIGENS($\mathcal{K}, m_0, N, \boldsymbol{\Delta}$)
  for $j \leftarrow 0$ to $N-1$ **do**
    $\text{tmp}_j \leftarrow m_0^2 - N^2\Delta_0 - 2\sum_{\ell=1}^{2\mathcal{K}-1} N^2\Delta_\ell \cos\left(\frac{2\pi\ell}{N} j\right)$ ▷ computes j^{th} eigenvalue of the coupling matrix $\mathbf{K}_{ss}$ (23)
    $\lambda_j \leftarrow \sqrt{\text{tmp}_j}$ ▷ computes j^{th} eigenvalue of $\mathbf{A}_{ss}$
  return $\boldsymbol{\lambda}$

Algorithm 3. Classical algorithm for computing eigenvalues of ground-state ICM

circulant row of $\mathbf{K}_{ss}$: the wavelet index $\mathcal{K}$, the second-order derivative overlaps Δ_ℓ, Eq. (13), the number of modes N and the free mass m_0; see Eq. (23). The same parameters specify the unique matrix elements of the fixed-scale ICM.

We now provide the rationale for computing the eigenvalues of $\mathbf{A}_{ss}$, Eq. (24), by a DHT. Any circulant matrix is diagonalizable by a discrete Fourier transform [43, p. 100]. By the real and symmetric properties of the ICM, we diagonalize this matrix by a DHT. In particular, the eigenvalues of any real, symmetric, and circulant matrix are obtained by computing the DHT of the matrix's circulant row [43, p. 100]. As the ICM is the principal square root of the coupling matrix, first we compute the coupling-matrix eigenvalues by the DHT of its circulant row. Then we take the square root of the coupling-matrix eigenvalues to obtain the eigenvalues of the ICM $\mathbf{A}_{ss}$. Algorithm 3 provides the procedure for computing the eigenvalues of the ICM using the parameters that specify unique elements of the coupling matrix.

The coupling matrix, Eq. (30), in a multiscale wavelet basis is obtained from the coupling matrix, Eq. (23), in a fixed-scale wavelet basis by a wavelet transform, which is a unitary transformation. For each basis, the ground-state ICM is the principal square root of the coupling matrix. Consequently, the ICM in a multiscale wavelet basis is obtained by the same unitary wavelet transform from the ICM in a fixed-scale basis, and they have identical eigenvalues. Therefore, we use Algorithm 3 as a subroutine in classical preprocessing of both Fourier- and wavelet-based algorithms for computing the eigenvalues of the ground-state ICM.

2. Elements of the ground state's ICM

We now construct a classical algorithm to compute the unique matrix elements of the ground-state ICM in a multiscale wavelet basis. First we state key properties of the coupling matrix $\mathbf{K}$, Eq. (30), represented in this basis. Then we explain how we approximate the multiscale ICM and discuss the classical memory requirement to store unique elements of the approximate ICM. Next we explain our algorithm's procedure for computing the unique matrix elements, and finally, we present our algorithm as pseudocode.

We begin with a few observations about the multiscale coupling matrix $\mathbf{K}$ in Eq. (30). We then extend these observations to the approximate ICM. The coupling matrix $\mathbf{K}$ has the following key properties:

1. *Block structure.* The matrix $\mathbf{K}$ has a block-matrix structure imposed by a wavelet transform, with three types of blocks: *ss*, *sw*, and *ww*; see Eq. (30). We therefore present entries of $\mathbf{K}$ based on their block's location in the block matrix.
2. *Symmetry.* The coupling matrix is symmetric. Therefore, we consider only the unique blocks, which are the main and upper-diagonal blocks of $\mathbf{K}$.
3. *Banded circulant blocks.* The main and upper-diagonal blocks of $\mathbf{K}$ have a circulant structure; specifically, each block is a banded 2^k-circulant matrix for some non-negative integer k. Each diagonal block is a banded 1-circulant matrix with bandwidth $w_d := 2(2\mathcal{K}-2)+1$. The *ww* block at entry (r, c) of the block matrix $\mathbf{K}$, Eq. (30), for any $c > r$, is a banded 2^{c-r}-circulant matrix with bandwidth $w_d + (2^{c-r}-1)(2\mathcal{K}-1)$. The *sw* block at entry (s_0, c) is a banded 2^{c-s_0}-circulant matrix with the same bandwidth as the *ww* block at entry (s_0, c).

These observations are also true for the multiscale ICM $\mathbf{A}$, Eq. (32), except that the blocks are no longer banded. However, imposing a cutoff condition on elements of $\mathbf{A}$ by some threshold value ε_{th} reimposes the banded structure for blocks of this matrix. To describe the cutoff condition and how we approximate the ICM, first we define an ε_{th}-approximate ICM as follows.

Definition 7: (ε_{th}-approximate ICM). Given any ICM $\mathbf{A} \in \mathbb{R}^{N\times N}$ and any $\varepsilon_{\text{th}} > 0$, an ε_{th}-approximate ICM is a

symmetric matrix $\mathbf{A}_{\varepsilon_{\text{th}}}$ such that

$$\left[A_{\varepsilon_{\text{th}}}\right]_{ij} := \begin{cases} 0 & \text{if } |A_{ij}| < \varepsilon_{\text{th}}, \\ A_{ij} & \text{otherwise.} \end{cases} \tag{50}$$

As per Definition 7, we approximate the ground-state ICM by replacing its near-zero matrix elements, i.e., the elements with magnitude less than some close-to-zero threshold value ε_{th}, with exactly zero. This replacement rule enables a sparse structure for the ICM that we exploit to perform its UDU decomposition in quasilinear time.

We show, in Proposition 8, that not only the approximate ICM obtained by imposing the cutoff condition is a positive-definite matrix but also the infidelity between the Gaussian state with the approximate ICM and the free-field ground state is bounded from above by the error tolerance ε_{vac} in Table II.

Proposition 8: *Given any* $\varepsilon_{\text{vac}} \in (0,1)$ *and any* ICM $\mathbf{A} \in \mathbb{R}^{N\times N}$*, then for any* ε_{th} *satisfying*

$$0 < \varepsilon_{\text{th}} \leq \varepsilon_{\text{vac}} N^{-3/2} \min \text{spec} \mathbf{A}, \tag{51}$$

every ε_{th}*-approximate* ICM $\mathbf{A}_{\varepsilon_{\text{th}}}$ *is a positive-definite matrix such that*

$$\text{infid}\left(|G_N(\mathbf{A})\rangle, \left|G_N(\mathbf{A}_{\varepsilon_{\text{th}}})\right\rangle\right) \leq \varepsilon_{\text{vac}}, \tag{52}$$

where $|G_N(\mathbf{A})\rangle$ *is a* N*-dimensional continuous Gaussian state with the* ICM $\mathbf{A}$ *as per Definition 1.*

This proposition is proven in Appendix D 1.

The approximate ICM obtained by imposing the cutoff condition is sparse because most elements of the exact ICM $\mathbf{A}$, Eq. (32), have an exponentially close-to-zero value. In particular, we show in Proposition 9 that diagonal blocks of $\mathbf{A}$ decay exponentially away from the diagonal elements. A corollary of the exponential decay, shown in Corollary 10, is that the diagonal blocks are banded circulant matrices with a bandwidth that is logarithmic in the number of modes N; here we refer to the number of cyclically nonzero elements in any row of a circulant matrix as "bandwidth" of the matrix.

Proposition 9: (Exponentially decaying diagonal blocks). *Let* $m_0 \in \mathbb{R}^+$ *be the free mass and* $\mathbf{A} \in \mathbb{R}^{N\times N}$*, Eq. (32), be the ground-state* ICM *in a multiscale wavelet basis with the wavelet index* $\mathcal{K} \in \mathbb{Z}_{\geq 3}$*. Also let* $\{\mathbf{A}^{(r,r)}_{ww} : \lceil \log_2(4\mathcal{K}-2)\rceil \leq r < \log_2 N\}$ *be the diagonal* ww *blocks of* $\mathbf{A}$ *as in Eq. (32). Then for any* r *and* $j \geq 2\mathcal{K}-1$

$$|A^{(r,r)}_{ww;\,0,j}| \leq 16\mathcal{K} m_0 \kappa^{(r+1)} 2^{-|j|/\xi^{(r+1)}},$$
$$\xi^{(r)} := (2\mathcal{K}-1)2^{r+1}/m_0, \tag{53}$$

where $\kappa^{(r)} > 1$ *is the spectral condition number of* $\mathbf{K}^{(r)}_{ss}$ *in Eq. (23).*

This proposition is proven in Appendix D 2.

Corollary 10: (Banded circulant blocks). *The diagonal blocks of the approximate* ICM *in a multiscale wavelet basis are banded matrices with the upper bandwidth*

$$w = \left\lceil \frac{2(2\mathcal{K}-1)}{m_0} \log N \log_2\left(\frac{4\mathcal{K}N}{m_0\varepsilon_{\text{vac}}}\right)\right\rceil, \tag{54}$$

where all parameters are specified in Table II.

By this corollary, proven in Appendix D 3, each diagonal ww block of the approximate ICM $\tilde{\mathbf{A}}$ is a banded 1-circulant matrix with bandwidth $2w+1$. The wavelet transform implies that the off-diagonal ww blocks are also banded matrices. Specifically, the off-diagonal ww block at entry (r,c) of the block matrix $\tilde{\mathbf{A}}$ is a banded 2^{c-r}-circulant matrix ($\forall\, c > r$) with bandwidth

$$\text{WIDTH}(r,c) := 2w + 1 + (2^{c-r}-1)(2\mathcal{K}-1). \tag{55}$$

However, the ss and sw blocks of $\tilde{\mathbf{A}}$ are not necessarily banded matrices. We therefore treat these blocks as dense matrices.

We now describe a data structure for representing the block matrix in a multiscale wavelet basis. The date structure that we describe here takes advantage of the matrix's block and circulant structures for efficient storage. In particular, we use the data structure for storing the multiscale ICM $\mathbf{A}$, Eq. (32). We store the block matrix in a multiscale wavelet basis by an associative array, i.e., by a collection of (key, value) pairs. Each key is a tuple (z,r,c), where $z \in \{ss, sw, ww\}$ specifies if the block belongs to the ss, sw, or ww part of the block matrix. The positive integers r and c, respectively, specify the row and column indices of the block matrix; these integers also specify the scales to which the block belongs. The value of the associative array specifies the block at entry (z,r,c) of the block matrix. Each block, being a circulant matrix, is specified by a vector, which is the first row of the block, and two additional parameters: the block's size and the amount by which the vector is shifted when moving from one row to the next. For our application, each key specifies the block's size and the amount by which the vector is shifted. Therefore, the values are vectors that specify blocks of the block matrix.

The memory requirement for storing a symmetric block matrix with circulant blocks using the described data stricture is quasilinear in the matrix's dimension. The fact that each block matrix is circulant means that we need only to store one row of each circulant block; the other rows are shifted versions of the stored row. Assuming unit cost for storing each element, the cost of storing a block matrix with circulant blocks is equal to the sum of the row size for each block, which is altogether quasilinear in the matrix's dimension. The symmetry of the matrix reduces this cost

by a factor of 2. The memory cost can be further reduced by noting that most of the entries in the vector specifying the circulant blocks are equal to zero. We do not use this technique because we are only concerned with producing a quasilinear algorithm for ground-state generation. The memory-cost reduction delivers only a constant-factor improvement and makes the algorithm much more complicated. However, it is an obvious way to improve the classical preprocessing of the wavelet-based algorithm further.

We now describe our algorithm's procedure for computing the circulant row in unique blocks of the multiscale ICM $\mathbf{A}$, Eq. (32). To elucidate the algorithm's procedure, first we state recursive relations for blocks of the multiscale ICM. The ICM in a multiscale wavelet basis is obtained by a wavelet transform from the ICM in a fixed-scale wavelet basis. Specifically, for $d := k - s_0$,

$$\mathbf{A}^{(k)} = \mathbf{W}_d^{(k)} \mathbf{A}_{ss}^{(k)} \mathbf{W}_d^{(k)\mathsf{T}}, \tag{56}$$

where $\mathbf{W}_d^{(k)}$, Eq. (A7), is the d-level wavelet-transform matrix at scale k. By this equation, for scales r and c with $s_0 \le r \le c < k$, the wavelet transform imposes the recursive relations

$$\mathbf{A}_{ss}^{(c)} = \mathbf{H} \mathbf{A}_{ss}^{(c+1)} \mathbf{H}^{\mathsf{T}}, \tag{57}$$

$$\mathbf{A}_{sw}^{(r,c)} = \mathbf{H}^{c-r+1} \mathbf{A}_{ss}^{(c+1)} \mathbf{G}^{\mathsf{T}}, \tag{58}$$

$$\mathbf{A}_{ww}^{(r,c)} = \mathbf{G} \mathbf{H}^{c-r} \mathbf{A}_{ss}^{(c+1)} \mathbf{G}^{\mathsf{T}}, \tag{59}$$

for blocks of the multiscale ICM $\mathbf{A}$ and the fixed-scale ICM $\mathbf{A}_{ss}^{(c+1)}$, where $\mathbf{H}$ and $\mathbf{G}$ are the upper and lower half of $\mathbf{W}_d^{(k)}$, Eq. (A7), respectively. We use these recursive formulae to compute the circulant in unique blocks of the multiscale ICM $\mathbf{A}$.

We start by computing the circulant row of the bottom-right block in $\mathbf{A}$ and proceed to compute the circulant row of the top-left block column by column. For each column c of the block matrix, first we compute the circulant row of $\mathbf{A}_{ss}^{(c)}$ by Eq. (57). Next we compute circulant row of the diagonal block in column c using Eq. (59) with $r = c$. For each ww block above the diagonal block, i.e., for ww blocks with row index r form $c-1$ to s_0, we then iteratively update $\mathbf{A}_{ss}^{(c+1)}$ as $\mathbf{A}_{ss}^{(c+1)} \leftarrow \mathbf{H}\mathbf{A}_{ss}^{(c+1)}$ and compute circulant row of $\mathbf{A}_{ww}^{(r,c)}$ by $\mathbf{A}_{ww}^{(r,c)} = \mathbf{G}\mathbf{A}_{ss}^{(c+1)}\mathbf{G}^{\mathsf{T}}$. Finally, we compute the circulant row of the sw block in column c by $\mathbf{A}_{sw}^{(s_0,c)} = \mathbf{A}_{ss}^{(c+1)}\mathbf{G}^{\mathsf{T}}$ because $\mathbf{A}_{ss}^{(c+1)}$ in Eq. (58) is updated $c - s_0 + 1$ times while computing the circulant row of the ww blocks in column c. The explicit procedure of our algorithm for computing the circulant rows is presented in Algorithm 4.

3. UDU decomposition of the ground state's ICM

Here we present our classical algorithm for computing the UDU decomposition of the approximate ICM in a multiscale wavelet basis. First we describe a block variant of the UDU decomposition for a real-symmetric matrix. Next we explain how we approximate the UDU decomposition of the approximate ICM. We then specify the inputs and outputs for our UDU-decomposition algorithm for a sparse matrix and explain an efficient method for storing the outputs. Finally, we describe our algorithm's procedure and present the algorithm as pseudocode.

We begin by describing a block variant for the UDU decomposition of a dense real-symmetric matrix $\mathbf{A}$ that has a block structure as the matrix in Eq. (32); see Appendix F for the standard UDU decomposition of $\mathbf{A}$. By Eq. (F1), we write the UDU decomposition of the block matrix $\mathbf{A}$, Eq. (32), as

$$\begin{aligned}\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}} &:= \mathbf{U}\mathbf{V}^{\mathsf{T}} \\ &= \begin{bmatrix} \mathbf{U}_{ss}^{(s_0)} & \mathbf{U}_{sw}^{(s_0,\,s_0)} & \cdots & \mathbf{U}_{sw}^{(s_0,\,k-1)} \\ & \mathbf{U}_{ww}^{(s_0,\,s_0)} & \cdots & \mathbf{U}_{ww}^{(s_0,\,k-1)} \\ & & \ddots & \vdots \\ & & & \mathbf{U}_{ww}^{(k-1,\,k-1)} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{V}_{ss}^{(s_0)} & \mathbf{V}_{sw}^{(s_0,\,s_0)} & \cdots & \mathbf{V}_{sw}^{(s_0,\,k-1)} \\ & \mathbf{V}_{ww}^{(s_0,\,s_0)} & \cdots & \mathbf{V}_{ww}^{(s_0,\,k-1)} \\ & & \ddots & \vdots \\ & & & \mathbf{V}_{ww}^{(k-1,\,k-1)} \end{bmatrix}^{\mathsf{T}},\end{aligned} \tag{60}$$

where

$$\mathbf{D} := \mathbf{D}_{ss}^{(s_0)} \bigoplus_{s=s_0}^{k-1} \mathbf{D}_{ww}^{(s)}, \tag{61}$$

is a block-diagonal matrix and

$$\begin{aligned}&\mathbf{V}_{ss}^{(s_0)} := \mathbf{U}_{ss}^{(s_0)}\mathbf{D}_{ss}^{(s_0)}, \mathbf{V}_{sw}^{(s_0,c)} := \mathbf{U}_{sw}^{(s_0,c)}\mathbf{D}_{ww}^{(c)}, \\ &\mathbf{V}_{ww}^{(r,c)} := \mathbf{U}_{ww}^{(r,c)}\mathbf{D}_{ww}^{(c)} \quad (s_0 \le r \le c < k),\end{aligned} \tag{62}$$

are blocks of the block matrix $\mathbf{V}$, which has the same block structure as $\mathbf{U}$. Diagonal elements of $\mathbf{D}$ and shear elements of $\mathbf{U}$ are computed in the UDU matrix decomposition. By Eq. (60), the i^{th} diagonal elements of $\mathbf{D}_{ww}^{(c)}$ and $\mathbf{D}_{ss}^{(s_0)}$ are

$$d_{ww;\,i}^{(c)} = a_{ww;\,i,i}^{(c,c)} - \sum_{s=c}^{k-1} \mathbf{u}_{ww;\,i,(i+1)\delta_{sc}:2^s-1}^{(c,s)} \cdot \mathbf{v}_{ww;\,i,(i+1)\delta_{sc}:2^s-1}^{(c,s)}, \tag{63}$$

$$\begin{aligned}d_{ss;\,i}^{(s_0)} = a_{ss;\,i,i}^{(s_0)} &- \mathbf{u}_{ss;\,i,i+1:2^{s_0}-1}^{(s_0)} \cdot \mathbf{v}_{ss;\,i,i+1:2^{s_0}-1}^{(s_0)} \\ &- \sum_{s=s_0+1}^{k-1} \mathbf{u}_{sw;\,i,0:2^s-1}^{(s_0,s)} \cdot \mathbf{v}_{sw;\,i,0:2^s-1}^{(s_0,s)},\end{aligned} \tag{64}$$

respectively; see also Eq. (F4).

Input:
$\mathcal{K} \in \mathbb{Z}_{\geq 3}$ ▷ wavelet index
$m_0 \in \mathbb{R}^+$ ▷ free-QFT mass
$N \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ ▷ number of modes
$\boldsymbol{\Delta} \in \mathbb{R}^{2\mathcal{K}-1}$ ▷ derivative overlaps (13) for Laplace operator
$p \in \mathbb{Z}^+$ ▷ working precision
Output:
$\mathbf{a} := \left\{ \mathbf{a}_{ss}^{(s_0)} \in \mathbb{R}^{2^{s_0}}, \mathbf{a}_{sw}^{(s_0,c)} \in \mathbb{R}^{2^c}, \mathbf{a}_{ww}^{(r,c)} \in \mathbb{R}^{2^{c-r}} \,\middle|\, \lceil \log_2(4\mathcal{K}-2) \rceil =: s_0 \leq r \leq c < k := \log_2 N \right\}$ ▷ circulant row in main and upper-diagonal blocks of the multi-scale ICM $\mathbf{A}$ (32)

function INVCOVCIRCROWS($\mathcal{K}, m_0, N, \boldsymbol{\Delta}, p$)
  $\mathbb{R}^N \ni \boldsymbol{\lambda} \leftarrow$ INVCOVEIGENS($\mathcal{K}, m_0, N, \boldsymbol{\Delta}, p$) ▷ computes eigenvalues of $\mathbf{A}$ by Algorithm 3
  for $j \leftarrow 0$ to $N-1$ **do** ▷ computes circulant row of $\mathbf{A}_{ss}^{(k)}$
    $a_{ss;\,j}^{(k)} \leftarrow \frac{1}{N} \sum_{i=0}^{N-1} \lambda_i \cos\left(\frac{2\pi j}{N} i\right)$
  $\mathbb{R}^{2\mathcal{K}} \ni \mathbf{h} \leftarrow$ LOWPASSFILTER($\mathcal{K}, p$) ▷ computes low-pass filter for index-$\mathcal{K}$ wavelet by Algorithm 10
  for $i \leftarrow 0$ to $2\mathcal{K}-1$ **do** ▷ computes high-pass filter
    $g_i \leftarrow (-)^i h_{2\mathcal{K}-1-i}$
  for $c \leftarrow k-1$ to s_0 **do** ▷ iterates over column index of the block matrix $\mathbf{A}$
    for $j \leftarrow 0$ to $2^c - 1$ **do**
      $a_{ss;\,j}^{(c)} \leftarrow \sum_{m,n=0}^{2\mathcal{K}-1} h_m h_n a_{ss;\,(m+2j-n) \bmod 2^{c+1}}^{(c+1)}$ ▷ computes circulant row of $\mathbf{A}_{ss}^{(c)}$ from the circulant row of $\mathbf{A}_{ss}^{(c+1)}$ by Eq. (57)
    for $r \leftarrow c$ to s_0 **do** ▷ iterates over row index of the block matrix $\mathbf{A}$
      for $j \leftarrow 0$ to $2^c - 1$ **do**
        $a_{ww;\,j}^{(r,c)} \leftarrow \sum_{m,n=0}^{2\mathcal{K}-1} g_m g_n a_{ss;\,(m+2j-2^{c-r}n) \bmod 2^{c+1}}^{(c+1)}$ ▷ computes circulant row of $\mathbf{A}_{ww}^{(r,c)}$
      $\mathbf{x} \leftarrow \mathbf{a}_{ss}^{(c+1)}$ ▷ to be updated, the circulant row of $\mathbf{A}_{ss}^{(c+1)}$ is stored on scratchpad as a temporary vector $\mathbf{x}$
      for $j \leftarrow 0$ to $2^{c+1} - 1$ **do**
        $a_{ss;\,j}^{(c+1)} \leftarrow \sum_{m=0}^{2\mathcal{K}-1} h_m x_{(j-2^{c-r}m) \bmod 2^{c+1}}$
    for $j \leftarrow 0$ to $2^c - 1$ **do** ▷ computes circulant row of the sw block in column c
      $a_{sw;\,j \bmod 2^c}^{(s_0,c)} \leftarrow \sum_{m=0}^{2\mathcal{K}-1} g_m a_{ss;\,(m+2j) \bmod 2^{c+1}}^{(c+1)}$
  return a

Algorithm 4. Classical algorithm for computing circulant row in blocks of the ICM, Eq. (32), in multiscale wavelet basis

We now establish formulae to compute shear elements in various blocks of $\mathbf{U}$. For $0 < i < 2^{s_0}$, the shear elements in the i^{th} column of $\mathbf{U}$ are the same as the shear elements in the i^{th} column of $\mathbf{U}_{ss}^{(s_0)}$. By Eq. (F5), these elements are

$$\mathbf{u}_{ss;\,0:i-1,i}^{(s_0)} = \frac{1}{d_{ss;\,i}^{(s_0)}} \left[\mathbf{A}_{ss;\,0:i-1,i}^{(s_0)} - \mathbf{u}_{ss;\,0:i-1,i+1:2^{s_0}-1}^{(s_0)} \cdot \mathbf{v}_{ss;\,i,i+1:2^{s_0}-1}^{(s_0)} - \sum_{s=s_0+1}^{k-1} \mathbf{u}_{sw;\,0:i-1,0:2^s-1}^{(s_0,s)} \cdot \mathbf{v}_{sw;\,i,0:2^s-1}^{(s_0,s)} \right] \quad \forall i \neq 0. \tag{65}$$

For $c \geq s_0$ and $0 \leq i < 2^c$, the shear elements in the $(2^c + i)^{\text{th}}$ column of $\mathbf{U}$ are the elements in the i^{th} column of $\mathbf{U}_{sw}^{(s_0,c)}$ and $\mathbf{U}_{ww}^{(r,c)}$ for $s_0 \leq r < c$, and the elements above the diagonal entries of $\mathbf{U}_{ww}^{(c,c)}$. By Eq. (F5), these elements are

$$\mathbf{u}_{sw;\,0:2^{s_0}-1,i}^{(s_0,c)} = \frac{1}{d_{ww;\,i}^{(c)}} \left[\mathbf{A}_{sw;\,0:2^{s_0}-1,i}^{(s_0,c)} - \sum_{s=c}^{k-1} \mathbf{u}_{sw;\,0:2^{s_0}-1,(i+1)\delta_{sc}:2^s-1}^{(s_0,s)} \cdot \mathbf{v}_{ww;\,i,(i+1)\delta_{sc}:2^s-1}^{(c,s)} \right], \tag{66}$$

$$\mathbf{u}_{ww;\,0:2^r-1,i}^{(r,c)} = \frac{1}{d_{ww;\,i}^{(c)}} \left[\mathbf{A}_{ww;\,0:2^r-1,i}^{(r,c)} - \sum_{s=c}^{k-1} \mathbf{u}_{ww;\,0:2^r-1,(i+1)\delta_{sc}:2^s-1}^{(r,s)} \cdot \mathbf{v}_{ww;\,i,(i+1)\delta_{sc}:2^s-1}^{(c,s)} \right], \tag{67}$$

$$\mathbf{u}_{ww;\,0:i-1,i}^{(c,c)} = \frac{1}{d_{ww;\,i}^{(c)}} \left[\mathbf{A}_{ww;\,0:i-1,i}^{(c,c)} - \sum_{s=c}^{k-1} \mathbf{u}_{ww;\,0:i-1,(i+1)\delta_{sc}:2^s-1}^{(c,s)} \cdot \mathbf{v}_{ww;\,i,(i+1)\delta_{sc}:2^s-1}^{(c,s)} \right] \quad \forall i \neq 0, \tag{68}$$

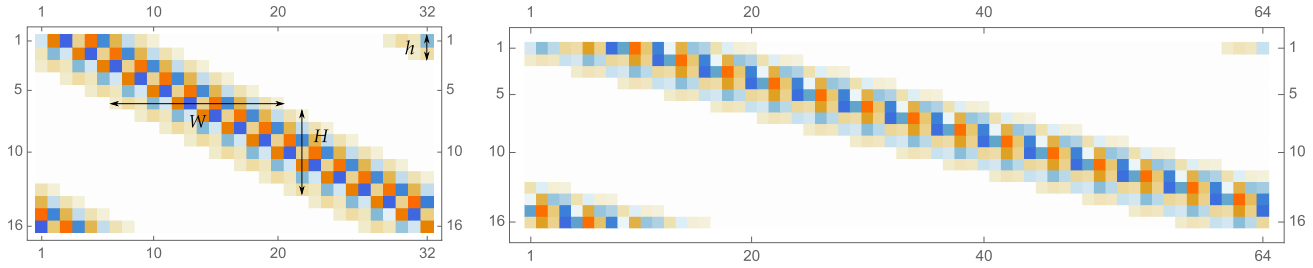

FIG. 8. Visualization of two *ww* blocks of the approximate ICM. The left block is a 2-circulant matrix and the right block is a 4-circulant matrix. Each *ww* block has nonzero elements in the top-right, bottom-left, and main part. The parameter h is the NNZ elements in the last column at the top-right part of the block; W is the bandwidth and H is the vertical bandwidth of the block. These parameters specify the location of nonzero entries in the block.

respectively. We use these formulae in our UDU-decomposition algorithm to compute the nonzero shear elements of the upper unit-triangular matrix $\mathbf{U}$ in the UDU decomposition of the approximate ICM in a multiscale wavelet basis.

In our algorithm for the UDU decomposition of $\tilde{\mathbf{A}}$, we compute an approximation of the upper unit-triangular matrix $\mathbf{U}$. We take the above-diagonal nonzero elements of $\mathbf{U}$ to be in the same position as the above-diagonal nonzero elements of $\tilde{\mathbf{A}}$; we set any entry of $\mathbf{U}$ to zero if the corresponding entry in $\tilde{\mathbf{A}}$ is also zero. We refer to this decomposition as the incomplete UDU decomposition by position.

We specify the location of nonzero entries of the approximate ICM by helper functions. The helper functions return various parameters that specify nonzero entries in each *ww* block of the approximate ICM. As shown in Fig. 8, each *ww* block has nonzero elements in three parts: top-right part (TRP), bottom-left part (BLP), and main part (MP). For each block, we compute the following parameters by the five helper functions in Library 1: (1) the number of nonzero (NNZ) entries in the last column at the TRP of the block; (2) bandwidth of the block, which is the number of cyclically consecutive nonzero entries in each row of the block; (3) vertical bandwidth of the block, which is the number of cyclically consecutive nonzero entries in each column of the block; (4) the column index of the first and last nonzero entries in the block's main part for a given row index; and (5) the row index of the first and last nonzero entries in the block's main part for a given column index. These parameters fully specify the location of nonzero entries in each *ww* block.

We now specify inputs and outputs of our UDU-decomposition algorithm. The algorithm's inputs are

global $\mathcal{K}, w$ ▷ wavelet index $\mathcal{K}$ and upper bandwidth w of the diagonal ww blocks of $\tilde{\mathbf{A}}$ are global variables
function LASTCOLNNZ(r,c) ▷ computes NNZ elements in last column in top-right part of (r,c) block
  return $\left\lceil \frac{w}{2^{c-r}} \right\rceil$
function WIDTH(r,c) ▷ computes bandwidth of (r,c) block
  return $2w+1+(2^{c-r}-1)(2\mathcal{K}-1)$
function VERTWIDTH(r,c) ▷ computes vertical bandwidth of (r,c) block
  return $\left\lceil \frac{\text{WIDTH}(r,c)}{2^{c-r}} \right\rceil$
function MAINCOL(i,r,c) ▷ computes column index of first and last nonzero entry in i^{th} row in main part of (r,c) block
  $h \leftarrow$ LASTCOLNNZ(r,c)
  $W \leftarrow$ WIDTH(r,c)
  $j_{\text{F}} \leftarrow (i-h)2^{c-r} \times \Theta(i-h)$ ▷ $\Theta(n)$ is the unit-step function whose value is zero for $n<0$ and is one for $n \geq 0$
  $j_{\text{L}} \leftarrow \min(j_{\text{F}}+W-1, 2^c-1) \times \Theta(i-h) + \left(W-w+i2^{c-r}-1\right) \times \Theta(h-i-1)$
  return $(j_{\text{F}}, j_{\text{L}})$
function MAINROW(j,r,c) ▷ computes row index of first and last nonzero entry in j^{th} column in main part of (r,c) block
  $h \leftarrow$ LASTCOLNNZ(r,c)
  $W \leftarrow$ WIDTH(r,c)
  $H \leftarrow$ VERTWIDTH(r,c)
  $i_{\text{F}} \leftarrow \left\lceil \frac{j-(W-w-1)}{2^{c-r}} \right\rceil \times \Theta(j-(W-w))$
  $i_{\text{L}} \leftarrow \min(i_{\text{F}}+H-1, 2^r-1) \times \Theta(j-(W-w)) + \left(h+\left\lfloor \frac{j}{2^{c-r}} \right\rfloor\right) \times \Theta((W-w)-j-1)$
  return $(i_{\text{F}}, i_{\text{L}})$

Library 1. Helper functions for location of nonzero elements in blocks of the approximate ICM in multiscale wavelet basis

parameters that specify unique elements of the approximate ICM $\tilde{\mathbf{A}}$: the wavelet index $\mathcal{K}$, the number of modes N in the discretized QFT, the upper bandwidth w for diagonal blocks of $\tilde{\mathbf{A}}$, and the circulant row in unique blocks of $\tilde{\mathbf{A}}$. Outputs are shear elements of the approximate upper unit-triangular matrix $\tilde{\mathbf{U}}$ and diagonals of the diagonal matrix $\mathbf{D}$ in the incomplete UDU decomposition of $\tilde{\mathbf{A}}$.

The approximate upper unit-triangular $\tilde{\mathbf{U}}$ matrix in the incomplete UDU decomposition has the same sparse and block-matrix structure as the matrix $\tilde{\mathbf{A}}$. We exploit these structures and store the shear elements of $\tilde{\mathbf{U}}$ by a sparse representation. Similar to sparse representation of $\tilde{\mathbf{A}}$, we represent shear elements in blocks of $\tilde{\mathbf{U}}$ by an associative array, i.e., a collection of (key, value) pairs. Each key is a tuple (z, r, c), where $z \in \{ss, sw, ww\}$ specifies if the block belongs to the *ss*, *sw*, or *ww* part of the block matrix. The positive integers r and c specify the row and column indices of the block matrix, respectively. The associative array's value specifies the shear elements of the block at entry (z, r, c) of the block matrix. Unlike blocks of $\tilde{\mathbf{A}}$, blocks of $\tilde{\mathbf{U}}$ are not circulant matrices, so we cannot specify each block by a single row of the block. Therefore, each value in the associative array is a matrix that specifies the shear elements of a block in the block matrix.

The *ss* block of $\tilde{\mathbf{U}}$ is a unit-triangular matrix. As illustrated in Fig. 9(b), we store this block's shear elements as a vector of size $2^{s_0}(2^{s_0}-1)/2$ by concatenating the shear elements from top-left to bottom-right corner sequentially row by row; note that the *ss* block is a matrix of size $2^{s_0} \times 2^{s_0}$. Similar to the *sw* blocks of $\tilde{\mathbf{A}}$, the *sw* blocks of $\tilde{\mathbf{U}}$ are not sparse matrices. Therefore, we store the shear elements in these blocks by a matrix of the same size. The diagonal *ww* blocks are each an upper unit-triangular matrix. The shear element of $\tilde{\mathbf{U}}_{ww}^{(r,r)}$ are in the same position of the above-diagonal nonzero elements of $\tilde{\mathbf{A}}_{ww}^{(r,r)}$; see Fig. 9(c). As illustrated in Fig. 9(d), we store the shear elements of $\tilde{\mathbf{U}}_{ww}^{(r,r)}$ by a matrix of size $2^r \times w$, where w is the upper bandwidth of the diagonal *ww* blocks of $\tilde{\mathbf{A}}$. The off-diagonal *ww* blocks of $\tilde{\mathbf{U}}$ are sparse with the same sparse structure as the off-diagonal *ww* blocks of $\tilde{\mathbf{A}}$. Specifically,

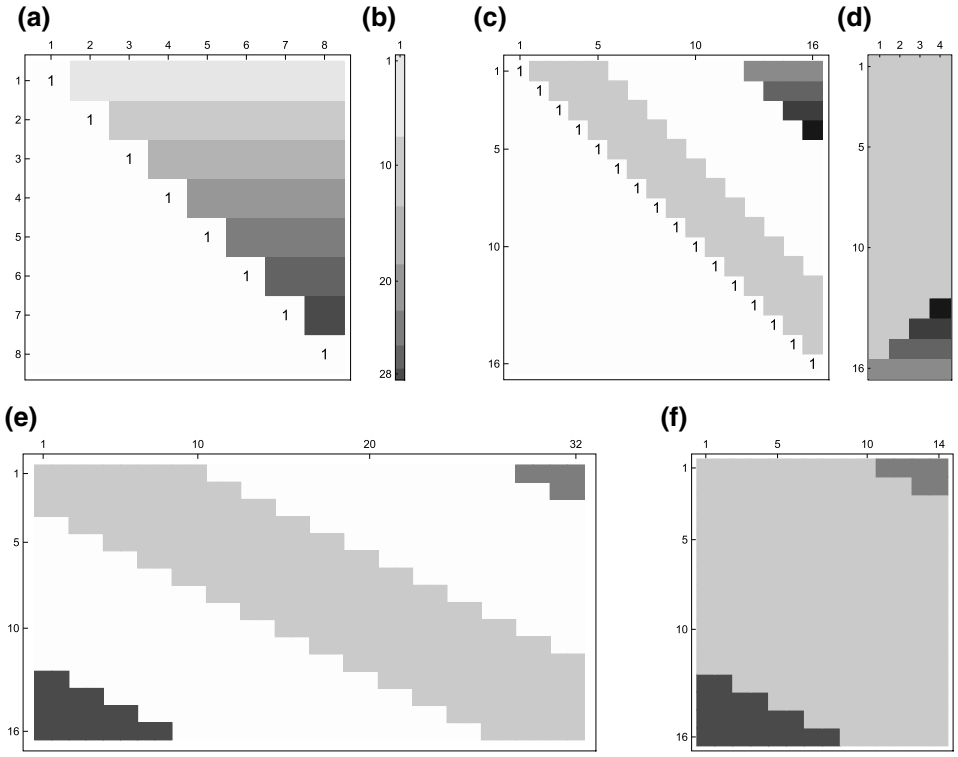

FIG. 9. Schematic illustration of our method for storing shear elements in the *ss* and *ww* blocks of the upper unit-triangular matrix in the incomplete UDU decomposition of the approximate ICM in a multiscale wavelet basis. To elucidate how we store the shear elements, we use gray color with different scales to show the location of nonzero elements in different parts of a block; gray color's scale here does not represent the relative magnitude of the elements (a) Visualization of the *ss* block. This block is an upper unit-triangular matrix. (b) Visualization of the vector storing the *ss* block's shear elements. (c) Visualization of a diagonal *ww* block. This block is also an upper unit-triangular matrix. (d) Visualization of the matrix storing shear elements of the diagonal *ww* block. (e) Visualization of an off-diagonal *ww* block. (f) Visualization of the matrix storing the shear elements of the off-diagonal *ww* block.

the block $\tilde{\mathbf{U}}_{ww}^{(r,c)}$ with $c > r$ is a banded matrix with bandwidth $\text{WIDTH}(r,c)$, Eq. (55). We store shear element of this block by a matrix of size $2^r \times \text{WIDTH}(r,c)$, which is illustrated in Fig. 9(f). For convenience, we use the helper functions in Library 2 for storing the shear elements in the *ss* and *ww* blocks of $\tilde{\mathbf{U}}$.

We now describe the procedure of our UDU-decomposition algorithm. We employ the block variant for the UDU decomposition in our algorithm, but we compute only the nonzero shear elements in each column of $\mathbf{U}$. The *ss* and *sw* blocks of $\mathbf{U}$ are dense matrices, so we compute all elements of these blocks. The *ww* blocks, however, are sparse matrices similar to the *ww* blocks of $\tilde{\mathbf{A}}$. We therefore use the helper functions in Library 1 to specify the location of nonzero elements in the *ww* blocks. We start from the top-right block of $\mathbf{U}$ and proceed to compute the nonzero shear elements in each block of $\mathbf{U}$ column by column. For each column c of the block matrix $\mathbf{U}$, we compute diagonals $d_{ww;\,i}^{(c)}$ of $\mathbf{D}_{ww}^{(c)}$, all elements of $\mathbf{U}_{sw}^{(s_0,c)}$ and nonzero shear elements of $\mathbf{U}_{ww}^{(r,c)}$ for $r \in \{s_0, \ldots, c\}$.

To compute the diagonal element $d_{ww;\,i}^{(c)}$, first we compute nonzero elements in the i^{th} row of $\mathbf{V}_{ww}^{(c,s)}$, Eq. (62), for $s \in \{c, \ldots, k-1\}$. Then we multiply each nonzero element in the i^{th} row of $\mathbf{V}_{ww}^{(c,s)}$ to its corresponding nonzero element in the i^{th} row of $\mathbf{U}^{(c,s)}$ and add the results. Next we negate the result and add $a_{ww;\,i,i}^{(c,c)}$ to obtain $d_{ww;\,i}^{(c)}$. The i^{th} diagonal element in the *ss* block of $\mathbf{D}$ is computed by a similar procedure. In this case, we compute all elements in the i^{th} row of the *ss* and *sw* blocks of $\mathbf{V}$, Eq. (62), because these blocks are dense matrices.

To compute the nonzero shear element at entry (i,j) of a *ww* block at entry (r,c) of the block matrix $\mathbf{U}$, we multiply nonzero elements in the j^{th} row of $\mathbf{V}^{(c,s)}$ by their corresponding entries in the i^{th} row of $\mathbf{U}^{(c,s)}$ for $s \in \{c, \ldots, k-1\}$, as per Eq. (67); note that off-diagonal elements of diagonal blocks and all elements of off-diagonal blocks are shear elements. We then add them all and negate the result. Next we add $a_{ww;\,i,j}^{(r,c)}$ to the obtained value and divide the result by $d_{ww;\,i}^{(c)}$. The final result is the shear element at the (i,j) entry of (r,c) block of $\mathbf{U}$. The shear elements in the *ss* and *sw* blocks are computed by a similar procedure, but being dense matrices, all elements in these blocks must be computed.

D. Quantum algorithms

In this subsection, we construct the quantum routine of our Fourier- and wavelet-based algorithms for ground-state generation. The quantum routine of our algorithms has two subroutines. The first subroutine generates an approximation for a continuous 1DG state, and the second subroutine executes a basis transformation.

We have two quantum algorithms for generating a 1DG state. The first algorithm is presented in Sec. IV D 1, and the second algorithm, which is based on inequality testing, is described in Sec. IV D 2. We present our algorithm for quantum fast Fourier transform in Sec. IV D 3, and the algorithm for quantum shear transform in Sec. IV D 4. The QFFT and QST algorithms serve as the basis-transformation subroutine in the Fourier- and wavelet-based algorithms, respectively.

1. One-dimensional Gaussian-state generation

Here we present our first quantum algorithm for generating a discrete approximation for a continuous 1DG state, Eq. (19), on a quantum register. We begin by explaining the inputs and output of the algorithm. We then describe the involved quantum registers in our algorithm and explain the algorithm's procedure. Finally, we present our algorithm as pseudocode.

The output of our 1DG-state-generation algorithm is a discrete 1DG state with the standard deviation $\tilde{\sigma}$ over a lattice with 2^m points and lattice spacing δ; see Sec. III C 1 for a description of our method for approximating a continuous 1DG state. The discrete 1DG state, Eq. (33), is a linear combination of basis states $|j\,\delta\rangle$, where δ is a real number. Therefore, the discrete 1DG state can be regarded as a superposition of real numbers.

For convenience, we use the fixed-point number representation [45, p. 255] to treat the real numbers in our algorithm. Specifically, we consider each real number as a p-bit number in this representation. The positive integer p is the working precision in our main algorithms for ground-state generation. As the largest value for the real numbers is 2^m, we use lattice parameter m to be the radix-point position, i.e. the number of bits to the left of the radix point in the fixed-point number representation. Therefore, the following parameters are taken as classical inputs to our 1DG-state-generation algorithm: the working precision p, the radix-point position m, and the p-bit numbers $\tilde{\sigma}$ and δ.

To elucidate the procedure of our quantum algorithm, we now describe various quantum registers involved in the algorithm. Our algorithm involves the following quantum registers; all registers start in the all-zero state and have p qubits unless otherwise specified. (1) `out`, a register that is to be prepared in the discrete 1DG state, Eq. (33); (2) `std`, a register that stores the value of $\tilde{\sigma}$; (3) `spc`, a register that stores the value of δ; (4) `mean`, a register that stores a mean value μ; (5) `ang`, a register that stores a single-qubit rotation angle; and (6) `scratch`, a $\mathcal{O}(p)$-qubit register used to assist in various operations throughout the algorithm. We index the qubits of each register from 0 to $p-1$, where the 0^{th} qubit is the rightmost qubit. The first $p-m$ qubits in each p-qubit register represent the fractional part of a number, and the last m qubits represent its integer part in the fixed-point number representation.

The main strategy to generate the discrete 1DG state, Eq. (33), is as follows. First we prepare the state

Input:

$\mathcal{K} \in \mathbb{Z}_{\geq 3}$ ▷ wavelet index

$N \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ ▷ number of modes; for convenience, we assume N is a power of 2

$w \in \mathbb{Z}^{+}$ ▷ upper bandwidth of diagonal ww blocks in the approximate ICM $\tilde{\mathbf{A}}$

$\mathbf{a} := \left\{ \mathbf{a}_{ss}^{(s_0)} \in \mathbb{R}^{2^{s_0}}, \mathbf{a}_{sw}^{(s_0,c)} \in \mathbb{R}^{2^c}, \mathbf{a}_{ww}^{(r,c)} \in \mathbb{R}^{2^{c-r}} \,\middle|\, \lceil \log_2(4\mathcal{K}-2) \rceil =: s_0 \leq r \leq c < k := \log_2 N \right\}$ ▷ circulant rows of ICM $\mathbf{A}$

Output:

$\mathbf{d} \in \mathbb{R}^N$ ▷ diagonals of the diagonal matrix $\mathbf{D}$ in UDU decomposition of $\tilde{\mathbf{A}}$

$\mathbf{S} := \left\{ \mathbf{s}_{ss}^{(s_0)} \in \mathbb{R}^{\frac{1}{2}2^{s_0}(2^{s_0}-1)}, \mathbf{S}_{sw}^{(s_0,c)} \in \mathbb{R}^{2^{s_0}\times 2^c}, \mathbf{S}_{ww}^{(r,c)} \in \mathbb{R}^{2^r \times (\text{WIDTH}(r,c)-(w+1)\delta_{rc})} \,\middle|\, s_0 \leq r \leq c < k \right\}$ ▷ shear elements in main and upper-diagonal blocks of $\mathbf{U}$ in UDU decomposition of $\tilde{\mathbf{A}}$; here $\text{WIDTH}(r,c)$ (55) is bandwidth of $\tilde{\mathbf{A}}_{ww}^{(r,c)}$

function INVCOVUDU($\mathcal{K}, N, w, \mathbf{a}$)

  for $c \leftarrow k-1$ to s_0 **do** ▷ iterates over columns of $\mathbf{U}$

    for $i \leftarrow 2^c - 1$ to 0 **do** ▷ iterates over columns/rows of $\mathbf{U}_{ww}^{(c,c)}$ from right/bottom to left/top

      $d_{ww;i}^{(c)} \leftarrow a_{ww;i,i}^{(c,c)}$

      for $s \leftarrow c$ to $k-1$ **do** ▷ lines (2–14) compute $d_i^{(c)}$ and nonzero elements in i^{th} row of $\mathbf{V}_{ww}^{(c,s)}$

        $(d_{ww;i}^{(c)}, \mathbf{v}_{ww;i}^{(c,s)}) \xleftarrow{-}$ DIAGANDV $\left(i, c, s, \mathbf{u}_{ww;i}^{(c,s)}, \mathbf{d}_{ww}^{(s)}\right)$

      $(i_1, i_2) \leftarrow (0, 2^{s_0}-1)$

      $\mathbf{u}_{sw;i_1:i_2,i}^{(s_0,c)} \leftarrow \mathbf{a}_{sw;i_1:i_2,i}^{(s_0,c)} / d_{ww;i}^{(c)}$

      for $s \leftarrow c$ to $k-1$ **do**

        $\mathbf{u}_{sw;i_1:i_2,i}^{(s_0,c)} \xleftarrow{+}$ MAINSHEARS $\left(i_1, i_2, i, r, c, s, d_{ww;i}^{(c)}, \mathbf{u}_{sw;i_1:i_2}^{(s_0,s)}, \mathbf{v}_{ww;i}^{(c,s)}\right)$

      for $r \leftarrow s_0$ to c **do** ▷ lines (11–26) compute nonzeros in TRP, MP and BLP of $\mathbf{U}_{ww}^{(r,c)}$ $(r < c)$

        if $r \neq c$ **then**

          $(i_F, i_L) \leftarrow$ MAINROW(i, r, c)

          $\mathbf{u}_{ww;i_F:i_L,i}^{(r,c)} \leftarrow \mathbf{a}_{ww;i_F:i_L,i}^{(r,c)} / d_{ww;i}^{(c)}$

          for $s \leftarrow c$ to $k-1$ **do**

            $\mathbf{u}_{ww;i_F:i_L,i}^{(r,c)} \xleftarrow{+}$ MAINSHEARS $\left(i_F, i_L, i, r, c, s, d_{ww;i}^{(c)}, \mathbf{u}_{ww;i_F:i_L}^{(r,s)}, \mathbf{v}_{ww;i}^{(c,s)}\right)$

        if $i \geq 2^c - w$ **then** ▷ lines (18–35) compute nonzeros in TRP of $\mathbf{U}_{ww}^{(r,c)}$

          $(i_F, i_L) \leftarrow \left(0, \text{LASTCOLNNZ}(r,c) - 1 - \left\lfloor \frac{2^c-1-i}{2^{c-r}} \right\rfloor\right)$ ▷ row index of last nonzero in i^{th} column in TRP of $\mathbf{U}_{ww}^{(r,c)}$

          $\mathbf{u}_{ww;i_F:i_L,i}^{(r,c)} \leftarrow \mathbf{a}_{ww;i_F:i_F,i}^{(r,c)} / d_{ww;i}^{(c)}$

          for $s \leftarrow c$ to $k-1$ **do**

            $\mathbf{u}_{ww;i_F:i_L,i}^{(r,c)} \xleftarrow{+}$ TOPRIGHTSHEARS $\left(i_F, i_L, i, r, c, s, d_{ww;i}^{(c)}, \mathbf{u}_{ww;i_F:i_L}^{(r,s)}, \mathbf{v}_{ww;i}^{(c,s)}\right)$

        if $i < \text{WIDTH}(r,c) - w - 2^{c-r}$ and $r \neq c$ **then** ▷ lines (22–42) compute nonzeros in BLP of $\mathbf{U}_{ww}^{(r,c)}$

          $(i_1, i_2) \leftarrow \left(i_F + \left\lfloor \frac{i}{2^{c-r}} \right\rfloor, 2^r - 1\right)$

          $\mathbf{u}_{ww;i_1:i_2,i}^{(r,c)} \leftarrow \mathbf{a}_{ww;i_1:i_2,i}^{(r,c)} / d_i^{(c)}$

          for $s \leftarrow c$ to $k-1$ **do**

            $\mathbf{u}_{ww;i_1:i_2,i}^{(r,c)} \xleftarrow{+}$ BOTTOMLEFTSHEARS$(r, c, i, i_1, i_2, d_i^{(c)}, \mathbf{U}_{ww;i_1:i_2}^{(r,s)}, \mathbf{v}_i^{(c,s)}, \mathcal{K}, w)$

      if $i > 0$ **then** ▷ lines (27–31)) compute nonzeros in MP of $\mathbf{U}_{ww}^{(c,c)}$

        $(i_F, i_L) \leftarrow (\max(0, i-w), i-1)$

        $\mathbf{u}_{ww;i_F:i_L,i}^{(c,c)} \leftarrow \mathbf{a}_{ww;i_F:i_L,i}^{(c,c)} / d_{ww;i}^{(c)}$

        for $s \leftarrow c$ to $k-1$ **do**

          $\mathbf{u}_{ww;i_F:i_L,i}^{(c,c)} \xleftarrow{+}$ MAINSHEARS $\left(i_F, i_L, i, r, c, s, d_{ww;i}^{(c)}, \mathbf{u}_{ww;i_F:i_L}^{(c,s)}, \mathbf{v}_{ww;i}^{(c,s)}\right)$

    for $i \leftarrow 2^{s_0} - 1$ to 0 **do** ▷ lines (32–38) compute d_i and nonzero elements in i^{th} row of $\mathbf{V}_{ss}^{(s_0)}$ and $\mathbf{V}_{sw}^{(s_0,c)}$

      $(j_F, j_L) \leftarrow (i+1, 2^{s_0}-1)$

      $\mathbf{v}_{ss;i,j_F:j_L}^{(s_0)} \leftarrow \mathbf{u}_{ss;i,j_F:j_L}^{(s_0)} \odot \mathbf{d}_{j_F:j_L}$

      $d_i \leftarrow a_{ss;i,i}^{(s_0)} - \mathbf{u}_{ss;i,j_F:j_L}^{(s_0)} \odot \mathbf{v}_{ss;j_F:j_L}^{(s_0)}$

      for $s \leftarrow s_0$ to $k-1$ **do**

        $v_{sw;i,0:2^s-1}^{(s_0,s)} \leftarrow \mathbf{u}_{sw;i,0:2^s-1}^{(s_0,c)} \odot \mathbf{d}_{0:2^s-1}$

        $d_i \xleftarrow{-} \mathbf{u}_{sw;i,0:2^s-1}^{(s_0,s)} \cdot \mathbf{v}_{sw;i,0:2^s-1}^{(s_0,s)}$

      if $i > 0$ **then** ▷ lines (39–42) compute upper-diagonal elements of $\mathbf{U}_{ss}^{(s_0)}$

        $\mathbf{u}_{ss;0:i-1,i}^{(s_0)} \leftarrow \left[a_{ss;0:i-1,i}^{(s_0)} - \mathbf{u}_{ss;0:i-1,i+1:2^{s_0}-1}^{(s_0)} \cdot \mathbf{v}_{ss;i,i+1:2^{s_0}-1}^{(s_0)}\right] / d_i$

        for $s \leftarrow s_0$ to $k-1$ **do**

          $\mathbf{u}_{ss;0:i-1,i}^{(s_0)} \xleftarrow{-} \left[\mathbf{u}_{sw;0:i-1,0:2^s-1}^{(s_0,s)} \cdot \mathbf{v}_{sw;i,0:2^s-1}^{(s_0,s)}\right] / d_i$

    $\mathbf{s}^{(s_0)} \leftarrow$ UPPERUNITRIANGSTORE $\left(2^{s_0}, \mathbf{U}_{ss}^{(s_0)}\right)$ ▷ stores shear elements of $\mathbf{U}_{ss}^{(s_0)}$ as a vector; see line 1 of Library 2

    for $c \leftarrow s_0$ to $k-1$ **do**

      $\mathbf{d}_{2^c:2^{c+1}-1} \leftarrow \mathbf{d}_{ww;0:2^c-1}^{(c)}$

      $\mathbf{S}_{sw}^{(s_0,c)} \leftarrow \mathbf{U}_{sw}^{(s_0,c)}$

      $\mathbf{S}_{ww}^{(c,c)} \leftarrow$ DIAGSTORE $\left(c, w, \mathbf{U}_{ww}^{(c,c)}\right)$ ▷ stores shear elements of $\mathbf{U}_{ww}^{(c,c)}$ as a matrix; see line 6 of Library 2

    for $r \leftarrow s_0$ to $k-1$ **do**

      for $c \leftarrow r+1$ to $k-1$ **do**

        $\mathbf{S}_{ww}^{(r,c)} \leftarrow$ OFFDIAGSTORE $\left(r, c, \mathcal{K}, w, \mathbf{U}_{ww}^{(r,c)}\right)$ ▷ stores shear elements of $\mathbf{U}_{ww}^{(r,c)}$ as a matrix; see line 13 of Library 2

    return $(\mathbf{d}, \mathbf{S})$

Algorithm 5. Classical algorithm for UDU decomposition of the approximate ICM in multiscale wavelet basis

function UPPERUNITRIANGSTORE($N, \mathbf{U}$) ▷ stores shear elements of an $N \times N$ upper unit-triangular matrix $\mathbf{U}$
  for $i \leftarrow 0$ to $N-2$ **do**
    for $j \leftarrow i+1$ to $N-1$ **do**
      $s_{iN-i(i+1)/2+j-(i+1)} \leftarrow u_{ij}$
  return s
function DIAGSTORE($c, \mathbf{U}_{ww}^{(c,c)}$) ▷ stores shear elements of a diagonal ww block of $\mathbf{U}$
  for $i \leftarrow 0$ to 2^c-1 **do** ▷ iterates over rows of $\mathbf{U}_{ww}^{(c,c)}$
    $(j_{\mathrm{FM}}, j_{\mathrm{LM}}) \leftarrow (i+1, \min(i+w, 2^c-1))$ ▷ computes column index of first and last shear elements of i^{th} row in MP of $\mathbf{U}_{ww}^{(c,c)}$
    $\mathbf{s}_{ww;i,0:j_{\mathrm{LM}}-j_{\mathrm{FM}}}^{(c,c)} \leftarrow \mathbf{u}_{ww;i,j_{\mathrm{FM}}:j_{\mathrm{LM}}}^{(c,c)}$
    if $i \leq w-1$ **then** ▷ $w-1$ is row index of last nonzero in last column of $\mathbf{U}_{ww}^{(c,c)}$
      $\mathbf{s}_{ww;2^c-1-i,i:w-1}^{(c,c)} \leftarrow \mathbf{u}_{ww;i,2^c-w-i:2^c-1}^{(c,c)}$ ▷ stores shear elements of i^{th} row in TRP of $\mathbf{U}_{ww}^{(c,c)}$ into $(2^c-1-i)^{\mathrm{th}}$ row of $\mathbf{S}_{ww}^{(c,c)}$
  return $\mathbf{S}_{ww}^{(c,c)}$
function OFFDIAGSTORE($r, c, \mathbf{U}_{ww}^{(r,c)}$) ▷ stores shear elements of the off-diagonal ww block at entry (r,c) of $\mathbf{U}$
  $h \leftarrow$ LASTCOLNNZ(r,c)
  $H \leftarrow$ VERTWIDTH(r,c)
  $W \leftarrow$ WIDTH(r,c)
  $(i_{\mathrm{LT}}, i_{\mathrm{FB}}) \leftarrow (h-1, 2^r-(H-h-1))$ ▷ i_{LT} is row index of last column's last nonzero in TRP and i_{FB} is row index of first column's first nonzero in BLP of $\mathbf{U}_{ww}^{(r,c)}$
  for $i \leftarrow 0$ to 2^r-1 **do** ▷ iterates over rows of $\mathbf{U}_{ww}^{(r,c)}$
    $(j_{\mathrm{FM}}, j_{\mathrm{LM}}) \leftarrow$ MAINCOL(i,r,c) ▷ computes column index of first and last nonzero in i^{th} row in MP of $\mathbf{U}_{ww}^{(r,c)}$
    if $i \leq i_{\mathrm{LT}}$ **then**
      $\mathbf{s}_{ww;i,0:j_{\mathrm{LM}}}^{(r,c)} \leftarrow \mathbf{u}_{ww;i,0:j_{\mathrm{LM}}}^{(r,c)}$ ▷ note that $j_{\mathrm{FM}} = 0$ for $i < i_{\mathrm{LT}}$
      $(j_{\mathrm{FT}}, j_{\mathrm{LT}}) \leftarrow (2^c - w + i2^{c-r}, 2^c-1)$ ▷ computes column index of first and last nonzero in i^{th} row in TRP of $\mathbf{U}_{ww}^{(r,c)}$
      $\mathbf{s}_{ww;i,j_{\mathrm{LM}}+1:W-1}^{(r,c)} \leftarrow \mathbf{u}_{ww;i,j_{\mathrm{FT}}:j_{\mathrm{LT}}}^{(r,c)}$
    else if $i_{\mathrm{LT}} < i < i_{\mathrm{FB}}$ **then**
      $\mathbf{s}_{ww;i,0:W-1}^{(r,c)} \leftarrow \mathbf{u}_{ww;i,j_{\mathrm{FM}}:j_{\mathrm{LM}}}^{(r,c)}$
    else
      $(j_{\mathrm{FB}}, j_{\mathrm{LB}}) \leftarrow (0, (i-i_{\mathrm{FB}})2^{c-r}-1)$ ▷ computes column index of first and last nonzero in i^{th} row in BLP of $\mathbf{U}_{ww}^{(r,c)}$
      $\mathbf{s}_{ww;i,0:j_{\mathrm{LB}}}^{(r,c)} \leftarrow \mathbf{u}_{ww;i,0:j_{\mathrm{LB}}}^{(r,c)}$
      $\mathbf{s}_{ww;i,j_{\mathrm{LB}}+1:W-1}^{(r,c)} \leftarrow \mathbf{u}_{ww;i,j_{\mathrm{FM}}:j_{\mathrm{LM}}}^{(r,c)}$
  return $\mathbf{S}_{ww}^{(r,c)}$

Library 2. Helper functions for storing shear elements of the upper unit-triangular matrix in UDU decomposition

$$|\Psi(\tilde{\sigma},\mu,m)\rangle := \frac{1}{\sqrt{f\ (\tilde{\sigma},\mu,m)}} \sum_{j=-2^{m-1}}^{2^{m-1}-1} e^{-\frac{(j+\mu)^2}{4\tilde{\sigma}^2}} |j\rangle\,,$$
$$f\ (\tilde{\sigma},\mu,m) := \sum_{j=-2^{m-1}}^{2^{m-1}-1} e^{-\frac{(j+\mu)^2}{2\tilde{\sigma}^2}}, \tag{69}$$

with the initial value $\mu = 0$ on the leftmost m qubits of the p-qubit `out` register, i.e., the qubits that represent the integer part of a number. This state is a discrete 1DG state with the mean value μ over a lattice with unit spacing. The parameter μ is used here because our algorithm for generating the state in Eq. (69) is iterative, and the value of μ changes in each iteration. Having prepared the $|\Psi(\tilde{\sigma},0,m)\rangle$, we then transform it into the discrete 1DG state $|G_{\mathrm{lattice}}(\tilde{\sigma},\delta,m)\rangle$ by multiplying j to δ. To this end, we write the classical input δ into the `spc` register and implement the transformation $|j\rangle_{\mathtt{out}}|\delta\rangle_{\mathtt{spc}} \mapsto |j\,\delta\rangle_{\mathtt{out}}|\delta\rangle_{\mathtt{spc}}$. The state prepared on the `out` register is then $|G(\tilde{\sigma},\delta,m)\rangle$.

We now proceed with a detailed description of the procedure for generating $|\Psi(\tilde{\sigma},\mu,m)\rangle$. This state is a linear combination of basis states $|j\rangle$ where j is a signed integer. Using two's complement to represent signed integers [45, p. 16], we recursively decompose the state as

$$\begin{aligned}&|\Psi(\tilde{\sigma}_\ell,\mu_\ell,m_\ell)\rangle\\ &\quad= |\Psi(\tilde{\sigma}_\ell/2,\mu_\ell/2,m_\ell-1)\rangle\otimes\cos\theta_\ell\,|0\rangle\\ &\qquad+|\Psi(\tilde{\sigma}_\ell/2,(\mu_\ell+1)/2,m_\ell-1)\rangle\otimes\sin\theta_\ell\,|1\rangle\,,\end{aligned} \tag{70}$$

where we define the four terms $\tilde{\sigma}_0 := \tilde{\sigma}$, $\mu_0 := \mu$, $m_\ell := m-\ell$ and $\forall \ell \in \{0,\ldots,m-1\}$

$$\theta_\ell := \arccos\sqrt{\frac{f\ (\tilde{\sigma}_\ell/2,\mu_\ell/2,m_\ell-1)}{f\ (\tilde{\sigma}_\ell,\mu_\ell,m_\ell)}}. \tag{71}$$

function DIAGANDV($i, c, s, \mathbf{u}^{(c,s)}_{ww;i}, \mathbf{d}^{(s)}_{ww}$)
  $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow$ MAINCOL(i, c, s)
  $\mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \leftarrow \mathbf{u}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \odot \mathbf{d}^{(s)}_{ww;j_{\mathrm{F}}:j_{\mathrm{L}}}$
  $d^{(c)}_{ww;i} \overset{+}{\leftarrow} \mathbf{u}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}$ ▷ $\overset{+}{\leftarrow}$ denotes the addition assignment, i.e., $a \overset{+}{\leftarrow} b := a \leftarrow a + b$
  $(h, H) \leftarrow$ (LASTCOLNNZ(c, s), VERTWIDTH(c, s))
  $(i_{\mathrm{LT}}, i_{\mathrm{FB}}) \leftarrow (h - 1, 2^c - (H - h - 1))$
  if $i \le i_{\mathrm{LT}}$ **then** ▷ i_{L} is row index of last nonzero in last column in TRP of $\mathbf{U}^{(c,s)}_{ww}$
    $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow (2^s - w + i2^{s-c}, 2^s - 1)$ ▷ j_{F} is column index of first nonzero in i^{th} row in TRP of $\mathbf{U}^{(r,c)}_{ww}$
    $\mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \leftarrow \mathbf{u}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \odot \mathbf{d}^{(s)}_{ww;j_{\mathrm{F}}:j_{\mathrm{L}}}$
    $d^{(c)}_{ww;i} \overset{+}{\leftarrow} \mathbf{u}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}$ ▷ multiplies TRP of $\mathbf{U}^{(c,s)}_{ww}$ by TRP of $\mathbf{V}^{(c,s)}_{ww}$
  if $i \ge i_{\mathrm{FB}}$ and $s \ne c$ **then** ▷ i_{F} is row index of first nonzero in first column in BLP
    $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow (0, (i - i_{\mathrm{FB}})2^{s-c} - 1)$ ▷ j_{L} is column index of last nonzero in i^{th} row in BLP of $\mathbf{U}^{(c,s)}_{ww}$
    $\mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \leftarrow \mathbf{u}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \odot \mathbf{d}^{(s)}_{ww;j_{\mathrm{F}}:j_{\mathrm{L}}}$
    $d^{(c)}_{ww;i} \overset{+}{\leftarrow} \mathbf{u}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}$ ▷ multiplies BLP of $\mathbf{U}^{(c,s)}_{ww}$ by BLP of $\mathbf{V}^{(c,s)}_{ww}$
  return $(d^{(c)}_{ww;i}, \mathbf{v}^{(c,s)}_{ww;i})$
function MAINSHEARS($i_1, i_2, i, r, c, s, d^{(c)}_{ww;i}, \mathbf{u}^{(r,s)}_{ww;i_1:i_2}, \mathbf{v}^{(c,s)}_{ww;i}$)
  $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow$ MAINCOL(i, c, s)
  $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i} \overset{+}{\leftarrow} \left[\mathbf{u}^{(r,s)}_{ww;i_1:i_2,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}\right] / d^{(c)}_{ww;i}$ ▷ contribution to shear elements from multiplying MP of $\mathbf{U}^{(r,s)}_{ww}$ to MP of $\mathbf{V}^{(c,s)}_{ww}$
  $(h, H) \leftarrow$ (LASTCOLNNZ(c, s), VERTWIDTH(c, s))
  $(i_{\mathrm{LT}}, i_{\mathrm{FB}}) \leftarrow (h - 1, 2^c - (H - h - 1))$
  if $i \le i_{\mathrm{LT}}$ **then** ▷ i_{L} is row index of last nonzero in last column in TRP of $\mathbf{V}^{(c,s)}_{ww}$
    $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow (2^s - w + i2^{s-c}, 2^s - 1)$ ▷ j_{F} is column index of first nonzero in j^{th} row in TRP of $\mathbf{V}^{(c,s)}_{ww}$
    $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i} \overset{+}{\leftarrow} \left[\mathbf{u}^{(r,s)}_{ww;i_1:i_2,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}\right] / d^{(c)}_{ww;i}$ ▷ contribution from multiplying TRP of $\mathbf{U}^{(r,s)}_{ww}$ to TRP of $\mathbf{V}^{(c,s)}_{ww}$
  if $i \ge i_{\mathrm{FB}}$ and $s \ne c$ **then** ▷ $s \ne c$ as $\mathbf{V}^{(c,c)}_{ww}$ does not have BLP part
    $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow (0, (i - i_{\mathrm{FB}})2^{s-c} - 1)$
    $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i} \overset{+}{\leftarrow} \left[\mathbf{u}^{(r,s)}_{ww;i_1:i_2,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}\right] / d^{(c)}_{ww;i}$ ▷ contribution to shear elements from multiplying BL of $\mathbf{U}^{(r,s)}_{ww}$ to BLP of $\mathbf{V}^{(c,s)}_{ww}$
  return $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i}$
function TOPRIGHTSHEARS($i_1, i_2, i, r, c, s, d^{(c)}_{ww;i}, \mathbf{u}^{(r,s)}_{ww;i_1:i_2}, \mathbf{v}^{(c,s)}_{ww;i}$)
  $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow$ MAINCOL(i, c, s)
  $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i} \overset{+}{\leftarrow} \left[\mathbf{u}^{(r,s)}_{ww;i_1:i_2,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}\right] / d^{(c)}_{ww;i}$ ▷ contribution to shear elements from multiplying TRP of $\mathbf{U}^{(r,s)}_{ww}$ to MP of $\mathbf{V}^{(c,s)}_{ww}$
  $(h, H) \leftarrow$ (LASTCOLNNZ(c, s), VERTWIDTH(c, s))
  $i_{\mathrm{FB}} \leftarrow 2^c - (H - h - 1)$ ▷ i_{F} is row index of first nonzero in first column in BLP of $\mathbf{V}^{(c,s)}_{ww}$
  if $i \ge i_{\mathrm{FB}}$ and $s \ne c$ **then** ▷ BLP of $\mathbf{V}^{(c,s)}_{ww}$ has nonzeros only for rows $i > i_{\mathrm{F}}$; $s \ne c$ as $\mathbf{V}^{(c,c)}_{ww}$ does not have BLP part
    $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow (0, (i - i_{\mathrm{FB}})2^{s-c} - 1)$
    $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i} \overset{+}{\leftarrow} \left[\mathbf{u}^{(r,s)}_{ww;i_1:i_2,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}\right] / d^{(c)}_{ww;i}$ ▷ multiplies TLP of $\mathbf{U}^{(s,r)}_{ww}$ by BLP of $\mathbf{V}^{(k,r)}_{ww}$
  return $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i}$
function BOTTOMLEFTSHEARS($i_1, i_2, i, r, c, s, d^{(c)}_{ww;i}, \mathbf{u}^{(r,s)}_{ww;i_1:i_2}, \mathbf{v}^{(c,s)}_{ww;i}$)
  $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow$ MAINCOL(i, c, s)
  $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i} \overset{+}{\leftarrow} \left[\mathbf{u}^{(r,s)}_{ww;i_1:i_2,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}\right] / d^{(c)}_{ww;i}$ ▷ contribution to shear elements from multiplying BLP of $\mathbf{U}^{(r,s)}_{ww}$ to MP of $\mathbf{V}^{(c,s)}_{ww}$
  if $i \le i_{\mathrm{TL}} \leftarrow$ LASTCOLNNZ$(c, s, w) - 1$ **then**
    $(j_{\mathrm{F}}, j_{\mathrm{L}}) \leftarrow (2^s - w + i2^{s-c}, 2^s - 1)$
    $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i} \overset{+}{\leftarrow} \left[\mathbf{u}^{(r,s)}_{ww;i_1:i_2,j_{\mathrm{F}}:j_{\mathrm{L}}} \cdot \mathbf{v}^{(c,s)}_{ww;i,j_{\mathrm{F}}:j_{\mathrm{L}}}\right] / d^{(c)}_{ww;i}$ ▷ contribution to shear elements from multiplying MP of $\mathbf{U}^{(r,s)}_{ww}$ to TRP of $\mathbf{V}^{(c,s)}_{ww}$
  return $\mathbf{u}^{(r,c)}_{ww;i_1:i_2,i}$

Library 3. Helper functions for computing shear elements

We use the recursive formula (70) to devise an iterative algorithm for generating $|\Psi(\tilde{\sigma}, \mu, m)\rangle$. We start by writing $\tilde{\sigma}$ into the `std` register and μ into the `mean` register. For ℓ from 0 to $m-1$, we iteratively perform the following operations; see the quantum circuit in Fig. 10.

```
function ITERONEDG(ℓ, m, p, std, mean, ang)        ▷ performs the iterative steps described in Sec. IV D 1
    ANGLE(std, mean, ang)                          ▷ computes θ_ℓ/2π (71) into the ang register as per Eq. (72)
    ROT(ang, out[ℓ])                               ▷ rotates the ℓ^th qubit of out register as per Eq. (73)
    ANGLE(std, mean, ang)                          ▷ erases ang register
    SHIFT(std)                                     ▷ along with the next two lines, divides the number stored in std by two
    if σ̃_ℓ = 1 then
        X(std[p − 1])                              ▷ flips the leftmost qubit of std
    SHIFT(mean)                                    ▷ divides the number stored in mean by two; the leftmost qubit of mean is always |0⟩
    CNOT(out[ℓ], mean[p − m − 1])                  ▷ adds 1/2 to mean register if out[ℓ] is |1⟩
    yield out[ℓ]
```

Library 4. Helper function for 1DG-state generation in Algorithm 6

1. Compute a p-bit approximation for $\eta_\ell := \theta_\ell/2\pi$, Eq. (71), and store the result in the ang register. We perform this operation by

$$\text{ANGLE} : |\tilde{\sigma}\rangle_{\texttt{std}} |\mu\rangle_{\texttt{mean}} |0\rangle_{\texttt{ang}} \mapsto |\tilde{\sigma}\rangle_{\texttt{std}} |\mu\rangle_{\texttt{mean}} |\eta\rangle_{\texttt{ang}}, \tag{72}$$

   which we describe by by ANGLE(std, mean, ang) in our quantum algorithm.
2. Perform a single-qubit rotation on out[ℓ], the ℓ^{th} qubit of out, where the angle of rotation is read from the ang register. The rotation is performed by implementing the operation

$$\text{ROT} : |\eta\rangle_{\texttt{ang}} |0\rangle_{\texttt{out}[\ell]} \mapsto |\eta\rangle_{\texttt{ang}} \left(\cos(2\pi\eta) |0\rangle_{\texttt{out}[\ell]} + \sin(2\pi\eta) |1\rangle_{\texttt{out}[\ell]}\right), \tag{73}$$

   which we describe by ROT(ang, out[ℓ]) in our algorithm.
3. Erase ang by uncomputing η_ℓ. We uncompute η_ℓ by performing the ANGLE, Eq. (72), operation.
4. Divide the numbers stored in std and mean by two. To divide the number stored in std by two, we cyclically shift the qubits of this register one qubit to the right by performing

$$\text{SHIFT} : |b_{p-1} \dots b_1 b_0\rangle_{\texttt{out}} \mapsto |b_0 b_{p-1} \dots b_1\rangle_{\texttt{out}}, \tag{74}$$

   and then flip the leftmost qubit of std if the ℓ^{th} bit of the classical input $\tilde{\sigma}$ is 1. As we start by the initial value $\mu = 0$, the rightmost qubit of mean remains in the zero state throughout the computation. Therefore, to divide the number stored in mean by two, we perform only SHIFT, Eq. (74), on this register.
5. Add $1/2$ to mean if the state of out[ℓ] is $|1\rangle$. As we start by $\mu = 0$, after dividing the value encoded in mean by two in the previous step, the $(p - m - 1)^{\text{th}}$ qubit of this register is $|0\rangle$. Therefore, we add $1/2$ to mean by flipping the state of this qubit from $|0\rangle$ to $|1\rangle$.

To simplify the readability of our 1DG-state-generation algorithm, we describe these iterative operations by a helper function in Library 4.

By performing the described iterative operations, the quantum state $|\Psi(\tilde{\sigma}, 0, m)\rangle$, Eq. (69), is prepared on the first m qubits of the out register. To transform this state into the integer part of out, we swap the first m qubits

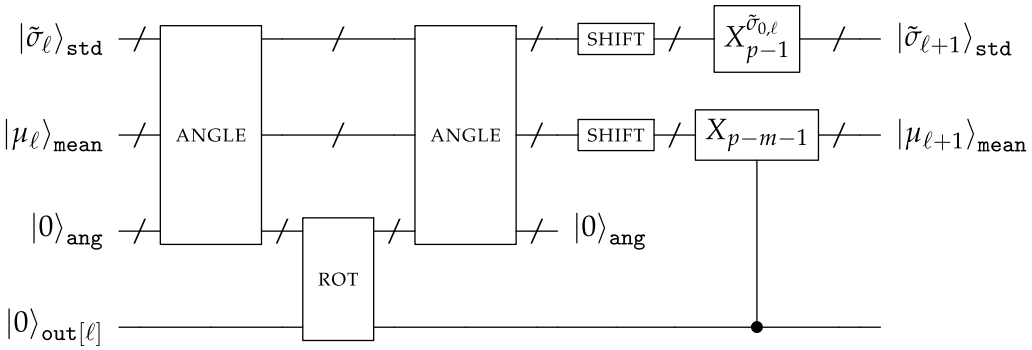

FIG. 10. Quantum circuit for implementing the iterative steps for 1DG-state generation. Positive integers p and m are, respectively, the total number of bits and position of the radix point in the fixed-point number representation; $+$ represents p qubits. ANGLE, Eq. (72), computes a rotation angle, Eq. (71), into the ang register, ROT, Eq. (73), rotates the ℓ^{th} qubit of out, out[ℓ], by the rotation angle stored in ang and SHIFT, Eq. (74), cyclically shifts qubits of a register one qubit to the right. The Pauli-X gate X_i^b acts on i^{th} qubit of a register if the binary b is 1. Intermediate results are uncomputed by running the appropriate operations in reverse order.

Input:
$p \in \mathbb{Z}^+$ ▷ working precision
$m \in \mathbb{Z}^+$ ▷ position of the radix point in fixed-point number representation
$\tilde{\sigma} \in \mathbb{B}^p$ ▷ p-bit approximate standard deviation of the approximate 1DG pure state
$\delta \in \mathbb{B}^p$ ▷ p-bit approximate lattice spacing
Output:
$\mathtt{out} \in \mathscr{H}_2^p$ ▷ p-qubit approximate 1DG state (33) prepared on out register
1: **function** ONEDG($p, m, \tilde{\sigma}, \delta$)
2: $\mathscr{H}_2^p \ni \mathtt{std} \leftarrow \tilde{\sigma}$ ▷ encodes $\tilde{\sigma}$ into the std register
3: $\mathscr{H}_2^p \ni \mathtt{spc} \leftarrow \delta$ ▷ encodes δ into the spc register
4: **for** $\ell \leftarrow 0$ to $m-1$ **do**
5: ITERONEDG($\ell, m, p, \mathtt{std}, \mathtt{mean}, \mathtt{ang}$) ▷ see the helper function in Library 4
6: **for** $\ell \leftarrow 0$ to $m-1$ **do**
7: SWAP($\mathtt{out}[\ell], \mathtt{out}[p-m+\ell]$)
8: MUL($\mathtt{out}, \mathtt{spc}, \mathtt{tmp}$)
9: **for** $\ell \leftarrow 0$ to $m-1$ **do** ▷ Lines (9–12) are the reverse of lines (4–7) and uncomputes out
10: SWAP($\mathtt{out}[\ell], \mathtt{out}[p-m+\ell]$)
11: **for** $\ell \leftarrow m-1$ to 0 **do**
12: INVITERONEDG($\ell, m, p, \mathtt{std}, \mathtt{mean}, \mathtt{ang}$) ▷ preforms inverse of the helper function in line 5
13: **for** $\ell \leftarrow 0$ to $p-1$ **do** ▷ swaps the qubits of out and tmp registers
14: SWAP($\mathtt{out}[\ell], \mathtt{tmp}[\ell]$)
15: $\mathtt{std} \leftarrow \tilde{\sigma}$ ▷ erases std register
16: $\mathtt{spc} \leftarrow \delta$ ▷ erases spc register
17: **yield** out

Algorithm 6. Quantum algorithm for generating a one-dimensional Gaussian state

with the last m qubits of out. Specifically, we swap the ℓ^{th} qubit of out with the $(p-m+\ell)^{\text{th}}$ qubit for $\ell \in \{0, \ldots, m-1\}$.

The last step is to transform $|\Psi(\tilde{\sigma}, 0, m)\rangle$ to the desired discrete 1DG state $|G_{\text{lattice}}(\tilde{\sigma}, \delta, m)\rangle$, Eq. (33). To this end, we write the classical input δ into the spc register and perform the operation

$$\text{MUL} : |j\rangle_{\mathtt{out}} |\delta\rangle_{\mathtt{spc}} |0\rangle_{\mathtt{tmp}} \mapsto |j\rangle_{\mathtt{out}} |\delta\rangle_{\mathtt{spc}} |j\,\delta\rangle_{\mathtt{tmp}}, \tag{75}$$

which we describe by MUL(out, spc, tmp) in our algorithm. This operation is an out-of-place multiplication and, therefore, we need to uncompute the out register. To uncompute out, we perform the operations that generates $|\Psi(\tilde{\sigma}, 0, m)\rangle$ in the reverse order. We then swap qubits of tmp with qubits of out. By this swapping, tmp is erased and $|G(\tilde{\sigma}, \delta, m)\rangle$ is transformed into out.

Finally, we erase the spc and std registers. Note that the values stored in these registers do not change throughout the computation. Therefore, we erase spc and std by writing the classical inputs δ and $\tilde{\sigma}$ into them, respectively. The full description of our 1DG-state-generation algorithm is presented as pseudocode in Algorithm 6.

2. One-dimensional Gaussian-state generation by inequality testing

We now construct our alternative quantum algorithm for generating a 1DG state. Our algorithm is based on testing an inequality on a quantum computer. We begin with a high-level description for state generation by inequality testing and our algorithm for 1DG-state generation. Then we proceed with a detailed description of the algorithm, and finally, we present our algorithm as pseudocode.

The general principle to prepare a state of the form $\sum_j f(j)\,|j\rangle$ by inequality testing is as follows [15]. First prepare an equal superposition over $|j\rangle$. Then, for each j, compute $f(j)$ into a new quantum register. Next perform an inequality test between the value encoded in this register and the value encoded in an ancillary register prepared in an equal superposition over all possible values of the function. Then erase the ancillary register and measure the qubit storing result of the inequality test. The prepared state by this method is the desired state with certain probability. The success probability can then be boosted by amplitude amplification [24]. The complexity of this approach could be large in the case where the distribution of amplitudes has a sharp peak in an unknown location because the amplitude amplification would then essentially be solving a Grover search, which has a square-root speed limit [24].

The amplitude distribution for a Gaussian state has a single peak but in a known location, and we take advantage of the known location in preparing a Gaussian state. Our approach for using the known location is similar to that used for preparing a state with amplitudes $1/\|k_v\|$ in Ref. [46]. Instead of initially preparing a state with an equal superposition over j, we prepare a state with approximate amplitudes upper bounding the amplitudes to be prepared, as shown by the orange points in Fig. 5. Instead of computing $f(j)$, we compute the ratio between $f(j)$ and

the initial upper bound on $f(j)$ and perform an inequality test with that ratio. The inequality test corrects the amplitudes and results in a much larger amplitude for success. Hence only a single step of amplitude amplification can be used, which significantly reduces the algorithm's complexity.

We now proceed with a detailed description of the inequality-testing-based algorithm for Gaussian-state generation. The state that we aim to prepare by inequality testing is

$$|\Psi(\sigma,m)\rangle := \frac{1}{\sqrt{\mathcal{N}(\sigma,m)}} \sum_{j=-(2^{m-1}-1)}^{2^{m-1}-1} e^{-\frac{j^2}{4\sigma^2}} |j\rangle,$$
$$\mathcal{N}(\sigma,m) := 1 + 2\sum_{j=1}^{2^{m-1}-1} e^{-\frac{j^2}{2\sigma^2}}, \tag{76}$$

with classical inputs $m \in \mathbb{Z}^+$ and $\sigma \in \mathbb{R}^+$.

Having prepared this state on a quantum register, labeled `out`, we then incorporate a lattice spacing $\delta \in \mathbb{R}^+$ by implementing $|j\rangle_{\texttt{out}} \mapsto |j\delta\rangle_{\texttt{out}}$ as per Eq. (75). Thereby an approximation for the desired 1DG state, Eq. (33), is prepared on `out`. Considering the range of the index j in Eq. (76) and Eq. (33), the infidelity between the approximate and the desired 1DG states is exponentially close to zero.

Our strategy to prepare the state in Eq. (76) by inequality testing is as follows. First we prepare the state

$$|\Psi_+(\sigma,m)\rangle = \frac{1}{\sqrt{\mathcal{N}(\sigma,m)}} \left(|0\rangle + \sqrt{2}\sum_{j=1}^{2^{m-1}-1} e^{-\frac{j^2}{4\sigma^2}} |j\rangle\right), \tag{77}$$

on m qubits. Controlled on $j \neq 0$, we then perform a Hadamard on the leftmost qubit. The controlled operation gives a sign bit for j being a signed integer with positive and negative values. Then we convert from signed integer to two's complement representation [45, p. 16] to obtain the state in Eq. (76).

To prepare the state in Eq. (77), we use an initial amplitude according to the value of j rounded down to the nearest power of 2; the initial amplitude is illustrated by orange points in Fig. 5. Specifically, the initial state that we prepare is

$$\left|\tilde{\Psi}_+(\sigma,m)\right\rangle = \frac{1}{\sqrt{\tilde{\mathcal{N}}(\sigma,m)}} \left(|0\rangle + \sqrt{2}\sum_{j=1}^{2^{m-1}-1} e^{-2^{2\lfloor\log_2 j\rfloor}/4\sigma^2} |j\rangle\right),$$
$$\tilde{\mathcal{N}}(\sigma,m) := 1 + 2\sum_{j=1}^{2^{m-1}-1} e^{-2^{2\lfloor\log_2 j\rfloor}/2\sigma^2}, \tag{78}$$

where $\tilde{\mathcal{N}}(\sigma,m)$ is the state's normalization factor. Having prepared this initial state, we then transform it to the state in Eq. (77) by inequality testing. To prepare the initial state, first we prepare the state

$$\frac{1}{\sqrt{\mathcal{M}(\sigma,m)}} \left(|0\rangle^{\otimes m} + \sum_{j=1}^{m-1} 2^{j/2} e^{-2^{2(j-1)}/4\sigma^2} |0\rangle^{\otimes(m-j)} |1\rangle^{\otimes j}\right),$$
$$\mathcal{M}(\sigma,m) := 1 + \sum_{j=1}^{m-1} 2^j e^{-2^{2(j-1)}/2\sigma^2}, \tag{79}$$

on an m-qubit register `G`. Then we sequentially perform a CNOT followed by a controlled Hadamard (CHAD) from `G`$[\ell]$ to `G`$[\ell-1]$ for ℓ from 1 to $m-1$; note that quits are ordered from right to left, so the rightmost qubit is 0^{th} qubit. Upon performing these operations, the initial state, Eq. (78), is generated on register `G`.

We now describe how to transform the initial state, Eq. (78), into the state in Eq. (77) by inequality testing. Let us define $r_0 := 1$,

$$r_j := \exp\left(\frac{2^{2\lfloor\log_2 j\rfloor} - j^2}{4\sigma^2}\right) \quad \forall j \in \{j = 1,\ldots,2^m - 1\}, \tag{80}$$

for the ratio of the unnormalized amplitudes from Eq. (77) to the unnormalized amplitudes from Eq. (78). For some positive integer t, we compute a t-bit approximation of r_j into a t-qubit temporary register, labeled `tmp`, by an operation defined as

$$\text{RATIO} : |j\rangle_{\texttt{out}} |\sigma\rangle_{\texttt{std}} |0\rangle_{\texttt{tmp}} \mapsto |j\rangle_{\texttt{out}} |\sigma\rangle_{\texttt{std}} |r_j\rangle_{\texttt{tmp}}, \tag{81}$$

which we describe by RATIO(`out`, `std`, `tmp`) in our quantum algorithm. As the ratio $r_j \leq 1$ for each j, the encoded t-bit string in `tmp` represents the value of r_j, Eq. (80), with an implied division by 2^t. Next we prepare a t-qubit reference register, labeled `ref`, in the uniform superposition state $2^{-t/2}\sum_{z=0}^{2^t-1}|z\rangle$ using t Hadamard gates. With an implied division of the encoded value by 2^t, the register `ref` can be viewed as being prepared in a uniform superposition of all possible values from 0 to 1. Finally, we test an inequality between the value encoded in `tmp` and the value encoded in `ref` with the result of the inequality test written to a fresh qubit labeled `ineq`. Specifically, we perform a comparison operation defined as

$$\text{COMP} : |r\rangle_{\texttt{tmp}} |z\rangle_{\texttt{ref}} |0\rangle_{\texttt{ineq}} \mapsto \begin{cases} |r\rangle_{\texttt{tmp}} |z\rangle_{\texttt{ref}} |0\rangle_{\texttt{ineq}} & \text{if } r < z, \\ |r\rangle_{\texttt{tmp}} |z\rangle_{\texttt{ref}} |1\rangle_{\texttt{ineq}} & \text{if } r \geq z, \end{cases} \tag{82}$$

where r and z are t-bit numbers; this operation is described by COMP(`tmp`, `ref`, `ineq`) in our quantum algorithm. The

state after inequality testing is

$$|\Psi_{\text{comp}}\rangle := \frac{1}{\sqrt{\tilde{\mathcal{N}}(\sigma,m)2^t}} \sum_{j=0}^{2^{m-1}-1} g(j)\,|j\rangle_{\texttt{out}}\,|r_j\rangle_{\texttt{tmp}} \times \left(\sum_{z=0}^{r_j^{(t)}-1} |z\rangle_{\texttt{ref}}\,|0\rangle_{\texttt{ineq}} + \sum_{z=r_j^{(t)}}^{2^{t-1}-1} |z\rangle_{\texttt{ref}}\,|1\rangle_{\texttt{ineq}}\right), \tag{83}$$

where $\tilde{\mathcal{N}}(\sigma,m)$ is the normalization factor in Eq. (78), $r^{(t)} := \lfloor 2^t r_j \rfloor$ and

$$g(0) := 1, \quad g(j) := \sqrt{2}\exp\left(-\frac{2^{2\lfloor \log_2 j\rfloor}}{4\sigma^2}\right) \quad \forall\, j \in \{1,\ldots,2^m-1\}, \tag{84}$$

are the unnormalized amplitudes in Eq. (78). Next we unprepare the uniform superposition on `ref` with t Hadamard gates. Then projecting the single qubit `ineq` onto the success state $|0\rangle_{\texttt{ineq}}$ yields

$$|\Psi_{\text{out}}\rangle := \frac{1}{\sqrt{\tilde{\mathcal{N}}(\sigma,m)2^t}} \sum_j g(j) r_j^{(t)}\,|j\rangle, \tag{85}$$

with success probability

$$P_{\text{success}} := \frac{1}{\tilde{\mathcal{N}}(\sigma,m)2^{2t}} \sum_j \left(g(j)\,\lfloor 2^t r_j\rfloor\right)^2. \tag{86}$$

Figure 11 shows the success probability for a wide range of the standard deviation, and realistic size (number of qubits m) for the quantum register encoding the 1DG state.

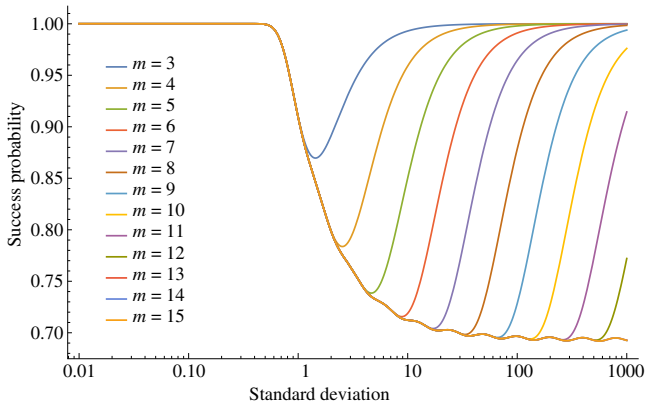

FIG. 11. The success probability for preparing a 1DG state by inequality testing for a wide range of standard deviation and number of qubits m; see Eq. (76). The success probability is at least about 70%.

The success probability is greater than $0.67 \approx 2/3$ for any realistic example. Therefore, one step of amplitude amplification is sufficient to achieve a high success probability. However, as 1DG-state preparation is at the beginning of the main algorithm for ground-state generation, it would be sufficient to prepare the state probabilistically and repeat until success. As the probability of success is at least about 70%, the repeat-until-success procedure is more efficient on average than amplitude amplification.

The state in Eq. (79) is a unary state, which can be prepared by rotations and controlled rotations [47, pp. 7–8]. We now describe how to prepare this state on an m-qubit register `out` by these operations. To this end, first we compute the rotation angles θ_ℓ in

$$\sin(\theta_\ell) = \frac{\sqrt{2}2^{\ell/2}e^{-\frac{2^{2\ell}}{4\sigma^2}}}{\sqrt{\mathcal{M}(\sigma,\ell+2)}} \quad \forall \ell \in \{0,\ldots,m-2\}, \tag{87}$$

with $\mathcal{M}(\sigma,\ell)$ given in Eq. (79), by a classical algorithm. Then we perform the following operations:

1. Perform a single-qubit rotation with angle θ_{m-2} on `out`$[m-2]$; `out`$[i]$ denotes the i^{th} qubit of `out`.
2. For ℓ from $m-3$ to zero, perform a CNOT from `out`$[\ell+1]$ to `out`$[\ell]$, and a single-qubit rotation with angle θ_ℓ on `out`$[\ell]$ if state of `out`$[\ell+1]$ is $|0\rangle$. We perform the controlled rotation by

$$\text{CROT}: |b\rangle_{\texttt{out}[\ell+1]}\,|\eta\rangle_{\texttt{ang}}\,|0\rangle_{\texttt{out}[\ell]} \mapsto |b\rangle_{\texttt{out}[\ell+1]}\,\text{ROT}^{1-b}\,|\eta\rangle_{\texttt{ang}}\,|0\rangle_{\texttt{out}[\ell]}, \tag{88}$$

   where $\text{ROT}^1 := \text{ROT}$, Eq. (73), and $\text{ROT}^0 := \mathbb{1}$. This operation is described by CROT(`out`$[\ell+1]$, `ang`, `out`$[\ell]$) in our algorithm.
3. If $j \neq 0$, perform a Hadamard on `out`$[m-1]$, i.e., the leftmost qubit of `out`. To do this, we test an inequality between the value j encoded in `out` register and the value 0 encoded in `ref` register, and write the result of inequality test to a fresh qubit labeled `flag`. Then we perform a CHAD from `flag` to `out`$[m-1]$. That is, we perform a Hadamard on `out`$[m-1]$ if state of `flag` is $|1\rangle$. Finally, we erase `flag`.
4. Convert the signed integer encoded in `out` to its two's complement representation.

The classical algorithm for computing the rotation angles is formally described by the helper function in Library 7. The purpose of this helper function is only to distinguish the classical versus quantum parts of the state-generation algorithm by inequality testing presented as pseudocode in Algorithm 7.

function ROTANGLES(m, σ) ▷ computes rotation angles θ_ℓ (87) required to prepare the state in Eq. (79)
  for $\ell \leftarrow 0$ to $m-2$ **do**
    $\mathcal{M} \leftarrow 1 + \sum_{j=1}^{\ell-1} 2^j \exp\left(-2^{2(j-1)} / \left(2\sigma^2\right)\right)$
    $\theta_\ell \leftarrow 2^{\ell/2}\sqrt{2/\mathcal{M}} \exp\left(-2^{2\ell} / \left(4\sigma^2\right)\right)$
  return θ ▷ θ is a vector with components θ_ℓ

Library 5. Helper functions for 1DG-state generation in Algorithm 7

3. Quantum fast Fourier transform

Here we present our quantum algorithm for performing a discrete Fourier transform on a quantum computer. We begin by specifying the task here and distinguishing it from the standard quantum Fourier transform [42, Chap. 5]. We then explain how a classical fast Fourier transform algorithm can be modified for execution on a quantum computer. We next explain that, for our purposes, the usual complex-valued DFT can be replaced with the DHT. Finally, we present a detailed description of our quantum algorithm for performing a DHT on a quantum computer.

For a given positive integer N and a complex-valued vector $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$, the DFT of $\mathbf{x}$ is a complex-valued vector $\tilde{\mathbf{x}} = (\tilde{x}_0, \tilde{x}_1, \ldots, \tilde{x}_{N-1})$ with components

$$\tilde{x}_\ell := \frac{1}{\sqrt{N}} \sum_{\ell'=0}^{N-1} x_{\ell'} e^{-2\pi i \ell \ell'/N}. \tag{89}$$

A quantum fast Fourier transform is then a quantum circuit QFFT designed so that

$$\text{QFFT} : |x_0\rangle |x_1\rangle \cdots |x_{N-1}\rangle \mapsto |\tilde{x}_0\rangle |\tilde{x}_1\rangle \cdots |\tilde{x}_{N-1}\rangle , \tag{90}$$

where $\tilde{x}_\ell$, Eq. (89), are components of the transformed vector $\tilde{\mathbf{x}}$. Note that QFFT is distinct from the standard quantum Fourier transform [42, p. 217], which is defined as

$$\text{QFT} : |\ell\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{\ell'=0}^{N-1} e^{-2\pi i \ell \ell'/N} \left|\ell'\right\rangle \quad \forall \ell \in \{0, 1, \ldots, N-1\}, \tag{91}$$

for any N-dimensional qudit. Specifically, in contrast to the standard quantum Fourier transform, the QFFT requires the quantum computer to perform arithmetic operations.

There are well-known techniques to execute the DFT on a classical computer using $\mathcal{O}(N \log_2 N)$ arithmetic operations, rather than the $\mathcal{O}(N^2)$ required by a naive implementation. Such techniques are called "fast Fourier transforms." Fast Fourier transforms can be straightforwardly ported to quantum algorithms by executing the prescribed arithmetic operations reversibly [21]. These arithmetic operations always take the form of an in-place transformation $(a, b) \mapsto (a + \omega b, a - \omega b)$, called "butterfly" operation, for $a, b \in \mathbb{C}$ read from the input array and $\omega \in \mathbb{C}$ some precomputed constant. As per the cost model in Sec. III A 3, we assign a unit cost for each butterfly operation, as the overall operation is roughly as computationally expensive as multiplication. Hence we can execute any fast Fourier transform on a quantum computer and report the same complexity of $\mathcal{O}(N \log_2 N)$.

The problem with the QFFT is that the resulting representation for the Gaussian state requires us to have complex-valued position states, Eq. (19). As all operations for ground-state generation require only real numbers, we avoid the complex numbers required by the DFT and instead use a discrete Hartley transform, which requires only real numbers [43, p. 100]. The DHT is valid because the ground-state ICM in the Fourier-based method (Sec. III C 2) is a real, symmetric, and circulant matrix, and hence is diagonalizable by a DHT [22, Theorem 1]. The fast Hartley transform has a similar structure to the fast Fourier transform but with a different butterfly operation [48]. The number of butterfly operations is the same. Hence, by porting these operations to quantum algorithms, we achieve the same quantum complexity of $\mathcal{O}(N \log_2 N)$ for executing a fast Hartley transform on a quantum computer.

We replace the discrete Fourier transform circuit QFFT, Eq. (90), by a new circuit QFHT defined so that

$$\text{QFHT} : |x_0\rangle |x_1\rangle \cdots |x_{N-1}\rangle \mapsto |\bar{x}_0\rangle |\bar{x}_1\rangle \cdots |\bar{x}_{N-1}\rangle ,$$
$$\bar{x}_\ell := \frac{1}{\sqrt{N}} \sum_{\ell'=0}^{N-1} x_{\ell'} \operatorname{cas}(2\pi \ell \ell'/N), \tag{92}$$

where the cas function is defined as $\operatorname{cas}(\theta) := \cos(\theta) + \sin(\theta)$ and $\bar{x}_\ell$ are components of the vector $\bar{\mathbf{x}}$, the DHT of $\mathbf{x}$. Analogous to the distinction between QFFT and the standard quantum Fourier transform, the QFFT is distinct from the standard quantum Hartley transform [49,50], which is defined as

$$\text{QHT} : |\ell\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{\ell'=0}^{N-1} \operatorname{cas}(2\pi \ell \ell'/N) \left|\ell'\right\rangle \quad \forall\, \ell \in \{0, 1, \ldots, N-1\}, \tag{93}$$

for any N-dimensional qudit.

Input:
$p \in \mathbb{Z}^+$ ▷ working precision
$m \in \mathbb{Z}^+$ ▷ position of the radix point in fixed-point number representation
$\sigma \in \mathbb{B}^p$ ▷ p-bit approximate standard deviation of the approximate 1DG pure state
$\delta \in \mathbb{B}^p$ ▷ p-bit approximate lattice spacing
Output:
out $\in \mathscr{H}_2^p$ ▷ p-qubit approximate 1DG state (33) prepared on out register
1: **function** INEQBASEDONEDG($p, m, \tilde{\sigma}, \delta$)
2: $\mathbb{R}^{m-1} \ni \theta \leftarrow$ ROTANGLES(m, σ) ▷ classically computes rotation angles for preparing the state in Eq. (79); see Library 5
3: ang $\leftarrow \theta_{m-2}/2\pi$
4: ROT(ang, out$[m-2]$) ▷ rotates $(m-2)^{\text{th}}$ qubit of out register by angle θ_{m-2}; see Eq. (73)
5: ang $\leftarrow \theta_{m-2}/2\pi$ ▷ erases ang register
6: **for** $\ell \leftarrow m-3$ to 0 **do**
7: CNOT(out$[\ell+1]$, out$[\ell]$)
8: ang $\leftarrow \theta_\ell/2\pi$
9: CROT(out$[\ell+1]$, ang, out$[\ell]$) ▷ rotates $(\ell-1)^{\text{th}}$ qubits of out if ℓ^{th} qubit of out is $|0\rangle$
10: ang $\leftarrow \theta_\ell/2\pi$ ▷ erases ang register
11: **for** $\ell \leftarrow 1$ to $m-1$ **do** ▷ lines (12–13): prepares the state in Eq. (78) on out register
12: CNOT(out$[\ell]$, out$[\ell-1]$)
13: CHAD(out$[\ell]$, out$[\ell-1]$)
14: $b \leftarrow 1$ ▷ sets the initial value for binary b
15: **while** $b = 1$ **do**
16: **for** $\ell \leftarrow 0$ to $t-1$ **do** ▷ prepares t-qubit ref register in uniform superposition state
17: H(ref$[\ell]$)
18: RATIO(out, std, tmp) ▷ computes the ratio r_j (80) into tmp as per Eq. (81)
19: COMP(tmp, ref,ineq) ▷ performs the inequality test (82) between the values encoded in tmp and ref
20: **for** $\ell \leftarrow 0$ to $t-1$ **do** ▷ erases ref register
21: H(ref$[\ell]$)
22: $b \leftarrow$ MEASURE(ineq) ▷ measures the single qubit ineq in computation basis; b is measurement's output
23: **if** $b = 1$ **then** ▷ if measurement's output is 1, the state in Eq. (77) is generated on out
24: COMP(out, ref, flag) ▷ performs the inequality test between the values encoded in out and flag
25: CHAD(flag, out$[m-1]$) ▷ controlled on flag applies a Hadamard on the leftmost qubit of out
26: COMP(out, ref, flag) ▷ erases flag
27: spc $\leftarrow \delta$ ▷ writes lattice spacing δ into spc register
28: MUL(out, spc, anc)
29: spc $\leftarrow \delta$ ▷ erases spc register
30: ERASE(out) ▷ erases out register by performing reverse of operations used in preparation
31: **for** $\ell \leftarrow 0$ to $p-1$ **do**
32: SWAP(out$[\ell]$, anc$[\ell]$) ▷ swaps the qubits of out and anc registers; erases anc
33: **yield** out

Algorithm 7. Quantum algorithm for generating a one-dimensional Gaussian state by inequality testing

To elucidate our QFHT algorithm, first we describe a recursive decomposition for the DHT. Let $\bar{\mathbf{x}}_E$ and $\bar{\mathbf{x}}_O$ be vectors comprised of the even- and odd-indexed components of $\bar{\mathbf{x}}$, Eq. (92), respectively, and let $\bar{\mathbf{x}}_L$ and $\bar{\mathbf{x}}_R$ be the left half and the right half of $\bar{\mathbf{x}}$, respectively. Then the Hartley-transformed vector $\bar{\mathbf{x}}$, Eq. (92), is written as [51, Chap. 25]

$$\bar{\mathbf{x}}_L = \bar{\mathbf{x}}_E + \mathrm{CHS}\,(\bar{\mathbf{x}}_O)\,, \quad \bar{\mathbf{x}}_R = \bar{\mathbf{x}}_E - \mathrm{CHS}\,(\bar{\mathbf{x}}_O)\,, \tag{94}$$

where CHS is the classical Hartley-shift operation with action

$$\begin{aligned} &\mathrm{CHS} : (x_0, x_1, \ldots, x_{N-1}) \mapsto (x_0^{\mathrm{s}}, x_1^{\mathrm{s}}, \ldots, x_{N-1}^{\mathrm{s}}), \\ &x_\ell^{\mathrm{s}} := x_\ell \cos(\pi\ell/N) + x_{N-\ell} \sin(\pi\ell/N). \end{aligned} \tag{95}$$

The decomposition in Eq. (94) enables a recursive algorithm for the DHT. The recursive algorithm requires a temporary workspace for writing the results of intermediate computations. We avoid the need for a temporary workspace by writing the algorithm in a nonrecursive way and performing in-place operations. Our in-place algorithm requires a quantum-data reordering, similar to the data-reordering operation known as "bit reversal" in the classical FHT algorithm. The bit reversal reorders the input data vector $\mathbf{x}$ such that the data x_i at an index i is swapped with the data $x_{\mathrm{REV}(i)}$ at index REV(i), where REV(i) is an integer obtained from i by reversing its binary digits.

Our QFHT algorithm is based on three key quantum operations that we define here. The first operation is a quantum version of the classical bit-reversal operation and

is defined as

$$\begin{aligned}\text{QREV} : &|x_0\rangle_{\texttt{vac}[0]}\,|x_1\rangle_{\texttt{vac}[1]}\cdots|x_{N-1}\rangle_{\texttt{vac}[N-1]}\\ &\mapsto \left|x_{\text{REV}(0)}\right\rangle_{\texttt{vac}[0]}\left|x_{\text{REV}(1)}\right\rangle_{\texttt{vac}[1]}\cdots\left|x_{\text{REV}(N-1)}\right\rangle_{\texttt{vac}[N-1]},\end{aligned} \tag{96}$$

where REV is the classical bit reversal. This operation swaps the values encoded in the registers $\texttt{vac}[i]$ and $\texttt{vac}[\text{REV}(i)]$ by swapping qubits of these registers; see the helper function QREV in line (1) of Library 6. The second key operation in our algorithm is a quantum Hartley shift defined as

$$\text{QHS} : |x_0\rangle\,|x_1\rangle\cdots|x_{N-1}\rangle \mapsto \left|x_0^{\text{s}}\right\rangle\left|x_1^{\text{s}}\right\rangle\cdots\left|x_{N-1}^{\text{s}}\right\rangle, \tag{97}$$

where x_ℓ^{s} is given in Eq. (95). Notice that this operation is a quantum version of the classical Hartley shift, Eq. (95). We implement QHS by $N/2$ applications of a primitive quantum Hartley-shift operation defined as

$$\begin{aligned}\text{HS} : &|c\rangle_{\texttt{cos}}\,|s\rangle_{\texttt{sin}}\,|x\rangle_{\texttt{vac}[i]}\,|y\rangle_{\texttt{vac}[j]}\\ &\mapsto |c\rangle_{\texttt{cos}}\,|s\rangle_{\texttt{sin}}\,|xc+ys\rangle_{\texttt{vac}[i]}\,|xs-yc\rangle_{\texttt{vac}[j]},\end{aligned} \tag{98}$$

where the values encoded in the quantum registers labeled cos and sin are, respectively, the cosine and sine in Eq. (95); see the helper function QHS in line (6) of Library 6. The third key operation used in the QFHT algorithm is a quantum butterfly operation defined as

$$\text{QBF} : |x\rangle_{\texttt{vac}[i]}\,|y\rangle_{\texttt{vac}[j]} \mapsto |x+y\rangle_{\texttt{vac}[i]}\,|x-y\rangle_{\texttt{vac}[j]}, \tag{99}$$

which we describe by $\text{QBF}(\texttt{vac}[i],\texttt{vac}[j])$ in our quantum algorithm. All of these operations can be implemented by quantum arithmetic, specifically the quantum multiplication MUL, Eq. (75); see Sec. IV E 4.

The QFHT algorithm proceeds as follows; see Fig. 12 for a schematic description of the algorithm. First we reorder the size-N input state vector to the algorithm by implementing the QREV, Eq. (96), operation. Then the algorithm has $\log_2 N$ stages. For each stage $s \in \{1,\ldots,\log_2 N\}$, we group the reordered state vector into $N/2^s$ blocks of size-2^s state vectors and perform the QHS, Eq. (97), on the right half of the state vector in each block. Then, for $\ell \in \{0,\ldots,2^s/2-1\}$, we perform QBF, Eq. (99), on the ℓ^{th} component of the left and right halves of the state vector in each block. Figure 12 shows a schematic representation of the QFHT algorithm for $N=8$. Components of the input state vector in this figure is represented from top to bottom. Hence the top and bottom halves of the state vector in each block represented in Fig. 12 corresponds to the left and right halves of the state vector in the block when the state vector is represented from left to right as in Eq. (97).

4. Quantum shear transform

We now present a quantum algorithm for performing a shear transform on a quantum computer. We begin by specifying the task in a shear transform and describe the quantum shear transform. Next we explain how a sequence of shear transforms with exactly one shear element can implement a general shear transform. We then discuss the shear transform required for the basis transformation in the wavelet-based algorithm. We specify the inputs and outputs of our QST algorithm and proceed with explaining the procedure. Finally, we present the QST algorithm as pseudocode.

For a shear transform, we are given a scalar $N \in \mathbb{Z}^+$, a vector $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1}) \in \mathbb{R}^N$ and a shear matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$, where the shear matrix is either a lower or an upper unit-triangular matrix. The shear transform $\tilde{\mathbf{x}} = (\tilde{x}_0, \tilde{x}_1, \ldots, \tilde{x}_{N-1})$ of $\mathbf{x}$ specified by the shear matrix $\mathbf{S}$ is defined so that $\tilde{\mathbf{x}} = \mathbf{S}\mathbf{x}$. For our application, we consider only a shear transform with an upper unit-triangular shear matrix. In this case,

$$\tilde{x}_i = x_i + \sum_{j=i+1}^{N-1} S_{ij}\,x_j, \tag{100}$$

```
1: function QREV(N, p, vac)          ▷ vac is a qubit register with N cells, each of which has p qubits; N is assumed to be an even integer
2:     for i ← 1 to N/2 − 1 do
3:         for j ← 0 to p − 1 do
4:             SWAP (vac[i, j], vac[REV(i), j])      ▷ REV(i) is an integer whose binary representations is revers of binary representation of i
5:     yield vac
6: function QHS(N, p, vac)
7:     if N > 1 then                 ▷ QHS (97) is trivial for N = 1
8:         for ℓ ← 1 to N/2 − 1 do
9:             (cos, sin) ← (cos(πℓ/N), sin(πℓ/N))      ▷ writes cosine and sine in Eq. (95) to cos and sin registers
10:            HS(cos, sin, vac[ℓ], vac[N − ℓ])         ▷ performs the primitive quantum Hartley transform (98)
11:            (cos, sin) ← (cos(πℓ/N), sin(πℓ/N))      ▷ erases cos and sin registers
12:    yield vac
```

Library 6. Helper functions for quantum fast Hartley transform in Algorithm 8

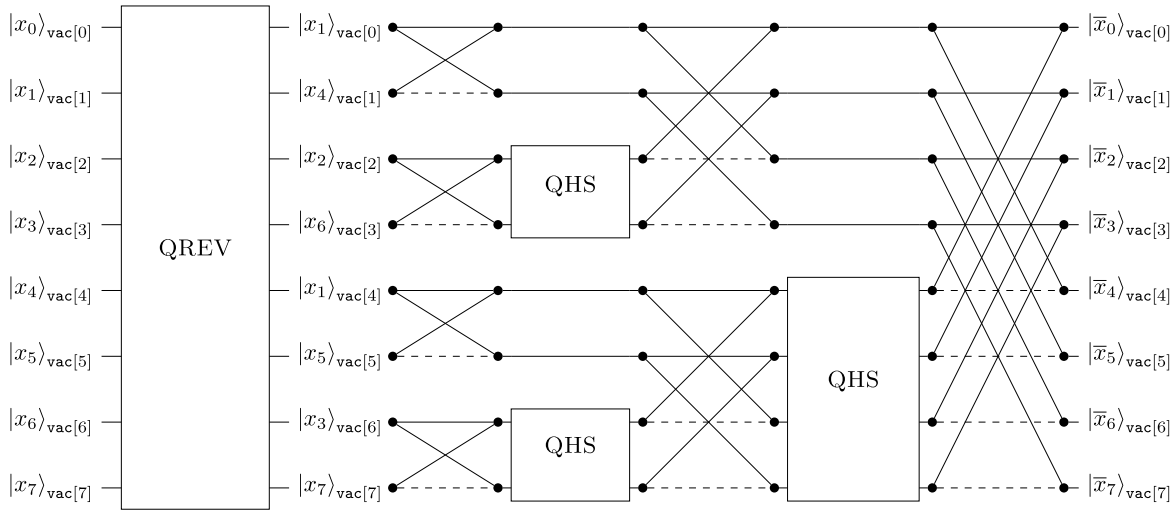

FIG. 12. Schematic description of our quantum fast Hartley-transform algorithm for $N = 8$; see Eq. (92). Here vac is a quantum register with eight cells, and each cell comprises p qubits encoding a p-bit number. QREV, Eq. (96), reorders the values encoded in the input state-vector by swapping qubits of vac[i] and vac[REV(i)], where REV(i) is an integer obtained from i by reversing its binary digits. The algorithm has $\log_2 N$ stages after QREV, each of which comprises $N/2$ QBF, Eq. (99), operations represented by crossed lines; there are $N/2^s$ blocks of QBF operations at stage $s \in \{1,\ldots,\log_2 N\}$ and each block comprises $2^s/2$ QBF. QHS performs the Hartley-shift operation on its input state vector as per Eq. (97).

is the shear transform $\tilde{\mathbf{x}}$ of $\mathbf{x}$.

The quantum shear transform is similar to the quantum fast Fourier transform. Specifically, the QST specified by $\mathbf{S}$ is a quantum circuit QST designed so that

$$\mathrm{QST} : |x_0\rangle\,|x_1\rangle\cdots|x_{N-1}\rangle \mapsto |\tilde{x}_0\rangle\,|\tilde{x}_1\rangle\cdots|\tilde{x}_{N-1}\rangle\,, \quad (101)$$

where $\tilde{x}_i$ are components of the shear-transformed vector. A shear matrix $\mathbf{S}$ is not unitary, so the map $|\mathbf{x}\rangle \mapsto |\mathbf{S}\mathbf{x}\rangle$ cannot be directly implemented by a quantum circuit. However, by storing the shear elements of $\mathbf{S}$ on an ancillary quantum register and performing quantum arithmetic, this map can be implemented by a quantum circuit.

To elucidate implementation of QST, Eq. (101), by a quantum circuit, first we describe how to decompose a dense shear matrix as a product of shear matrices with exactly one shear element. Let $\mathbf{U}$ be an N-by-N upper unit-triangular matrix and let $\mathbf{S}(i,j,s)$ be a N-by-N upper unit-triangular shear matrix with one shear element $s \in \mathbb{R}$ at entry (i,j) with $i < j$. We decompose $\mathbf{U}$ into a product of shear matrices with exactly one shear element as

$$\mathbf{U} = \prod_{i=N-2}^{0} \prod_{j=N-1}^{i+1} \mathbf{S}\left(i,j,u_{ij}\right), \quad (102)$$

where u_{ij} are elements of the matrix $\mathbf{U}$. This decomposition implies that a general shear transform can be implemented by performing a sequence of shear transforms with exactly

Input:
- $N \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ ▷ number of modes; for convenience, we assume that N is a power of 2
- $p \in \mathbb{Z}^+$ ▷ working precision
- vac $\in \bigotimes_{\ell=0}^{N-1} \mathscr{H}_2^p$ ▷ tensor product of N p-qubit 1DG states

Output:
- vac $\in \mathscr{H}_2^{N\times p}$ ▷ $(N \times p)$-qubit approximate ground state prepared on vac register

1: **function** QFHT(N, p, vac)
2: QREV(N, p, vac) ▷ performs the quantum bit-reversal operation in Eq. (96); see line (1) of Library 6
3: **for** $s \leftarrow 1$ to $\log_2 N$ **do** ▷ iterates over stages of QFHT; see FIG. 12
4: **for** $b \leftarrow 0$ to $N/2^s - 1$ **do** ▷ iterates over blocks of butterflies at state s
5: QHS($2^s/2$, p, vac$[(b+1/2)2^s : (b+1)2^s - 1]$) ▷ see the helper function in line 6 of Library 6
6: **for** $\ell \leftarrow 0$ to $2^s - 1$ **do** ▷ iterates over butterflies within block b and performs butterfly operations
7: QBF (vac$[b2^s + \ell]$, vac$[(b+1/2)2^s + \ell]$) ▷ performs the quantum butterfly operation in Eq. (99)
8: **yield** vac

Algorithm 8. Quantum fast Hartley transform

one shear element. Therefore, we consider only a shear transform with one nonzero shear element.

The required shear transform for basis transformation in the wavelet-based algorithm is the inverse transpose of the upper unit-triangular matrix $\mathbf{U}$ in the UDU decomposition for the ground-state ICM in a multiscale wavelet basis. By Eq. (102), we decompose this matrix as

$$\left(\mathbf{U}^{\mathsf{T}}\right)^{-1} = \prod_{i=N-2}^{0} \prod_{j=N-1}^{i+1} \mathbf{S}^{\mathsf{T}}\left(i,j,-u_{ij}\right), \tag{103}$$

where we use the fact that the inverse of a shear matrix with one shear element is a shear matrix with the shear element negated. To implement the shear transform specified by the shear matrix $\mathbf{S}^{\mathsf{T}}(i,j,-u_{ij})$ by a quantum circuit, we write the shear element $s = -u_{ij}$ into a p-qubit register labeled `shear` and implement

$$\begin{aligned} &|s\rangle_{\texttt{shear}}\,|x\rangle_{\texttt{vac}[i]}\,|y\rangle_{\texttt{vac}[j]} \\ &\quad\mapsto |s\rangle_{\texttt{shear}}\,|x\rangle_{\texttt{vac}[i]}\,|y+sx\rangle_{\texttt{vac}[j]}, \end{aligned} \tag{104}$$

where $\texttt{vac}[i]$ and $\texttt{vac}[j]$ are two p-qubit quantum registers that store the two p-bit numbers x and y, respectively. Notice that this map is identical to the quantum multiplication MUL in Eq. (75). Hence we implement the map by performing one MUL, Eq. (75), operation and describe it by $\text{MUL}(\texttt{shear},\texttt{vac}[i],\texttt{vac}[j])$ in our quantum algorithm.

We now discuss the inputs and output of our quantum algorithm for performing the required shear transform for ground-state generation. Inputs to the algorithm are N p-qubit quantum states and the parameters that specify the shear matrix $\mathbf{U}$ in the UDU decomposition of the approximate ICM: wavelet index $\mathcal{K}$, number of modes N, the upper bandwidth w, Eq. (54), and shear elements of $\mathbf{U}$. The algorithm's output is an $(N \times p)$-qubit quantum state that represents an approximation for the ground state, Eq. (31), of the discretized QFT in a multiscale wavelet basis.

We now describe the procedure of our quantum algorithm for performing the shear transform specified by the shear matrix in Eq. (103). To perform this shear transform, we need to perform a sequence of shear transforms with one shear element; order of the shear transforms is imposed by Eq. (103). We start by performing the rightmost shear transform in Eq. (103) and proceed to perform the leftmost shear transform. That is, we start by $i = 0$ and proceed to $i = N-1$, and for each i we perform the shear transform defined by $\mathbf{S}^{\mathsf{T}}(i,j,-u_{ij})$ for $j > i$.

The inputs, output, and explicit procedure of our quantum algorithm for performing the needed shear transform for ground-state generation is presented by the pseudocode in Algorithm 9. In this algorithm, we use the helper functions in Library 7 and our method for storing the nonzero shear elements—described in Sec. IV C 3 and illustrated in Fig. 9—to find the location of the nonzero shear element in the shear matrix and perform only the shear transform induced by these elements.

E. Complexity analysis

In this subsection, we analyze time complexity for our two ground-state-generation algorithms with respect to the primitive operations discussed in Sec. III A 3. We begin, in Sec. IV E 1, by analyzing time complexity for classical preprocessing of the Fourier-based algorithm and present time complexity for the wavelet-based algorithm's classical preprocessing in Sec. IV E 2. The time complexity for our quantum algorithm for generating a one-dimensional Gaussian state is discussed in Sec. IV E 3. We analyze time complexity for the quantum fast Fourier-transform algorithm and the quantum shear-transform algorithm in Sec. IV E 4 and Sec. IV E 5, respectively. Finally, we put all complexities together in Sec. IV E 6 and discuss the overall time complexity for the Fourier- and wavelet-based algorithms.

1. Classical preprocessing in Fourier-based algorithm

Here we analyze time complexity for classical preprocessing in the Fourier-based algorithm. Following Algorithm 1, classical preprocessing involves computing various elementary functions, such as logarithm and trigonometric functions, and has two key subroutines: computing the second-order derivative overlaps, Eq. (13), and eigenvalues of the ground state's ICM. First we discuss time complexity for computing the elementary functions and then analyze the key subroutines' time complexity. Finally, we build on these complexities and discuss the overall complexity of classical preprocessing.

The elementary functions used in various subroutines of the classical preprocessing in Algorithm 1 are logarithm, square-root, inverse-square-root, and trigonometric functions. The time complexity for computing these functions is analyzed with respect to time complexity for performing multiplication [39]. Multiplication is a primitive operation in our cost model and has a unit cost. Therefore, as per Sec. II D, the time complexity for computing square root or inverse square root of a number in our cost model is $\mathcal{O}(1)$, and the time complexity for computing logarithm or trigonometric functions to precision p are each $\mathcal{O}(\log p)$.

We now proceed with analyzing time complexity for computing the second-order derivative overlaps, Eq. (13). We compute these overlaps by Algorithm 11 using Beylkin's method [44]. The derivative overlaps in this method are elements of the unique solution vector for a system of $2\mathcal{K}$ linear algebraic equations with $2\mathcal{K}-1$ unknowns; see Appendix C. The standard algorithm for solving the system of linear equations is based on Gaussian elimination that requires $\mathcal{O}(\mathcal{K}^3)$ basic arithmetic operations. Hence the time complexity for computing the

1: **function** ssQST($i, s_0, \mathbf{s}_{ss}^{(s_0)}, p, \mathsf{vac}[0 : 2^{s_0}-1]$)
2: **for** $j \leftarrow i+1$ to $2^{s_0}-1$ **do** ▷ lines (2–5) perform the shear transform induced by the ss block of inverse-transpose of $\mathbf{U}$
3: $\mathsf{shear} \leftarrow -s^{(s_0)}_{ss;\, i2^{s_0}-i(i+1)/2+j-(i+1)}$ ▷ writes negate of the shear element at $(i,j)^{\text{th}}$ entry of $\mathbf{U}^{(s_0)}_{ss}$ into the shear register
4: MUL($\mathsf{shear}, \mathsf{vac}[i], \mathsf{vac}[j]$)
5: $\mathsf{shear} \leftarrow -s^{(s_0)}_{ss;\, i2^{s_0}-i(i+1)/2+j-(i+1)}$ ▷ erases the shear register by re-writing negate of the shear element
6: **yield** $\mathsf{vac}[0 : 2^{s_0}-1]$
7: **function** swQST($i, s_0, c, \mathbf{S}^{(s_0,c)}_{sw}, p, \mathsf{vac}[2^c : 2^{c+1}-1]$)
8: **for** $j \leftarrow 0$ to 2^c-1 **do** ▷ iterates over columns of $\mathbf{S}^{(s_0,c)}_{sw}$
9: $\mathsf{shear} \leftarrow -s^{(s_0,c)}_{sw;\, i,j}$
10: MUL $(\mathsf{shear}, \mathsf{vac}[i], \mathsf{vac}\,[2^c+j])$ ▷ note that j^{th} column of $\mathbf{U}^{(s_0,c)}_{sw}$ is $(2^k+j)^{\text{th}}$ column of $\mathbf{U}$
11: $\mathsf{shear} \leftarrow -s^{(s_0,c)}_{sw;\, i,j}$
12: **yield** $\mathsf{vac}[2^c : 2^{c+1}-1]$
13: **function** DIAGWWQST($i, r, \mathbf{S}^{(r,r)}_{ww}, p, w, \mathsf{vac}[2^r : 2^{r+1}-1]$)
14: **for** $j \leftarrow 0$ to $w-1$ **do** ▷ iterates over columns of $\mathbf{S}^{(r,r)}_{ww}$
15: $\mathsf{shear} \leftarrow -s^{(r,r)}_{ww;\, i,j}$
16: **if** $i < 2^r - w$ **then**
17: MUL $(\mathsf{shear}, \mathsf{vac}\,[2^r+i]\,, \mathsf{vac}\,[2^r+i+1+j])$ ▷ note that i^{th} row of $\mathbf{U}^{(r,r)}_{ww}$ is $(2^r+i)^{\text{th}}$ row of $\mathbf{U}$
18: **if** $i \geq 2^r - w$ and $j < 2^r - 1 - i$ **then**
19: MUL $\left(\mathsf{shear}, \mathsf{vac}\,[2^r+i]\,, \mathsf{vac}\,\left[2^{r+1}-w+j\right]\right)$
20: $\mathsf{shear} \leftarrow -s^{(r,r)}_{ww;\, i,j}$
21: **if** $i \leq w-1$ **then** ▷ perform shear transform corresponding to shear elements in TRP of $\mathbf{S}^{(r,r)}_{ww}$
22: **for** $j \leftarrow i$ to $w-1$ **do**
23: $\mathsf{shear} \leftarrow -s^{(r,r)}_{ww;\, 2^r-1-i,j}$
24: MUL $\left(\mathsf{shear}, \mathsf{vac}\,[2^r+i]\,, \mathsf{vac}\,\left[2^{r+1}-w+j\right]\right)$
25: $\mathsf{shear} \leftarrow -s^{(r,r)}_{ww;\, 2^r-1-i,j}$
26: **yield** $\mathsf{vac}[2^r : 2^{r+1}-1]$
27: **function** OFFDIAGWWQST($i, r, c, \mathbf{S}^{(r,c)}_{ww}, p, w, \mathsf{vac}[2^c : 2^{c+1}-1]$)
28: $h \leftarrow$ LASTCOLNNZ(r,c)
29: $H \leftarrow$ VERTWIDTH(r,c)
30: $W \leftarrow$ WIDTH(r,c)
31: $(i_{\text{LT}}, i_{\text{FB}}) \leftarrow (h-1, 2^r-(H-h-1))$ ▷ i_{LT} is row index of last column's last nonzero in TRP and i_{FB} is row index of first column's first nonzero in BLP of $\mathbf{U}^{(r,c)}_{ww}$
32: $j_{\text{FT}} \leftarrow$ WIDTH$(r,c) - w + i2^{c-r}$ ▷ computes column index of first nonzero in i^{th} row in TRP of $\mathbf{S}^{(r,c)}_{ww}$
33: **for** $j \leftarrow 0$ to WIDTH$(r,c)-1$ **do** ▷ iterates over columns of $\mathbf{S}^{(r,c)}_{ww}$
34: $\mathsf{shear} \leftarrow -s^{(r,c)}_{ww;\, i,j}$
35: **if** $i \leq i_{\text{LT}}$ and $j < j_{\text{FT}}$ **then** ▷ lines (35–40): perform shear transform corresponding to shear elements in MP of $\mathbf{S}^{(r,c)}_{ww}$
36: MUL($\mathsf{shear}, \mathsf{vac}\,[2^r+i]\,, \mathsf{vac}\,[2^c+j]$)
37: **if** $i_{\text{LT}} < i < i_{\text{FB}}$ **then**
38: MUL $\left(\mathsf{shear}, \mathsf{vac}\,[2^r+i]\,, \mathsf{vac}\,\left[2^c+2^{c-r}(i-i_{\text{LT}})+j\right]\right)$
39: **if** $i \geq i_{\text{FB}}$ and $j \geq 2^{c-r}(i-i_{\text{FB}})$ **then**
40: MUL $\left(\mathsf{shear}, \mathsf{vac}\,[2^r+i]\,, \mathsf{vac}\,\left[2^{c+1}-w-(2^c-i)\,2^{c-r}+j\right]\right)$
41: **if** $i \leq i_{\text{LT}}$ and $j \geq j_{\text{FT}}$ **then** ▷ perform shear transform corresponding to shear elements in TRP of $\mathbf{S}^{(r,c)}_{ww}$
42: MUL $\left(\mathsf{shear}, \mathsf{vac}\,[2^r+i]\,, \mathsf{vac}\,\left[2^{c+1}-W+j\right]\right)$
43: **if** $i \geq i_{\text{FB}}$ and $j < 2^{c-r}(i-i_{\text{FB}})$ **then** ▷ perform shear transform corresponding to shear elements in BLP of $\mathbf{S}^{(r,c)}_{ww}$
44: MUL $(\mathsf{shear}, \mathsf{vac}[2^r+i], \mathsf{vac}[2^c+j])$
45: $\mathsf{shear} \leftarrow -s^{(r,c)}_{ww;\, i,j}$
46: **yield** $\mathsf{vac}[2^c : 2^{c+1}-1]$

Library 7. Helper functions for quantum shear transform in Algorithm 9

Input:
$\mathcal{K} \in \mathbb{Z}_{\geq 3}$ ▷ wavelet index
$N \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ ▷ number of modes; for convenience, we assume N is a power of 2
$p \in \mathbb{Z}^+$ ▷ working precision
$w \in \mathbb{Z}^+$ ▷ upper bandwidth of diagonal ww blocks in approximate ICM $\tilde{\mathbf{A}}$
$\mathbf{S} := \left\{ \mathbf{s}_{ss}^{(s_0)} \in \mathbb{R}^{\frac{1}{2}2^{s_0}(2^{s_0}-1)}, \mathbf{S}_{sw}^{(s_0,c)} \in \mathbb{R}^{2^{s_0}\times 2^c}, \mathbf{S}_{ww}^{(r,c)} \in \mathbb{R}^{2^r \times (\text{WIDTH}(r,c)-(w+1)\delta_{rc})} \,\middle|\, \lceil \log_2(4\mathcal{K}-2) \rceil =: s_0 \leq r \leq c < k := \log_2 N \right\}$ ▷ shear elements in main and upper-diagonal blocks of $\mathbf{U}$ in UDU decomposition of $\tilde{\mathbf{A}}$; here $\text{WIDTH}(r,c)$ (55) is bandwidth of $\tilde{\mathbf{A}}_{ww}^{(r,c)}$
$\mathsf{vac} \in \bigotimes_{\ell=0}^{N-1} \mathscr{H}_2^p$ ▷ tensor product of N p-qubit 1DG states
Output:
$\mathsf{vac} \in \mathscr{H}_2^{N\times p}$ ▷ $(N \times p)$-qubit approximate ground state prepared on vac register
1: **function** QST($\mathcal{K}, N, p, w, \mathbf{S}$)
2: **for** $i \leftarrow 0$ to $2^{s_0} - 1$ **do** ▷ perform the shear transform induced by the ss block of inverse-transpose of $\mathbf{U}$
3: ssQST$(i, s_0, \mathbf{s}_{ss}^{(s_0)}, p, \mathsf{vac}[0 : 2^{s_0} - 1])$ ▷ see line (1) of Library 7
4: **for** $c \leftarrow s_0$ to $k-1$ **do** ▷ perform the shear transform induced by the sw blocks of inverse-transpose of $\mathbf{U}$
5: swQST$(i, s_0, c, \mathbf{S}_{sw}^{(s_0,c)}, p, \mathsf{vac}[2^c : 2^{c+1} - 1])$ ▷ see line (7) of Library 7
6: **for** $r \leftarrow s_0$ to $k-1$ **do** ▷ perform the shear transform induced by the ww blocks of inverse-transpose of $\mathbf{U}$
7: **for** $i \leftarrow 0$ to $2^r - 1$ **do** ▷ iterates over rows of $\mathbf{U}_{ww}^{(r,r)}$
8: DIAGWWQST$(i, r, \mathbf{S}_{ww}^{(r,r)}, p, w, \mathsf{vac}[2^r : 2^{r+1} - 1])$ ▷ see line (13) of Library 7
9: **for** $c \leftarrow r+1$ to $k-1$ **do**
10: OFFDIAGWWQST$(i, r, c, \mathbf{S}_{ww}^{(r,c)}, p, w, \mathsf{vac}[2^c : 2^{c+1} - 1])$ ▷ see line (27) of Library 7
11: **yield** vac

Algorithm 9. Quantum algorithm for a shear transform

derivative overlaps is cubic with respect to the wavelet index $\mathcal{K}$.

We compute eigenvalues of the ground state's ICM by Algorithm 3. By the lines (3)–(4) of this algorithm, computing each eigenvalue requires performing $\mathcal{O}(\mathcal{K})$ basic arithmetic operations and computing two elementary functions: square root and cosine. The square-root function is computed once, and cosine is computed $\mathcal{O}(\mathcal{K})$ times. By time complexity for computing these functions, computing each eigenvalue requires $\mathcal{O}(\mathcal{K}\log p)$ basic arithmetic operations. Therefore, time complexity for computing the eigenvalues is

$$T_{\text{eigens}} \in \mathcal{O}(N\mathcal{K}\log p), \tag{105}$$

because N eigenvalues are computed.

We lastly put all complexities together to achieve the overall time complexity for classical preprocessing in the Fourier-based algorithm. Following Algorithm 1, classical preprocessing requires computing the working precision p, the second-order derivative overlaps $\boldsymbol{\Delta}$, eigenvalues $\boldsymbol{\lambda}$ of the ICM, lattice spacing δ and the standard deviation $\tilde{\boldsymbol{\sigma}}$ for the discrete 1DG states. Computing p requires computing one inverse-square-root function and one logarithm function. Computing δ and $\tilde{\boldsymbol{\sigma}}$, respectively, require computing one and N inverse-square-root functions. Therefore, by the time complexities for computing logarithm and inverse-square-root functions, the time complexity for computing p, δ, and $\tilde{\boldsymbol{\sigma}}$ are $\mathcal{O}(\log p)$, $\mathcal{O}(1)$, and $\mathcal{O}(N)$, respectively. The combination of these complexities with the complexity for computing $\boldsymbol{\Delta}$ and $\boldsymbol{\lambda}$ yields $\mathcal{O}(\mathcal{K}^3 + N\mathcal{K}\log p)$. As p, Eq. (49), is logarithmic in N, the overall time complexity for classical preprocessing is quasilinear in the number of modes N.

2. Classical preprocessing in wavelet-based algorithm

We now analyze time complexity for classical preprocessing in the wavelet-based algorithm. In contrast to Algorithm 1, classical preprocessing in the wavelet-based algorithm, Algorithm 2, has two unique subroutines: computing the circulant row in unique blocks of the ground state's ICM and the UDU decomposition for the approximate ICM. First we elaborate on the time complexity for these unique subroutines and then discuss the overall time complexity for classical preprocessing in the wavelet-based algorithm.

We begin by analyzing time complexity for Algorithm 4, which computes the circulant row in unique blocks of the approximate ICM. This algorithm involves computing eigenvalues of the ground-state ICM, the low-pass filters, Eq. (1), and the circulant row of the fixed-scale ICM $\mathbf{A}_{ss}^{(c)}$, Eq. (24), at scale c for $c \in \{s_0, \ldots, k\}$. The time complexity to compute the low-pass filters for Daubechies $\mathcal{K}$ wavelets with precision p is

$$T_{\text{lowpass}} \in \mathcal{O}(\mathcal{K}\log^5 \mathcal{K}\log(3\mathcal{K}+p)), \tag{106}$$

see Appendix B. As per lines (3)–(4) of Algorithm 4, computing circulant row of $\mathbf{A}_{ss}^{(k)}$ requires performing one

division, $\Theta(N)$ multiplications, N additions, and computing N cosine functions. Computing trigonometric functions with precision p require $\mathcal{O}(\log p)$ basic arithmetic operations; see Sec. II D. Therefore, time complexity to compute circulant row of $\mathbf{A}_{ss}^{(k)}$ is $\mathcal{O}(N\log p)$. Computing circulant row of $\mathbf{A}_{ss}^{(c)}$ by circulant row of $\mathbf{A}_{ss}^{(c+1)}$ requires $\Theta(\mathcal{K}2^c)$ basic arithmetic operations. Therefore, the overall time complexity to compute the circulant rows of $\mathbf{A}_{ss}^{(c)}$ for $c \in \{s_0, \ldots, k-1\}$ is

$$T_{\text{circ}}^{ss} \in \Theta(\mathcal{K}2^k) = \Theta(\mathcal{K}N), \tag{107}$$

where we use $k = \log_2 N$. By a similar analysis, following lines (8)–(18) of Algorithm 4, we obtain

$$T_{\text{circ}}^{sw} \in \Theta(\mathcal{K}N\log_2 N), \quad T_{\text{circ}}^{ww} \in \Theta(\mathcal{K}N\log_2 N), \tag{108}$$

for time complexity to compute circulant rows in the sw and ww blocks of the ground-state ICM. The time complexity to perform the rest of computation is

$$T_{\text{highpass}} \in \mathcal{O}(\mathcal{K}), \tag{109}$$

the high-pass filters in lines (6)–(7) of Algorithm 4. Altogether, Eqs. (105) to (109) and the fact that p is logarithmic in the numbers of modes N, yield

$$\begin{aligned} T_{\text{circ}} &= T_{\text{eigens}} + T_{\text{lowpass}} + T_{\text{circ}}^{ss} + T_{\text{circ}}^{sw} + T_{\text{circ}}^{ww} + T_{\text{highpass}} \\ &\in \mathcal{O}(\mathcal{K}N\log_2 N), \end{aligned} \tag{110}$$

for the overall time complexity to compute the circulant rows.

We now proceed with analyzing time complexity for our UDU-decomposition algorithm presented in Algorithm 5. This algorithm computes diagonal elements of the diagonal matrix $\mathbf{D}$ and shear elements of the upper unit-triangular matrix $\mathbf{U}$ in the UDU decomposition of the approximate ICM $\tilde{\mathbf{A}}$. We separately analyze time complexity for computing the diagonal elements and time complexity for computing the shear elements. The sum of these complexities then yields the UDU decomposition's time complexity.

First we analyze time complexity for computing diagonal elements of $\mathbf{D}$, which, as per Eq. (61), is a block-diagonal matrix whose blocks are diagonal matrices $\mathbf{D}_{ss}^{(s_0)}$ and $\mathbf{D}_{ww}^{(c)}$ for $s_0 \le c < k$. Computing i^{th} diagonal element $d_{ww;\,i}^{(c)}$ of $\mathbf{D}_{ww}^{(c)}$ according to Eq. (63) requires computing nonzero elements in i^{th} row of $\mathbf{V}_{ww}^{(c,s)}$, Eq. (62), for each $s \in \{c, \ldots, k-1\}$, multiplying each nonzero element in i^{th} row of $\mathbf{U}_{ww}^{(c,s)}$, Eq. (60), by its corresponding element in i^{th} row of $\mathbf{V}_{ww}^{(c,s)}$ and adding them all. The number of nonzero elements in each row of $\mathbf{V}_{ww}^{(c,s)}$ is at most $\text{WIDTH}(c,s)$, where $\text{WIDTH}(c,s)$, Eq. (55), is bandwidth of the block $\tilde{\mathbf{A}}_{ww}^{(c,s)}$ and, by Eq. (62), computing each nonzero element of $\mathbf{V}_{ww}^{(c,s)}$ requires one multiplication. Therefore, computing each of the 2^c diagonal elements of $\mathbf{D}_{ww}^{(c)}$ requires at most $3\sum_s \text{WIDTH}(c,s)$ basic arithmetic operations. Consequently, time complexity to compute all diagonal elements in the ww blocks of $\mathbf{D}$ is

$$T_{\text{diags}}^{ww} = \sum_{c=s_0}^{k-1} 2^c \times \left(3\sum_{s=c}^{k-1} \text{WIDTH}(c,s)\right) \in \mathcal{O}(\mathcal{K}N\log_2 N), \tag{111}$$

where we use $k = \log_2 N$.

By a similar analysis, we obtain time complexity to compute diagonal elements in the ss block of $\mathbf{D}$. To compute i^{th} diagonal element in this block, we compute all elements in the i^{th} row of the ss and sw blocks of $\mathbf{V}$ because these blocks are dense matrices. Being dense blocks, the sum of nonzero elements in any row of the ss and sw blocks is at most N, the number of columns of $\mathbf{V}$. Therefore, computing each diagonal element requires at most $3N$ basic arithmetic operations, so we have $T_{\text{diags}}^{ss} = 2^{s_0} \times (3N)$ for time complexity to compute diagonal elements in the ss block of $\mathbf{D}$. This equation together with Eq. (111) yields

$$T_{\text{diags}} = T_{\text{diags}}^{ss} + T_{\text{diags}}^{ww} \in \mathcal{O}(\mathcal{K}N\log_2 N), \tag{112}$$

for the time complexity to compute all diagonal elements of $\mathbf{D}$.

We now analyze time complexity for computing shear elements of $\mathbf{U}$ in the UDU decomposition. As per Eq. (60), $\mathbf{U}$ is a block matrix with three types of blocks: ss, sw, and ww blocks. By Eq. (66), computing the shear element at entry (m,i) of the sw block $\mathbf{U}_{sw}^{(s_0,c)}$ requires multiplying each nonzero element in m^{th} row of $\mathbf{U}_{sw}^{(s_0,s)}$ by its corresponding element in m^{th} row of $\mathbf{V}_{ww}^{(c,s)}$, Eq. (62), for $s \in \{c, \ldots, k-1\}$ and adding them all. The NNZ elements in each row of $\mathbf{V}_{ww}^{(c,s)}$ is at most $\text{WIDTH}(c,s)$, Eq. (55), hence computing each shear element of $\mathbf{U}_{sw}^{(s_0,c)}$ requires at most $2\sum_s \text{WIDTH}(c,s)$ basic arithmetic operations. Being a dense matrix, $\mathbf{U}_{sw}^{(s_0,c)}$ has $2^{s_0} \times 2^c$ shear elements and, therefore, we have

$$\begin{aligned} T_{\text{shear}}^{sw} &= \sum_{c=k-1}^{s_0} \left[\left(2^{s_0} \times 2^c\right) \times \left(2\sum_{s=c}^{k-1} \text{WIDTH}(c,s)\right)\right] \\ &\in \mathcal{O}(\mathcal{K}N\log_2 N), \end{aligned} \tag{113}$$

for time complexity to compute all shear elements in the sw blocks of $\mathbf{U}$. Likewise, by Eq. (67), computing each shear element of the ww block $\mathbf{U}_{ww}^{(r,c)}$ requires at most $2\sum_s \text{WIDTH}(c,s)$ basic arithmetic operations. This block is a sparse matrix with at most $2^r \times \text{WIDTH}(r,c)$ shear

elements. We therefore achieve

$$T^{ww}_{\text{shear}} = \sum_{c=k-1}^{s_0} \left[\left(\sum_{r=s_0}^{c} 2^r \times \text{WIDTH}(r,c) \right) \times \left(2 \sum_{s=c}^{k-1} \text{WIDTH}(c,s) \right) \right] \in \mathcal{O}(\mathcal{K} N \log_2 N), \tag{114}$$

for time complexity to compute all shear elements in the *ww* blocks of $\mathbf{U}$.

A similar analysis yields the time complexity T^{ss}_{shear} to computing all shear elements in the *ss* bock of $\mathbf{U}$. By Eq. (65), computing the shear element at entry (m,i) of the *ss* block requires multiplying each nonzero element in m^{th} row of this block by its corresponding element in m^{th} row of the *ss* block of $\mathbf{V}$, multiplying each nonzero element in m^{th} row of $\mathbf{U}^{(s_0,s)}_{sw}$ by its corresponding element in m^{th} row of $\mathbf{V}^{(s_0,s)}_{sw}$ for every $s \in \{s_0, \ldots, k-1\}$ and adding them all. As the *ss* and *sw* blocks of $\mathbf{U}$ are dense matrices, computing each shear element in the ss block requires at most $2N$ basic arithmetic operations and because the *ss* block has $2^{s_0} \times (2^{s_0}-1)/2 \in \Theta(\mathcal{K}^2)$ shear elements, we have $T^{ss}_{\text{shear}} \in \mathcal{O}(\mathcal{K}^2 N)$. The sum of this time complexity by the time complexities in Eq. (113) and Eq. (114) yields

$$T_{\text{shear}} = T^{ss}_{\text{shear}} + T^{sw}_{\text{shear}} + T^{ww}_{\text{shear}} \in \mathcal{O}(\mathcal{K}^2 N \log_2 N), \tag{115}$$

for time complexity to computing all shear elements of $\mathbf{U}$. Finally, by the time complexities for computing the diagonal, Eq. (112), and shear elements, Eq. (115), we obtain

$$T_{\text{UDU}} = T_{\text{diags}} + T_{\text{shear}} \in \mathcal{O}(\mathcal{K}^2 N \log_2 N), \tag{116}$$

for the UDU decomposition's time complexity.

We finally discuss the overall time complexity for classical preprocessing in the wavelet-based algorithm. Classical preprocessing in Algorithms 1 and 2 have four identical subroutines: computing (1) working precision p; (2) second-order derivative overlaps $\boldsymbol{\Delta}$; (3) lattice spacing δ; and (4) the vector $\tilde{\boldsymbol{\sigma}}$ of standard deviations for the approximate 1DG states. As per the discussion in Sec. IV E 1, the overall time complexity for these subroutines is $\mathcal{O}(N)$. By the combination of this time complexity with the time complexities for the two unique subroutines, T_{circ}, Eq. (110), and T_{UDU}, Eq. (116), we conclude that the time complexity for classical preprocessing in the wavelet-based algorithm is quasilinear in the number of modes N.

3. One-dimensional Gaussian-state generation

Here we analyze time complexity for the quantum algorithm presented in Algorithm 6 that generates an approximation for a 1DG state on a quantum computer. We begin with analyzing time complexity for executing the quantum circuit in Fig. 10, which represents the iterative part of Algorithm 6. Then we discuss the algorithm's overall time complexity.

The quantum circuit in Fig. 10 involves performing two ANGLE, Eq. (72), operations, one ROT, Eq. (73), two SHIFT, Eq. (74), one Pauli-X, and one CNOT operation. The circuit's time complexity T_{iter} is therefore

$$T_{\text{iter}} = 2T_{\text{ANGLE}} + T_{\text{ROT}} + 2T_{\text{SHIFT}} + 2, \tag{117}$$

where $T_{\text{ANGLE}}, T_{\text{ROT}}$, and T_{SHIFT} are time complexities to implement ANGLE, ROT, and SHIFT operations, respectively. The ROT and SHIFT operations operate on quantum registers with p qubits, where p is logarithmic in the number of modes. As illustrated in Fig. 13, implementing ROT, Eq. (73), requires performing at most p standard rotations and SHIFT, Eq. (74), is implemented by performing p SWAP gates. Therefore, $T_{\text{ROT}} \in \mathcal{O}(p)$ and $T_{\text{SHIFT}} = p$. Our analysis in Appendix E shows that $T_{\text{ANGLE}} \in \mathcal{O}(p^2)$. The combination of these complexities yields

$$T_{\text{iter}} \in \mathcal{O}(p^2), \tag{118}$$

for time complexity of the quantum circuit in Fig. 10, which implements the iterative part of Algorithm 6.

The quantum algorithm in Algorithm 6 performs the operations of the quantum circuit in Fig. 10 m times. The algorithm also involves performing m SWAP operations, lines (6)–(7) of the algorithm, and one MUL, Eq. (75), operation in line (8). All of these operations are also used in the uncomputation part of Algorithm 6. Therefore, by Eq. (118) and $m < p$, the overall time complexity T_{1DG} for generating a 1DG state by Algorithm 6 is

$$T_{\text{1DG}} = (2m) \times T_{\text{iter}} + 2 \times (m+1) \in \mathcal{O}(p^3). \tag{119}$$

As p, Eq. (49), is logarithmic in the number of modes N, the overall complexity for generating a 1DG state is logarithmic in N.

4. Quantum fast Hartley transform

We now analyze time complexity for our quantum fast Hartley-transform algorithm presented in Algorithm 8. Our algorithm is based on three key quantum operations: QREV, Eq. (96), QBF, Eq. (99), and HS, Eq. (98); see Sec. IV D 3. First we analyze time complexity for performing these key operations with respect to the primitive operations in our cost model (Sec. III A 3). Then we build on these complexities and discuss the overall time complexity for the QFHT algorithm.

We begin with analyzing time complexity to perform the quantum data-reordering QREV, Eq. (96). This operation reorders the values encoded in the size-N input state

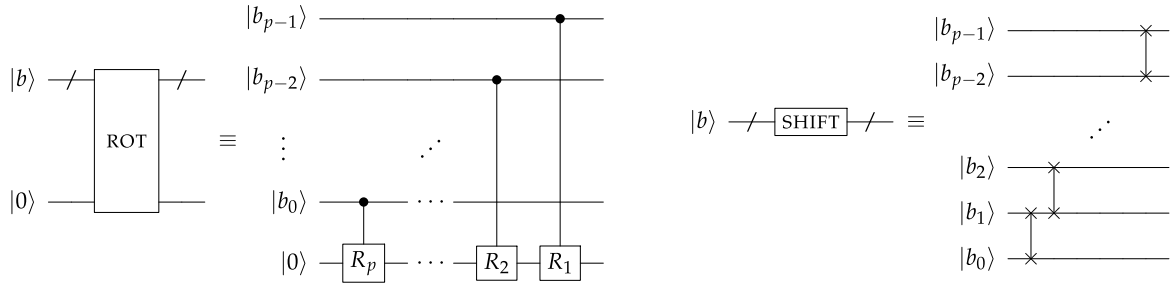

FIG. 13. Implementation of ROT, Eq. (73), and SHIFT, Eq. (74), by primitive operations. Left: implementing ROT using bits of a p-bit number b requires at most p standard rotations $R_\ell := \exp(-2\pi \mathrm{i}/2^\ell)$. Right: implementing SHIFT requires p SWAP gates.

vector vac by swapping qubits of $\mathtt{vac}[i]$ and $\mathtt{vac}[\mathrm{REV}(i)]$ for each $i \in \{1,\ldots,N/2-1\}$, where $\mathrm{REV}(i)$ is an integer obtained form i by reversing its binary digits. Hence, we have $T_{\mathrm{QREV}} \in \mathcal{O}(N)$ for time complexity to perform QREV, Eq. (96). Next we analyze time complexity T_{QBF} for performing the quantum butterfly QBF, Eq. (99). This operation can be implemented as

$$\begin{aligned}|x\rangle_{\mathtt{vac}[i]}\,|y\rangle_{\mathtt{vac}[j]} &\mapsto |x+y\rangle_{\mathtt{vac}[i]}\,|y\rangle_{\mathtt{vac}[j]}\\ &\mapsto |x+y\rangle_{\mathtt{vac}[i]}\,|2y\rangle_{\mathtt{vac}[j]}\\ &\mapsto |x+y\rangle_{\mathtt{vac}[i]}\,|(x+y)-2y\rangle_{\mathtt{vac}[j]},\end{aligned} \tag{120}$$

on two quantum registers $\mathtt{vac}[i]$ and $\mathtt{vac}[j]$ by two additions and one multiplication. The quantum addition and multiplication here are in-place operations, and no uncomputation is required. Therefore, $T_{\mathrm{QBF}} \in \mathcal{O}(1)$ with respect to the quantum primitive operations. We use the LDU decomposition

$$\begin{bmatrix} c & s \\ s & -c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ s/c & 1 \end{bmatrix} \begin{bmatrix} c & 0 \\ 0 & -1/c \end{bmatrix} \begin{bmatrix} 1 & s/c \\ 0 & 1 \end{bmatrix}, \tag{121}$$

to implement the primitive Hartley shift HS, Eq. (98), by in-place quantum arithmetic operations; note here $s^2 + c^2 = 1$. Assuming that the values of c and s are stored into two ancillary quantum registers, HS, Eq. (98), can be implemented by in-place operations as

$$\begin{aligned}|x\rangle_{\mathtt{vac}[i]}\,|y\rangle_{\mathtt{vac}[j]} &\mapsto |x+ys/c\rangle_{\mathtt{vac}[i]}\,|y\rangle_{\mathtt{vac}[j]}\\ &\mapsto |cx+sy\rangle_{\mathtt{vac}[i]}\,|-y/c\rangle_{\mathtt{vac}[j]}\\ &\mapsto |cx+sy\rangle_{\mathtt{vac}[i]}\,|xs-yc\rangle_{\mathtt{vac}[j]},\end{aligned} \tag{122}$$

on two quantum registers $\mathtt{vac}[i]$ and $\mathtt{vac}[j]$ by performing six basic arithmetic operations. We therefore have $T_{\mathrm{HS}} \in \mathcal{O}(1)$ for time complexity to perform HS.

We now discuss the overall time complexity for the QFHT algorithm. This algorithms starts with reordering the size-N input state vector and proceeds with $\log_2(N)$ stages. Each stage requires performing $N/2$ quantum butterfly operations and $N/4$ primitive quantum Hartley-shift operations; see Fig. 12 and implementation of QHS, Eq. (97), by HS, Eq. (98), in Library 6. Thus, by the time complexities for the key quantum operations, we have

$$\begin{aligned}T_{\mathrm{QFHT}} &= T_{\mathrm{QREV}} + \big((N/2)T_{\mathrm{QBF}} + (N/4)T_{\mathrm{HS}}\big)\log_2 N\\ &\in \mathcal{O}(N\log_2 N),\end{aligned} \tag{123}$$

for time complexity to perform the quantum fast Hartley transform, Eq. (92).

5. Quantum shear transform

Here we analyze time complexity for performing a shear transform on a quantum computer. First we discuss time complexity to perform shear transform with a dense shear matrix. Then we analyze time complexity for our QST algorithm presented in Algorithm 9, which performs the shear transform required for generating the free-field ground state on a quantum register.

We use the decomposition in Eq. (102) to analyze time complexity for performing a QST with a dense shear matrix. As per this decomposition, performing a general shear transform is accomplished by performing a sequence of shear transforms with exactly one shear element. The number of terms in the sequence is equal to the NNZ shear elements in the shear matrix, and the sequence's order is specified by Eq. (102).

To perform the shear transform specified by each term of the sequence, we write the shear element into an ancillary quantum register denoted shear and perform a multiplication as per Eq. (104). Then we erase the shear register by rewriting the shear element. Therefore, performing a shear transform with exactly one shear element requires three primitive operations: writing into a quantum register, performing one multiplication, and erasing the quantum register. Consequently, the time complexity for performing a shear transform scales linearly with the NNZ shear elements in the shear-transform matrix. The NNZ shear elements for a dense N-by-N shear matrix is $N(N-1)/2$,

so we have $\Theta(N^2)$ for time complexity to perform a shear transform with a dense shear matrix.

We now specify the time complexity to perform the required quantum shear transform for ground-state generation. The shear matrix for the required shear transform is sparse. We use the identified relationship between the time complexity to perform a shear transform and the NNZ shear elements to obtain the time complexity for performing a shear transform with a sparse shear matrix. In particular, the complexity for performing a sparse shear transform is achieved by counting the NNZ shear elements in the shear-transform matrix. The NNZ shear elements for the sparse shear matrix $\mathbf{U}$ in the UDU decomposition of the approximate ICM scales as $\mathcal{O}(Nw)$, where w, Eq. (54), is logarithmic in the number of modes N. Therefore, time complexity for Algorithm 9, which performs the QST with the sparse shear matrix $\mathbf{U}$, is quasilinear in the number of modes.

6. Overall complexity for Fourier- and wavelet-based algorithms

We now determine the overall time complexity for the Fourier- and wavelet-based algorithms for ground-state generation. Both algorithms have a classical preprocessing and quantum routine. We begin by discussing the overall time complexity for each algorithm's quantum routine. We then put together the overall time complexities for classical preprocessing and quantum routine to establish the overall time complexity for each of the two algorithms.

The quantum routine for each of the two state-generation algorithms comprises two subroutines: generating N different 1DG states and performing a basis transformation to transform the 1DG states to the free-field ground state. The basis transformation in the Fourier-based algorithm is performed by executing a quantum Hartley transform, and the basis transformation in the wavelet-based algorithm is performed by executing a quantum shear transform.

The time complexity for generating each 1DG state is logarithmic in the number modes N as per Sec. IV E 3, so time complexity for the first subroutine in each algorithm's quantum routine is quasilinear in N. The time complexities for performing the quantum Hartley transform and the quantum shear transform are also quasilinear in N, as discussed in Sec. IV D 3 and Sec. IV E 5, respectively. Specifically, the time complexity for quantum routine of the Fourier-based algorithm (FBA) and the wavelet-based algorithm (WBA) is

$$\begin{aligned} T_{\mathrm{FBA}}^{(\mathrm{quantum})} &\in \mathcal{O}(N\log_2 N + N\log_2^3(N/\sqrt{m_0\varepsilon_{\mathrm{vac}}})),\\ T_{\mathrm{WBA}}^{(\mathrm{quantum})} &\in \mathcal{O}(Nw + N\log_2^3(N/\sqrt{m_0\varepsilon_{\mathrm{vac}}})), \end{aligned} \tag{124}$$

respectively, where w is given in Eq. (54). Therefore, the overall time complexity for the quantum routine of each ground-state-generation algorithm is quasilinear in the number of modes N.

The sum of overall time complexities for each algorithm's classical preprocessing and quantum routine yields the algorithm's overall time complexity. By the complexity analysis in Sec. IV E 1 and Sec. IV E 2, the overall time complexity for classical preprocessing of the Fourier- and wavelet-based algorithms is

$$\begin{aligned} T_{\mathrm{FBA}}^{(\mathrm{classical})} &\in \mathcal{O}(\mathcal{K}^3 + \mathcal{K}N\log_2\log_2(N/\sqrt{m_0\varepsilon_{\mathrm{vac}}})),\\ T_{\mathrm{WBA}}^{(\mathrm{classical})} &\in \mathcal{O}(\mathcal{K}^3 + \mathcal{K}^2 N\log_2 N), \end{aligned} \tag{125}$$

respectively. By combining overall time complexities for each algorithm's classical preprocessing and quantum routine, we conclude that the overall time complexity for each of the two state-generation algorithms is quasilinear in the number of modes.

F. Lower bound for ground-state generation

In this subsection, we discuss a lower bound on the gate complexity for generating the free-field ground state with respect to the number of modes in the discretized QFT. We show that one cannot generate a good approximation for the ground state in a sublinear time. In particular, we argue that any sublinear algorithm results in an exponentially bad approximation for the free-field ground state regarding the number of modes.

The free-field ground state for an infinite-mass ($m_0 \to \infty$) theory is the tensor product of the all-zero state for every mode. The infinite-mass theory, however, is not a physical theory. For a physical field theory with a finite but very large mass, the free-field ground state is a tensor product of one-dimensional Gaussian states with small variances; the variance of each Gaussian is $\sigma^2 = 1/m_0$, which is small due to the large mass m_0. Now suppose a sublinear algorithm exists that generates an approximation for the ground state of a large-mass theory; i.e., let us assume that the algorithm's gate complexity scales as N^α for $0 < \alpha < 1$. In this case, the algorithm must leave the state of $N^{1-\alpha}$ modes untouched in order to have sublinear gate complexity, meaning that the Gaussian states for the untouched modes are approximated by the all-zero state. The fidelity between the qubit representation of a one-dimensional Gaussian state with variance σ^2, i.e., the discrete 1DG, Eq. (33), state over a lattice with spacing δ and 2^m points, and the all-zero state is

$$\begin{aligned} F_{\mathrm{1DG}} &:= \langle 0\cdots 0|G_{\mathrm{lattice}}(\tilde{\sigma},\delta,m)\rangle\\ &= \frac{\delta}{\tilde{\mathcal{N}}} = \left(\sum_j e^{-\frac{j^2}{2\tilde{\sigma}^2}}\right)^{-1/2}, \end{aligned} \tag{126}$$

where $\tilde{\sigma} = \sigma/\delta$ and $\tilde{\mathcal{N}}$ is the normalization in Eq. (33). For fixed δ, if $\sigma^2 = 1/m_0 \geq \delta$, then $\tilde{\sigma}^2 = \sigma^2/\delta^2 \geq 1/\sigma^2 =$

m_0. We therefore have

$$F_{\mathrm{1DG}} \le \left(\sum_j e^{-j^2/2m_0}\right)^{-1/2} \le \left(1+2e^{-1/(2m_0)}\right)^{-1/2} \le e^{-1/(4m_0)}, \tag{127}$$

where the last inequality follows from $1+2\exp(-x/2) \ge \exp(x/2)$ for any $x \in (0,1)$. In this case, the fidelity between the ground state and the generated state falls off exponentially with respect to the number of modes that are untouched. Specifically, the fidelity between the ground state $|G\rangle$ and the generated state $\left|\tilde{G}\right\rangle$ is $\left\langle G\middle|\tilde{G}\right\rangle = F_{\mathrm{1DG}}^{N^{1-\alpha}} \le \exp\left[-N^{1-\alpha}/(4m_0)\right]$, which falls off exponentially with respect to N.

For all other finite-mass field theories, including the zero-mass ($m_0 \to 0$) theory, generating the free-field ground state requires a nontrivial operation on every mode because the ground state, in this case, is a superposition of computational states with Gaussian amplitudes. Therefore, generating a good approximation for the free-field ground state requires a number of gates that scales at least linearly with respect to the number of modes; any sublinear algorithm results in an exponentially bad approximation for the ground state.

G. Fourier versus wavelet approach

In this subsection, we compare the Fourier and wavelet approaches for ground-state generation. We consider two cases where the wavelet approach could be advantageous over the Fourier approach: (1) QFTs with broken translational invariance and (2) generating states beyond the free-field ground state. In Sec. IV G 1, we consider a simple case of inhomogeneous-mass QFT and explain why the wavelet approach could be advantageous. We then follow in Sec. IV G 2 by comparing particle-state generation in both Fourier and wavelet approaches.

1. Inhomogeneous-mass QFT

Here we perform a numerical study to justify why the wavelet-based approach could be preferred over the Fourier-based approach for QFTs with broken translational invariance. We consider a simple case where the free-field Hamiltonian (4) has a position-dependent mass. In particular, we consider a point defect in the QFT where the mass is m_0 plus a δ-function term. The inhomogeneous mass can be seen as a space-dependent source term for a massive scalar QFT in one spatial dimension, a case for which estimating the vacuum-to-vacuum transition amplitude was shown to be a BQP-complete problem [52].

We model the point defect by adding to the fixed-scale coupling matrix, Eq. (23), a matrix containing all zeros except for one nonzero diagonal element that is significantly larger than the free QFT mass m_0. In this case, the Fourier-based approach is not appropriate because a discrete Fourier transform cannot diagonalize the modified coupling matrix. However, our numerical experiment demonstrates that the wavelet approach accommodates such a modification.

In our numerical experiment, we take the nonzero diagonal element to be $100 \times m_0$ and compute the ground-state ICM, Eq. (32), for a wide range of m_0. Figure 14 (left) shows a visualization of the approximate ICM, Eq. (50), derived from the modified coupling matrix and Fig. 14 (right) shows the bandwidth of the approximate ICM's diagonal blocks with and without the point defect for a wide range of m_0. As per these figures, the fingerlike sparse structure of the ground-state ICM in a multiscale wavelet basis is not affected by the point defect. Consequently, the wavelet-based algorithm is not affected by the point defect and successfully yields an approximation for the free-field ground state with a quasilinear gate complexity.

2. Particle-state generation for the free QFT

Here we describe a procedure used by the main server (Sec. III A 2) for generating a free QFT particle state in the Fourier and wavelet approaches. First we explain how to prepare a free wavepacket, i.e. a spatially localized free particle, in the Fourier approach. Then we describe wavepacket preparation in the wavelet approach. Finally, we compare time complexity for particle-state generation in the two approaches.

We begin with particle-state generation in the Fourier approach. For simplicity, we consider preparing a free particle whose position-space wavefunction is the scaling function $s_\ell^{(k)}(x)$ at scale k. Specifically, we aim to prepare the wavepacket state

$$\left|s_\ell^{(k)}\right\rangle := \hat{a}_{s;\ell}^{(k)\dagger}\left|G_{\mathrm{scale}}^{(k)}\right\rangle, \tag{128}$$

where $\left|G_{\mathrm{scale}}^{(k)}\right\rangle$, Eq. (24), is the free-field ground state in the Fourier approach and $\hat{a}_{s;\ell}^{(k)\dagger}$ is the creation operator constructed from the scale-field operator $\hat{\Phi}_{s;\ell}^{(k)}$, Eq. (9), and its conjugate momentum [41]. Following the Jordan-Lee-Preskill approach for preparing a free-particle state [2, p. 1027], we introduce one ancillary qubit denoted `anc`, and construct the Hamiltonian

$$\hat{H}_{s;\ell}^{(k)} := \hat{a}_{s;\ell}^{(k)\dagger} \otimes (|1\rangle_{\mathtt{anc}}\langle 0|) + \hat{a}_{s;\ell}^{(k)} \otimes (|0\rangle_{\mathtt{anc}}\langle 1|). \tag{129}$$

The time evolution generated by this Hamiltonian for time $t = \pi/2$ is

$$e^{-\mathrm{i}\hat{H}_{s;\ell}^{(k)}\pi/2}\left|G_{\mathrm{scale}}^{(k)}\right\rangle|0\rangle_{\mathtt{anc}} = -\mathrm{i}\left|s_\ell^{(k)}\right\rangle|1\rangle_{\mathtt{anc}}, \tag{130}$$

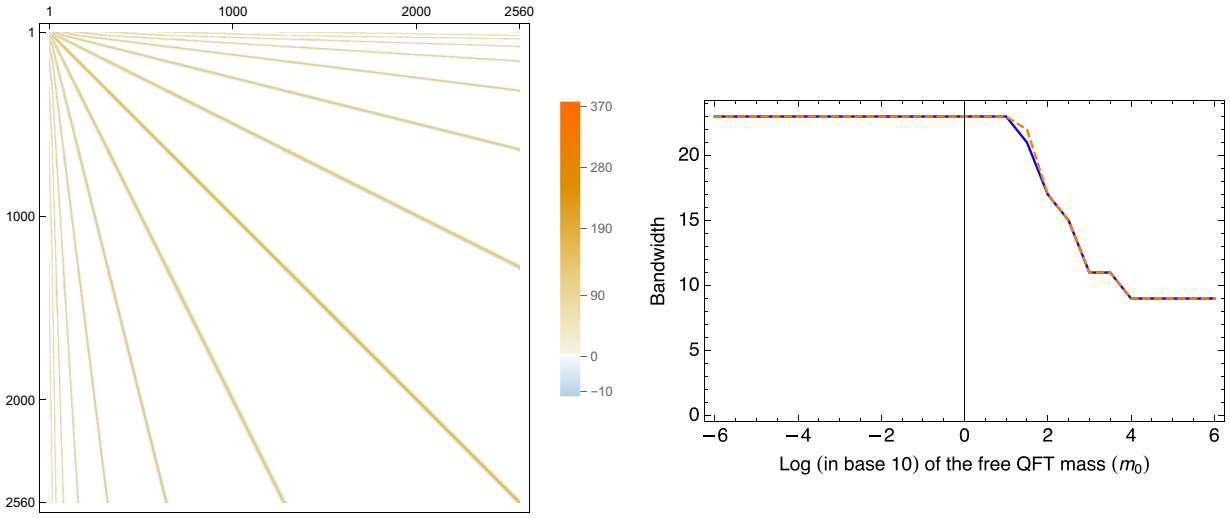

FIG. 14. Effect of a mass defect on the ground-state ICM, Eq. (32), represented in a multiscale wavelet basis. Left: visual representation for approximation of the modified ICM, Eq. (50), in a multiscale wavelet basis where elements with a magnitude less than 10^{-8} are replaced with exactly zero; rows and columns of the matrix are ordered as the matrix in Fig. 1. Right: bandwidth for diagonal blocks of the approximate ICM with mass defect (orange dashed line) and without mass defect (blue line) for a wide range of mass: $m_0 \in [10^{-6}, 10^{6}]$.

and we obtain the desired wavepacket state, Eq. (128), up to a global phase with no entanglement between the wavepacket and ancilla-qubit states. By expressing the constructed Hamiltonian (129) in terms of the scale-field operator $\hat{\Phi}^{(k)}_{s;\ell}$, Eq. (9), and its conjugate momentum, we obtain a local Hamiltonian that can be simulated by a technique described in Sec. 2.2.4 of Ref. [2] with gate complexity that is logarithmic in the numbers of modes.

For particle-state generation in the wavelet approach, we consider a free particle whose position-space wavefunction is the wavelet function $w^{(r)}_{\ell}(x)$ at scale r for some integer $r < k$. In this case, the wavepacket state we wish to prepare is

$$\left|w^{(r)}_{\ell}\right\rangle := \hat{a}^{(r)\dagger}_{s;\ell} \left|G^{(k)}_{\text{wavelet}}\right\rangle, \tag{131}$$

where $\left|G^{(k)}_{\text{wavelet}}\right\rangle$, Eq. (31), is the free-field ground state in the wavelet approach and $\hat{a}^{(r)\dagger}_{w;\,\ell}$ is the creation operator constructed from the wavelet-field operator $\hat{\Phi}^{(r)}_{w;\,\ell}$, Eq. (9), and its conjugate momentum at scale r. Analogous to particle creation in the Fourier approach, we construct the Hamiltonian

$$\hat{H}^{(r)}_{w;\ell} := \hat{a}^{(r)\dagger}_{w;\ell} \otimes (|1\rangle_{\text{anc}}\langle 0|) + \hat{a}^{(r)}_{w;\ell} \otimes (|0\rangle_{\text{anc}}\langle 1|), \tag{132}$$

and simulate time evolution according to this Hamiltonian for time $t = \pi/2$. By the time evolution, we obtain the wavepacket state in Eq. (131), up to a global phase, and an ancilla-qubit state that can be discarded.

We now compare time complexity for generating a single-particle state in the Fourier and wavelet approaches. For comparison, we assign a unit cost to simulating time evolution for a constant time induced by a local Hamiltonian in these two approaches. Specifically, we assign a unit cost to simulating time evolution induced by $\hat{H}^{(k)}_{s;\ell}$, Eq. (129), in the Fourier approach and time evolution induced by $\hat{H}^{(r)}_{w;\ell}$, Eq. (132), in the wavelet approach for a constant time.

Without loss of generality, we discuss time complexity for preparing the wavepacket state in Eq. (131) for both approaches. As described, preparing this state in the wavelet approach requires simulating time evolution generated by $\hat{H}^{(r)}_{w;\ell}$, Eq. (132), for constant time $t = \pi/2$. To prepare the same wavepacket state in the Fourier approach, we express time evolution generated by $\hat{H}^{(r)}_{w;\ell}$, Eq. (132), in terms of time evolution generated by $\hat{H}^{(k)}_{s;\ell'}$, Eq. (129), for various ℓ'. To this end, first we write the creation operation $\hat{a}^{(r)\dagger}_{w;\,\ell}$ as a linear combination of the creation operators $\hat{a}^{(k)\dagger}_{s;\,\ell}$. Let $d := k - s_0$, where $s_0 \leq k$ is the scale index for the lowest scale, and let

$$\begin{aligned}
\hat{\mathbf{a}}^{(k)\dagger}_{s} &:= \left(\hat{a}^{(k)\dagger}_{s;0}, \ldots, \hat{a}^{(k)\dagger}_{s;2^k-1}\right)^{\mathsf{T}}, \\
\hat{\mathbf{a}}^{(k)\dagger} &:= \Big(\hat{a}^{(s_0)\dagger}_{s;0}, \ldots, \hat{a}^{(s_0)\dagger}_{s;2^{s_0}-1}, \hat{a}^{(s_0)\dagger}_{w;0}, \ldots, \hat{a}^{(s_0)\dagger}_{w;2^{s_0}-1}, \\
&\qquad \hat{a}^{(s_0+1)\dagger}_{w;0}, \ldots, \hat{a}^{(k-1)\dagger}_{w;2^k-1}, \Big)^{\mathsf{T}},
\end{aligned} \tag{133}$$

be the vector of creation operators in the fixed- and multiscale wavelet bases, respectively. Then

$$\hat{\mathbf{a}}^{(k)\dagger} = \mathbf{W}^{(k)}_{d}\, \hat{\mathbf{a}}^{(k)\dagger}_{s}, \tag{134}$$

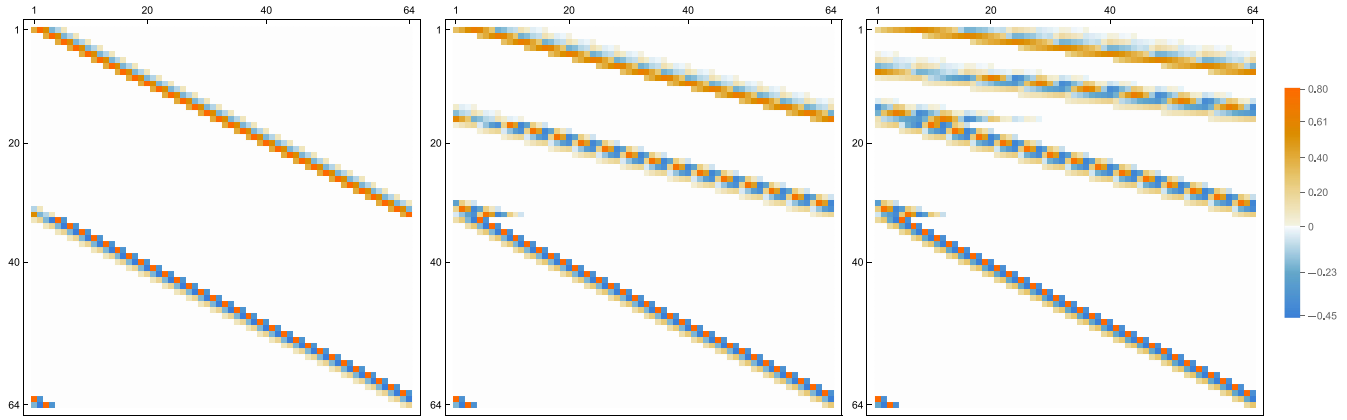

FIG. 15. From left to right: visualization of the d-level wavelet-transform matrix (WTM) with level $d \in \{1, 2, 3\}$ at scale $k = 6$ for the Daubechies wavelet with index $\mathcal{K} = 3$; size of matrix is $2^k \times 2^k$. The number of nonzero elements in each row of the one-level WTM is $2\mathcal{K}$. The NNZ elements in those rows of the two-level WTM that pertain to the first level, i.e., the bottom half rows, is $2\mathcal{K}$, and the NNZ elements in the rows pertaining to the second level, the top half rows, is $4\mathcal{K}$. For the three-level wavelet transform, the NNZ elements in those rows pertaining to the third level is 2 times the NNZ elements in the rows pertaining to the second level, which itself is two times the NNZ on the rows pertaining to the first level. The NNZ elements increases by a factor of 2 when we increase the level of wavelet transform by one.

where $\mathbf{W}_d^{(k)}$ is the d-level wavelet-transform matrix at scale k; see Appendix A. This equation yields

$$\hat{a}_{w;\ell}^{(r)\dagger} = \sum_{\ell'} W_{d;\ell\ell'}^{(k)} \hat{a}_{s;\ell'}^{(k)\dagger}, \tag{135}$$

for any $s_0 \leq r < k$.

The NNZ coefficients in this summation depends on two parameters: the wavelet index $\mathcal{K}$ and the difference $(k - r)$ between the finest scale k and the scale r. Specifically, the NNZ coefficients is $2^{k-r}\mathcal{K}$ for any $r < k$. This relation follows from the recursive relation for the scaling and wavelets function at different scales described by Eqs. (1) and (2), respectively. By Eq. (2), each wavelet function at a particular scale r is a linear combination of $2\mathcal{K}$ scaling functions at one higher scale $r + 1$. Similarly, by Eq. (1), each scaling function at a given scale can be written as a linear combination of $2\mathcal{K}$ scaling functions at one higher scale. Therefore, a wavelet function at scale r is a linear combination of $2^{k-r}\mathcal{K}$ scaling functions at scale $k > r$. See Fig. 15 for a visual insight into the relationship between the NNZ coefficients, $(k - r)$ and $\mathcal{K}$.

By the combination of Eqs. (135), (132), and (129), we have

$$\hat{H}_{w;\,\ell}^{(r)} = \sum_{\ell'} W_{d;\ell\ell'}^{(k)} \hat{H}_{s;\ell'}^{(k)}, \tag{136}$$

hence the time evolution induced by this Hamiltonian for time $t = \pi/2$ in the Fourier approach yields

$$e^{-\mathrm{i}\hat{H}_{w;\ell}^{(r)}\pi/2} \left| G_{\mathrm{scale}}^{(k)} \right\rangle |0\rangle_{\mathtt{anc}} = -\mathrm{i} \left| w_\ell^{(r)} \right\rangle |1\rangle_{\mathtt{anc}}, \tag{137}$$

and, therefore, we obtain the wavepacket state in Eq. (131) up to a global phase. Because of the bosonic commutation relations between the bosonic creation and annihilation operators in Eq. (129), any term in the right-hand side of Eq. (136) commutes with other terms. Consequently, we have the decomposition

$$e^{-\mathrm{i}\hat{H}_{w;\,\ell}^{(r)}\pi/2} = \prod_{\ell'} e^{-\mathrm{i}\hat{H}_{s;\,\ell'}^{(k)} W_{d;\,\ell\ell'}^{(k)}\pi/2}, \tag{138}$$

for the time evolution induced by the wavelet Hamiltonian (136). By this decomposition, simulating time evolution induced by the wavelet Hamiltonian for constant time $\pi/2$ in the Fourier approach is achieved by simulating time evolution induced by the scale Hamiltonian (129) with $2^{k-r}\mathcal{K}$ different ℓ' for constant time $t_{\ell'} := W_{d;\,\ell\ell'}^{(k)}\pi/2$. Therefore, generating a single-particle state at scale r in the Fourier approach is $\Theta(2^{k-r}\mathcal{K})$ times more expansive than generating the same state in the wavelet approach.

V. DISCUSSION

We have established two quasilinear quantum algorithms, one Fourier-based and the other wavelet-based, to generate an approximation for the ground state of a massive scalar bosonic free QFT. Specifically, each of the two algorithms' time complexity is quasilinear with respect to the number of modes in the discretized QFT. Our algorithms deliver a superquadratic speedup over the state-of-the-art quantum algorithm for ground-state generation and are optimal up to polylogarithmic factors. The Fourier-based algorithm is limited to translationally invariant QFTs. By numerical simulations, we have shown that

the wavelet-based algorithm successfully yields the ground state for a QFT with a broken translational invariance.

We have also developed two quantum algorithms for generating one-dimensional Gaussian states. Our first algorithm is based on the Kitaev-Webb method [19] for preparing a 1DG state, which itself is an application of the standard state-preparation method by Zalka [13], Grover and Rudolph [14]. The Kitaev-Webb method, however, is restricted to 1DG states that possess an extremely large standard deviation, whereas our algorithm generates a 1DG state with any standard deviation. Our second quantum algorithm for generating a 1DG state is based on inequality testing [15] that mitigates the number of arithmetic operations required by the standard state-preparation method.

Methodologically, in the Fourier-based algorithm, we discretize the continuum free QFT in a fixed-scale basis. The ground-state ICM for the discretized QFT has a circulant structure, and we utilize this structure to establish a quasilinear quantum algorithm for ground-state generation. In the wavelet-based algorithm, we discretize the continuum-free QFT in a multiscale wavelet basis. In this case, the ground-state ICM is a quasisparse matrix. Specifically, most elements of this matrix are nearly zero, and we replace these values with exactly zero. This replacement enables a fingerlike sparse structure for the ground-state ICM that we exploit to achieve a quasilinear time complexity for the wavelet-based algorithm.

We went beyond ground-state generation and constructed procedures for preparing free-field wavepackets in the Fourier- and wavelet-based approaches. We showed that, unlike the Fourier approach, the wavelet approach enables preparing particle states at different energy scales without an additional cost required for the Fourier approach. Specifically, we showed that preparing a free-field single-particle state at scale with index r is $\Theta(2^{k-r}\mathcal{K})$ times more expensive than preparing the same state in the wavelet approach, where $\mathcal{K}$ is the wavelet index and $k > r$ is the finest scale's index. The wavelet approach's cheaper cost to preparing states beyond the free-field ground state suggests that this approach is advantageous over the Fourier approach in other aspects of simulating a QFT. Moreover, as shown by the numerical simulations, the wavelet-based approach is applicable to field theories with broken translational invariance due to inhomogeneous mass, suggesting the wavelet-based approach allows simulating more general quantum field theories [12].

Our Fourier-based algorithm's key point is utilizing the circulant structure of the ground state's ICM due to the translational invariance of the free QFT. We note that the ground-state ICM in the Jordan-Lee-Preskill approach [1,2] has the same structure as the ICM in our Fourier-based approach. Hence our Fourier-based algorithm can be used for generating the ground state of the lattice QFT in the Jordan-Lee-Preskill approach and accelerate their algorithm for ground-state generation. However, in contrast to their approach, the coupling-matrix elements in our Fourier approach are exact. The discretization error, due to approximating the derivative operator in the free-field Hamiltonian by a discretized derivative operator, results in a nonexact coupling matrix in the Jordan-Lee-Preskill approach [1,2]. Consequently, the Fourier approach could be preferred over the lattice approach for generating the free-field ground state.

In developing our ground-state-generation algorithms, we cared only about producing quasilinear algorithms and opted to focus on their readability rather than optimizing their performance. Therefore, our algorithms' time complexity could be improved, but any improvement will not change the quasilinear scaling of the algorithms' complexities.

Our Fourier- and wavelet-based algorithms have classical preprocessing and quantum routine. The classical preprocessing of each algorithm produces a certain amount of classical information required for executing the quantum routine of the algorithm. We analyzed not only the quantum complexity but also the classical complexity of our state-generation algorithms in order to ensure that the resulting procedures are indeed quasilinear in the number of modes of the discretized QFT. We established the classical time complexities

$$T_{\mathrm{FBA}}^{(\mathrm{C})} \in \mathcal{O}\left(\mathcal{K}^3 + \mathcal{K}N \log_2 \log_2 \left(\frac{N}{\sqrt{m_0 \varepsilon_{\mathrm{vac}}}}\right)\right),$$
$$T_{\mathrm{WBA}}^{(\mathrm{C})} \in \mathcal{O}(\mathcal{K}^3 + \mathcal{K}^2 N \log_2 N), \tag{139}$$

for classical preprocessing of the Fourier- and wavelet-based algorithms, respectively, and the quantum time complexities

$$T_{\mathrm{FBA}}^{(\mathrm{Q})} \in \mathcal{O}\left(N \log_2 N + N \log_2^3 \left(\frac{N}{\sqrt{m_0 \varepsilon_{\mathrm{vac}}}}\right)\right),$$
$$T_{\mathrm{WBA}}^{(\mathrm{Q})} \in \mathcal{O}\left(\left(\frac{N\mathcal{K}}{m_0}\right) \log_2^2 \left(\frac{N\mathcal{K}}{m_0 \varepsilon_{\mathrm{vac}}}\right) + N \log_2^3 \left(\frac{N}{\sqrt{m_0 \varepsilon_{\mathrm{vac}}}}\right)\right), \tag{140}$$

for their quantum routine; all parameters are specified in Table II.

In contrast to the usual approach of using gate complexity, i.e., the number of low-level quantum operations in an algorithm, as a metric to analyze time complexity for a quantum algorithm, we have analyzed our quantum algorithm's time complexity with respect to high-level operations. The high-level quantum operations in our complexity analysis are similar to the high-level operations in the classical random-access machine model. In particular, we have assigned a unit cost to basic arithmetic

operations such as multiplication and addition on a quantum computer. By the high-level operations, we avoid the implementation details of these operations in the compilation step. Nevertheless, our algorithm's gate complexity remains quasilinear with respect to the number of modes, as the high-level operations are executed on quantum registers with size logarithmic in the number of modes. However, the power of logarithmic factors in our algorithms' gate complexity depends on implementations of the high-level operations, specifically quantum multiplication, as other basic arithmetic operations are cheaper to implement than multiplication. Whether to use schoolbook multiplication á la Häner, Roettler, and Svore [53] or Karatsuba multiplication á la Gidney [54] or even as-yet-undeveloped quantum multiplication algorithms based on asymptotically efficient classical multiplication algorithms such as the Schönhage-Strassen algorithm [55, Sec. 4.3], our algorithms' gate complexity stay quasilinear in the number of discretized QFT modes.

VI. CONCLUSIONS

Free-field ground-state generation is a bottleneck for the prior approaches to simulating a massive scalar bosonic QFT on a quantum computer. In this paper, we have established two quantum algorithms for generating an approximation for the free-field ground state with a quasilinear gate complexity in the discretized QFT number of modes. Our algorithms provide a superquadratic speedup over the prior approaches and overcome the ground-state-generation bottleneck in simulating a massive scalar bosonic QFT. We have shown that our ground-state-generation algorithms are optimal up to polylogarithmic factors. In particular, we have proved that any state-generation algorithm with a sublinear time complexity will result in an exponentially bad approximation, with respect to the number of modes, for the free-field ground state.

We have compared the Fourier- and wavelet-based algorithms and shown that the wavelet-based algorithm is advantageous over the alternative Fourier-based algorithm for two cases where we go beyond ground-state generation and translationally invariant QFTs. Specifically, for beyond ground-state generation, we have shown that the wavelet-based algorithm enables generating particle states at different energy scales directly, whereas the Fourier-based algorithm allows only direct preparation for particle states at a fixed energy scale. Preparing a particle state at different scales by the Fourier-based algorithm requires further transformations that add to the cost of initial-state generation for simulating the QFT. The Fourier-based algorithm is limited to translationally invariant QFTs. We have shown by numerical simulation that our wavelet-based algorithm is applicable to field theories with broken translational invariance due to inhomogeneous mass, suggesting the wavelet-based approach allows simulating more general quantum field theories.

We have also developed two quantum algorithms for generating a one-dimensional Gaussian state, which is required for preparing the free-field ground state—a multidimensional Gaussian state. Our first algorithm for 1DG-state generation is based on the standard state-preparation method [13,14], and the second algorithm is based on inequality testing [15] that mitigates the number of arithmetic operations required in the standard method for generating a 1DG state. Our inequality-testing-based method is more practical than the Zalka-Grover-Rudolph method [13,14], i.e., the standard state-preparation method, and can be used broadly in the state-preparation subroutine of quantum-simulation algorithms.

The key point of our ground-state-generation algorithms is to utilize the circulant structure and the fingerlike sparse structure for a differential operator represented in a fixed- and multiscale wavelet basis, respectively. Our techniques for exploiting these structures could be used more broadly in the quantum simulation of continuous classical or quantum systems whose Hamiltonian involves differential operators, including fermionic QFT [3], as well as quantum-chemistry simulations [56]. Our methods, particularly the wavelet-based method, could be replaced with finite-difference or finite-element methods to improve quantum algorithms for partial differential equations [57–59].

We have focused on ground-state generation here, but our methods could be employed to improve other aspects of simulating a QFT, namely time evolution and measurement. In particular, akin to particle-state generation at different energy scales, the wavelet approach could be advantageous over the Fourier approach for measuring mean momentum of a particle state at the final step of a QFT simulation.

We have particularly used Daubechies wavelets in developing our state-generation algorithms. These algorithms, however, work for any compactly supported wavelets that are differentiable. A future direction is to see how other types of compactly supported wavelets could improve the state-generation algorithms. In particular, least-asymmetric wavelets [28, pp. 254–257] also known as symlets [60, p. 27], a variant of Daubechies wavelets with nearly symmetrical basis functions, could result in a more sparse ground-state ICM for a given threshold value, Eq. (50), and improve the wavelet-based algorithm.

ACKNOWLEDGMENTS

This project is supported by the Government of Alberta and by the Natural Sciences and Engineering Research Council of Canada (NSERC). M.B. and B.C.S. acknowledge the traditional owners of the land on which some

of this work was undertaken at the University of Calgary: the Treaty 7 First Nations. Y.R.S., D.W.B., and G.K.B. acknowledge the Wallamattagal people of the Dharug nation, whose cultures and customs have nurtured, and continue to nurture, the land on which some of this work was undertaken: Macquarie University. Y.R.S. acknowledges the Gadigal and Guring-gai people of the Eora Nation upon whose ancestral lands the University of Technology Sydney now stands. M.B. thanks Mehdi Ahmadi for many helpful discussions. Y.R.S. is supported by Australian Research Council Grant No. DP200100950. D.W.B. worked on this project under a sponsored research agreement with Google Quantum AI. D.W.B. is also supported by Australian Research Council Discovery Projects DP190102633 and DP210101367. G.K.B. acknowledges support from the ARC from Grant No. DP200102152.

APPENDIX A: WAVELET TRANSFORM

In this Appendix, we review a family of basis transforms called "the" discrete wavelet transform. The specific member of that family of transforms will be clear from the context. Here we focus only on the one-dimensional wavelet transforms.

Two parameters specify a one-dimensional wavelet transform. The first parameter is a choice k of scale for $\mathscr{L}^2(\mathbb{R})$. The second parameter is a choice of decomposition level $d \geq 1$. The d-level wavelet transform at scale k is recursively defined as follows. The one-level wavelet transform at any scale k is defined to be the canonical isomorphism

$$\mathbf{W}^{(k)} : \mathcal{S}_k \to \mathcal{S}_{k-1} \oplus \mathcal{W}_{k-1}, \quad \mathbf{W}_1^{(k)} := \mathbf{W}^{(k)}, \tag{A1}$$

where $\mathcal{S}_k$ and $\mathcal{W}_k$ are the scale and wavelet subspaces, respectively, defined in Sec. II A. The d-level wavelet transform, for $d > 1$, is defined as

$$\mathbf{W}_d^{(k)} := \left(\mathbf{W}^{(k-d+1)} \oplus \mathbb{1}_{\mathcal{S}_{k-d+1}^{\perp}}\right) \cdot \mathbf{W}_{d-1}^{(k)}, \tag{A2}$$

where $\mathcal{S}_{k-d}^{\perp} := \mathcal{W}_{k-d+1} \oplus \mathcal{W}_{k-d+2} \oplus \cdots \oplus \mathcal{W}_{k-1}$ is the orthogonal complement of $\mathcal{S}_{k-d}$ considered as a subspace of $\mathcal{S}_k$. Thus $\mathbf{W}_d^{(k)}$ executes the sequence of transformations

$$\begin{aligned}
\mathcal{S}_k &\xrightarrow{\mathbf{W}^{(k)}} \mathcal{S}_{k-1} \oplus \mathcal{W}_{k-1} \\
&\xrightarrow{\mathbf{W}^{(k-1)} \oplus \mathbb{1}_{\mathcal{S}_{k-1}^{\perp}}} \mathcal{S}_{k-2} \oplus \mathcal{W}_{k-2} \oplus \mathcal{W}_{k-1} \\
&\xrightarrow{\mathbf{W}^{(k-2)} \oplus \mathbb{1}_{\mathcal{S}_{k-2}^{\perp}}} \mathcal{S}_{k-3} \oplus \mathcal{W}_{k-3} \oplus \mathcal{W}_{k-2} \oplus \mathcal{W}_{k-1} \\
&\quad \vdots \\
&\xrightarrow{\mathbf{W}^{(k-d+1)} \oplus \mathbb{1}_{\mathcal{S}_{k-d+1}^{\perp}}} \mathcal{S}_{k-d} \oplus \mathcal{W}_{k-d} \oplus \cdots \oplus \mathcal{W}_{k-1}.
\end{aligned} \tag{A3}$$

The action of the canonical isomorphism $\mathbf{W}^{(k)}$ can be expressed in terms of two infinite matrices

$$\mathbf{H} = \begin{bmatrix}
 & \vdots & \vdots & \vdots & \vdots & \vdots & \\
\cdots & h_0 & h_1 & h_2 & h_3 & h_4 & \cdots \\
\cdots & h_{-2} & h_{-1} & h_0 & h_1 & h_2 & \cdots \\
\cdots & h_{-4} & h_{-3} & h_{-2} & h_{-1} & h_0 & \cdots \\
 & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{bmatrix},$$

$$\mathbf{G} = \begin{bmatrix}
 & \vdots & \vdots & \vdots & \vdots & \vdots & \\
\cdots & g_0 & g_1 & g_2 & g_3 & g_4 & \cdots \\
\cdots & g_{-2} & g_{-1} & g_0 & g_1 & g_2 & \cdots \\
\cdots & g_{-4} & g_{-3} & g_{-2} & g_{-1} & g_0 & \cdots \\
 & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{bmatrix}, \tag{A4}$$

called the low-pass and high-pass filter matrix, respectively. These infinite matrices are to be multiplied to vectors $\mathbf{c}^{(k)} \in \mathcal{S}_k$ expressed in the basis $\left\{s_\ell^{(k)}\right\}$, meaning that these vectors take the form

$$\mathbf{c}^{(k)} = \begin{bmatrix}
\vdots \\
c_{-1}^{(k)} \\
c_0^{(k)} \\
c_1^{(k)} \\
\vdots
\end{bmatrix}, \quad c_\ell^{(k)} := \left\langle s_\ell^{(k)} \middle| f \right\rangle. \tag{A5}$$

The infinite matrix $\mathbf{H}$ is then defined so that $\mathbf{H} \cdot \mathbf{c}^{(k)} = \mathbf{c}^{(k-1)}$. Note that the form of $\mathbf{H}$ and $\mathbf{G}$ do not depend on k. Also note that $\mathbf{H}$ and $\mathbf{G}$ are both row-sparse operators if the wavelet is compactly supported, as only a finite number of h and g coefficients are nonzero. With these definitions, we have $\mathbf{W}^{(k)} \cdot \mathbf{c}^{(k)} = \left(\mathbf{H} \cdot \mathbf{c}^{(k)}\right) \oplus \left(\mathbf{G} \cdot \mathbf{c}^{(k)}\right)$.

We do not work with an infinite matrix representation in this paper and instead restrict attention to finite-sized systems. Specifically, we restrict attention to subset $\mathscr{L}^2(\mathbb{S})$ of $\mathscr{L}^2(\mathbb{R})$ that has support only on unit interval $\mathbb{S} := \{x \in \mathbb{R} | 0 \leq x \leq 1\}$. In this restricted space, the scale coefficient $c_\ell^{(k)} := \left\langle s_\ell^{(k)} \middle| f \right\rangle$ for the function $f \in \mathscr{L}^2(\mathbb{S})$ is guaranteed to be equal to zero for all but a finite number of values of ℓ. Specifically, the fact that

$$\operatorname{supp}\left(s_\ell^{(k)}\right) \subseteq \{x \in \mathbb{R} | 0 < 2^k x - \ell < 2\mathcal{K} - 1\}, \tag{A6}$$

where supp(•) refers to support of a function, implies that $c_\ell^{(k)} = 0$ if $\ell < 1 - 2\mathcal{K}$ or $\ell \geq 2^k$. That is to say, only $2^k + 2\mathcal{K} + 1$ scale coefficients $c_\ell^{(k)}$ could be nonzero. We thus treat the infinite-dimensional vector $\mathbf{c}^{(k)}$ as though it were finite-dimensional with dimension $2^k + 2\mathcal{K} + 1$. Similarly, we treat $\mathbf{H}$ and $\mathbf{G}$ as though they were $(2^{k-1} + 2\mathcal{K} + 1) \times$

$(2^k + 2\mathcal{K} + 1)$ matrices and hence $\mathbf{W}^{(k)}$ as though it were a $(2^k + 4\mathcal{K} + 2) \times (2^k + 2\mathcal{K} + 1)$ matrix. Applying $\mathbf{W}^{(k)}$ to a vector is described as calculating the "discrete wavelet transform" of the vector.

The treatment described here leaves $\mathbf{W}^{(k)}$ as a nonsquare matrix, which cannot be an automorphism of $\mathcal{S}_k \subset \mathscr{L}^2(\mathbb{S})$. We deal with this issue in two ways. In the first way, which we refer to as the "open boundaries" case, we discard the negative index values $\ell < 0$. Although this leaves us with a square $(2^k \times 2^k)$ matrix, the result is not invertible and hence is not an automorphism of $\mathcal{S}_k \subset \mathscr{L}^2(\mathbb{S})$ as desired. Despite its weaknesses, this approach is good enough for numerical studies. The second way we deal with the nonsquare $\mathbf{W}^{(k)}$ is what we call the "(anti)periodic boundaries" case. In this approach, we treat the signal f as though it depicts a periodic (antiperiodic) system, so the problematic negative index values $\ell < 0$ would be identical to (the negative of) the index values $2^k - \ell$. In this case, we apply the same condition to the derivative overlap coefficients, Eq. (13). By putting these two approaches together, the matrix $\mathbf{W}^{(k)}$ for the db3 wavelets becomes the $2^k \times 2^k$ matrix

$$\mathbf{W}^{(k)} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & 0 & 0 & \cdots \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & \cdots \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ bh_4 & bh_5 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ bh_2 & bh_3 & bh_4 & bh_5 & 0 & 0 & 0 & 0 & \cdots \\ g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & 0 & 0 & \cdots \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & \cdots \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ bg_4 & bg_5 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ bg_2 & bg_3 & bg_4 & bg_5 & 0 & 0 & 0 & 0 & \cdots \end{bmatrix} = \begin{bmatrix} \mathbf{H} \\ \mathbf{G} \end{bmatrix}, \tag{A7}$$

where $b = 0, +1, -1$ depending on whether the boundaries are open, periodic, or antiperiodic.

APPENDIX B: LOW-PASS FILTER FOR DAUBECHIES WAVELETS

This Appendix describes a method for computing the low-pass filter coefficients h_ℓ, Eq. (1), for Daubechies wavelets. We begin with describing the method and then construct a classical algorithm for computing the low-pass filter coefficients using the described method. Finally, we discuss the algorithm's time complexity with respect to the classical primitive operations described in Sec. III A 3.

Several methods are used to compute the numerical values of the low-pass filters h_ℓ, Eq. (1). A direct method is to solve the system of nonlinear equations [29, p. 3]

$$\begin{aligned} &\sum_\ell h_\ell = \sqrt{2}, \quad \sum_\ell h_\ell h_{\ell-2\ell'} = \delta_{0,\ell'}, \\ &\sum_\ell (-1)^\ell \ell^{\ell'} h_{2\mathcal{K}-1-\ell} = 0 \quad \forall \ell' < \mathcal{K}, \end{aligned} \tag{B1}$$

which are derived from various properties of the Daubechies wavelets, such as orthonormality. This nonlinear system can be solved analytically for $\mathcal{K} \leq 3$ and numerically otherwise. However, the numerical methods for solving the nonlinear system become increasingly inefficient as $\mathcal{K}$ increases. We use an alternative method proposed by Daubechies to compute the numerical values of the filter coefficients [28]; also see Ref. [61, p. 81]. The filter coefficients in Daubechies' method are obtained by first computing the $2\mathcal{K} - 2$ roots of the polynomial

$$\begin{aligned} f(z) &:= z^{\mathcal{K}-1} P\left(\frac{1}{2} - \frac{1}{4z} - \frac{z}{4}, \mathcal{K}\right), \\ P(y, \mathcal{K}) &:= \sum_{\ell=0}^{\mathcal{K}-1} \binom{\mathcal{K}-1+\ell}{\ell} y^\ell, \end{aligned} \tag{B2}$$

where $P(y, \mathcal{K})$ is a polynomial of degree $\mathcal{K} - 1$. Next those roots z_ℓ with $|z_\ell| < 1$ are selected. The filter coefficients are then obtained by identifying the coefficients H_ℓ in

$$(1+z)^{\mathcal{K}} \prod_{\ell=0}^{\mathcal{K}-1} (z - z_\ell) = \sum_{\ell=0}^{2\mathcal{K}-1} H_\ell z^{2\mathcal{K}-1-l}, \tag{B3}$$

for selected roots and normalizing the coefficients as $h_\ell = H_\ell/\sqrt{\mathbf{H}\cdot\mathbf{H}}$, where $\mathbf{H}$ is a vector with elements H_ℓ.

The pseudocode in Algorithm 10 describes a formal algorithm for computing the low-pass filters using Daubechies' method. This algorithm is used in classical preprocessing of the wavelet-based algorithm (Sec. IV A), specifically as a subroutine in Algorithm 4, which requires the low-pass filters to compute unique elements of the ground-state ICM in a multiscale wavelet basis.

The time complexity of this algorithm is dominated by the time complexity for computing roots of $f(z)$ in Eq. (B2) because the root-finding subroutine in line (4) is the computationally expensive part of the algorithm. The coefficients of $f(z)$, a polynomial of degree $2\mathcal{K}-2$, have a magnitude less than $2^{3\mathcal{K}}$. By these properties, the arithmetic complexity for finding roots of $f(z)$ with precision p is $\mathcal{O}(\mathcal{K}\log^5\mathcal{K}\log(3\mathcal{K}+p))$ using a known root-finding algorithm [62]. The classical primitive operations in the cost model described in Sec. III A 3 include basic arithmetic operations. Therefore, with respect to the classical primitives, the time complexity for computing the low-pass filter coefficients is quasilinear in the wavelet index.

APPENDIX C: DERIVATIVE OVERLAP COEFFICIENTS

In this Appendix, we describe Beylkin's method [44] for computing the derivative overlap coefficients, Eq. (13), for the Daubechies wavelet with the wavelet index $\mathcal{K}$. For our application, we restrict the method to computing the overlaps coefficients for the second-order derivative operator. Beylkin's method, however, is general and could be used to compute derivative overlaps for derivative operators with any integer or fractional order. Having described the method, we then present a formal algorithm for computing the second-order derivative overlaps.

The explicit expression for the Daubechies scaling function and its derivatives are unknown. Therefore, the numerical value of the derivative overlaps cannot be directly computed from Eq. (13). Beylkin's method is an indirect way to compute these coefficients. The second-order derivative overlaps in this method satisfy the system of linear algebraic equations [44, Proposition 2]

$$\sum_{\ell\in\mathbb{Z}}\ell^2 x_\ell = 2,$$
$$x_\ell = 4x_{2\ell} + 2\sum_{k=1}^{\mathcal{K}} a_{2k-1}\left(x_{2\ell-(2k+1)} + x_{2\ell+2k+1}\right) \quad \forall \ell\in\mathbb{Z}, \tag{C1}$$

where the coefficients

$$a_{2k-1} := \frac{(-1)^{k-1}C}{(\mathcal{K}-k)!(\mathcal{K}+k-1)!(2k-1)} \quad \forall k\in\{1,\ldots,\mathcal{K}\},$$
$$C := \left[\frac{(2\mathcal{K}-1)!}{(\mathcal{K}-1)!4^{\mathcal{K}-1}}\right]^2, \tag{C2}$$

are autocorrelation of the low-pass filter h_ℓ, Eq. (1). If $\mathcal{K}\geq 2$, then the system of linear equations has a unique solution with $x_\ell \neq 0$ for $2\mathcal{K}-2\leq\ell\leq 2\mathcal{K}-2$, and $x_\ell = x_{-\ell}$. Using these properties, we write the linear system in

Input:
$\mathcal{K}\in\mathbb{Z}^+$ ▷ Daubechies wavelet index
$p\in\mathbb{Z}^+$ ▷ working precision
Output:
$\mathbf{h}\in\mathbb{R}^{2\mathcal{K}}$ ▷ low-pass filter (1) for the Daubechies wavelet with index $\mathcal{K}$
1: **function** POLY($y,\mathcal{K}$) ▷ defines the polynomial $P(y,\mathcal{K})$ in Eq. (B2)
2: **return** $\sum_{\ell=0}^{\mathcal{K}-1}\binom{\mathcal{K}-1+l}{\ell}y^\ell$
3: **function** LOWPASSFILTER($\mathcal{K},p$)
4: $\mathbb{R}^{2\mathcal{K}-2}\ni\mathbf{r}\leftarrow$ ROOTS$\left(z^{\mathcal{K}-1}\text{POLY}(\frac{1}{2}-\frac{1}{4z}-\frac{z}{4},\mathcal{K}),\,p\right)$ ▷ computes roots of $f(z)$ in Eq. (B2) with precision p
5: $\mathbf{z}\leftarrow\varnothing$ ▷ initializes an empty list
6: **for** $\ell\leftarrow 0$ to $2\mathcal{K}-3$ **do** ▷ appends roots r_ℓ of $f(z)$ with $|r_\ell|<1$ to $\mathbf{z}$
7: **if** $|r_\ell|<1$ **then**
8: append r_ℓ to $\mathbf{z}$
9: **for** $\ell\leftarrow 0$ to $\mathcal{K}-1$ **do** ▷ appends roots of $(1+z)^{\mathcal{K}}$ to $\mathbf{z}$; see Eq. (B3)
10: append -1 to $\mathbf{z}$
11: $\mathbb{R}^{2\mathcal{K}}\ni\mathbf{H}\leftarrow$ POLYCOEFF($\mathbf{z}$) ▷ computes coefficients H_ℓ (B3) of a polynomial with roots in $\mathbf{z}$
12: $\mathbf{h}\leftarrow\mathbf{H}/\sqrt{\mathbf{H}\cdot\mathbf{H}}$
13: **return h**

Algorithm 10. Classical algorithm for computing the low-pass filter for Daubechies wavelets

the matrix-vector form as $\mathbf{Mx} = \mathbf{b}$, where $\mathbf{M}$ is a $(2\mathcal{K}-1)$-by-$(2\mathcal{K}-1)$ matrix with elements

$$M_{mn} = n^2 + 4\delta_{2m,n} - \delta_{m,n} + 2\sum_{k=1}^{\mathcal{K}} a_{2k-1}\left(\delta_{n,|2m-(2k+1)|} + \delta_{n,2m+2k+1}\right), \quad \text{(C3)}$$

the solution vector is $\mathbf{x} := (x_0, \ldots, x_{2\mathcal{K}-2})^\mathsf{T}$ and $\mathbf{b}$ is the vector of all 1's, i.e. $\mathbf{b} := (\mathbf{1}, \ldots, \mathbf{1})$.

The pseudocode in Algorithm 11 describes a formal algorithm for computing the derivative overlaps by Beylkin's method. The formal algorithm presented here is only used to simplify the description of our ground-state-generation algorithms (Sec. IV A).

APPENDIX D: PROOFS

This Appendix contains the statement and proofs of several propositions used in this paper.

Proposition 11: *Let N be a positive integer, $\{s_n(x) \mid n \in \{0, \ldots, N-1\}\}$ be a set of orthonormal, real-valued, differentiable, and compactly supported functions that span a subspace of $\mathscr{L}^2(\mathbb{R})$, and let $\mathbf{\Gamma} \in \mathbb{R}^{N\times N}$ be a matrix whose elements are*

$$\Gamma_{nm} := \int dx \frac{d}{dx} s_n(x) \frac{d}{dx} s_m(x) \quad \forall\, n, m \in \{0, \ldots, N-1\}. \quad \text{(D1)}$$

Then $\mathbf{\Gamma}$ is a symmetric and positive-semidefinite matrix.

Proof. The symmetry of $\mathbf{\Gamma}$ is immediate from the definition of its elements. By definition, an N-by-N symmetric real matrix $\mathbf{A}$ is positive semidefinite if $\boldsymbol{v}^\mathsf{T}\mathbf{A}\boldsymbol{v} \geq 0$ for all nonzero $\boldsymbol{v} \in \mathbb{R}^N$. We use this definition to prove that $\mathbf{\Gamma}$ is a positive-semidefinite matrix. Let $\boldsymbol{v}$ be a nonzero vector in $\mathbb{R}^N$ and let $f(x) := \sum_{n=0}^{N-1} v_n s_n(x)$, then

$$\begin{aligned}\boldsymbol{v}^\mathsf{T}\mathbf{\Gamma}\boldsymbol{v} &= \sum_{n,m=0}^{N-1} v_n \Gamma_{nm} v_m \\ &= \int dx \frac{d}{dx}\left(\sum_{n=0}^{N-1} v_n s_n(x)\right)\frac{d}{dx}\left(\sum_{n=0}^{N-1} v_m s_m(x)\right) \\ &= \int dx \left(\frac{d}{dx} f(x)\right)^2 \geq 0. \end{aligned} \quad \text{(D2)}$$

Thus, by definition, $\mathbf{\Gamma}$ is a positive-semidefinite matrix. ■

Proposition 12: *Let $m_0 \in \mathbb{R}^+$ be the free mass and $\mathbf{A} \in \mathbb{R}^{N\times N}$ be the ground-state* ICM *in a wavelet basis. Then the smallest eigenvalue of $\mathbf{A}$ is m_0.*

Proof. The spectrum of the ground-state ICM in a fixed- and multiscale wavelet basis are identical because they are unitarily equivalent. Therefore, we prove only this proposition for a fixed-scale ICM $\mathbf{A}_{ss}^{(k)}$, Eq. (24). This matrix is the principal square root of the fixed-scale coupling matrix in Eq. (23). The fixed-scale coupling matrix can be written as

$$\mathbf{K}_{ss}^{(k)} = m_0^2\mathbb{1} - 4^k\mathbf{\Delta}^{(2)} = m_0^2\mathbb{1} + 4^k\mathbf{\Gamma}, \quad \text{(D3)}$$

where $\mathbf{\Delta}^{(2)}$ is a matrix whose elements are the second-order derivative overlaps $\Delta_{mn}^{(2)}$, Eq. (13), and the second identity in Eq. (D3) follows from Eq. (D1) and Eq. (13). Eigenvalues of $\mathbf{K}_{ss}^{(k)}$ are

$$\lambda_j^{(k)} := m_0^2 - 4^k\Delta_0^{(2)} - 2\sum_{\ell=1}^{2\mathcal{K}-1} 4^k\Delta_\ell^{(2)}\cos\left(\frac{2\pi\ell}{N}j\right), \quad \text{(D4)}$$

Input:
$\mathcal{K} \in \mathbb{Z}^+$ ▷ wavelet index
$p \in \mathbb{Z}^+$ ▷ working precision
Output:
$\mathbf{\Delta} \in \mathbb{R}^{2\mathcal{K}-1}$ ▷ second-order derivative overlaps (13)

function DERIVATIVEOVERLAPS($\mathcal{K}, p$)
  $C \leftarrow \left[\frac{(2\mathcal{K}-1)!}{(\mathcal{K}-1)!4^{\mathcal{K}-1}}\right]^2$ ▷ see Eq. (C2)
  for k from 1 to $\mathcal{K}$ **do**
    $a_{2k-1} \leftarrow \frac{(-1)^{k-1}C}{(\mathcal{K}-k)!(\mathcal{K}+k-1)!(2k-1)}$ ▷ computes the auto-correlation coefficients in Eq. (C2)
    $b_{k-1} \leftarrow 1$
  $\mathbf{M} \leftarrow$ MTX($\mathcal{K}$) ▷ MTX is a function that constructs the matrix $\mathbf{M}$ by Eq. (C3)
  $\mathbf{\Delta} \leftarrow \mathbf{M}^{-1}\mathbf{b}$
  return $\mathbf{\Delta}$

Algorithm 11. Classical algorithm for computing the second-order derivative overlaps in Eq. (13)

which yields

$$\lambda_0^{(k)} = m_0^2 - 4^k \Delta_0^{(2)} - 2\sum_{\ell=1}^{2\mathcal{K}-1} 4^k \Delta_\ell^{(2)} = m_0^2, \tag{D5}$$

where the second identity follows from properties of the second-order derivative overlaps [33, Appendix A]. By Eqs. (D3), (D5), and Proposition 11, m_0^2 is the smallest eigenvalue of the coupling matrix and therefore m_0 is the smallest eigenvalue of the ground-state ICM in both fixed- and multiscale wavelet basis. ■

Proposition 13: (Bound on determinant of near-identity matrices [63].) *Let* $N \in \mathbb{Z}^+$, $\varepsilon \in [0,1)$ *and* $\mathbf{A} = \mathbb{1} - \mathbf{E} \in \mathbb{R}^{N\times N}$ *such that* $|E_{ij}| \le \varepsilon$ *for all* $i,j \in \{0,\dots,N-1\}$. *If* $N\varepsilon \le 1$, *then*

$$1 - N\varepsilon \le \det(\mathbf{A}) \le (1-N\varepsilon)^{-1}. \tag{D6}$$

Proposition 14: *Let* $J \in 2\mathbb{Z}^+$, $\sigma \in \mathbb{R}^+$ *and* $\delta \le \min(1/2, \sigma)$. *Define* $r := \delta/(\sigma\sqrt{2})$ *and*

$$\mathcal{N}^2 := \int_{\mathbb{R}} dx\, e^{-x^2/2\sigma^2} = \sigma\sqrt{2\pi}, \quad \tilde{\mathcal{N}}^2 := \delta^2 \sum_{j=-J/2}^{J/2-1} e^{-j^2 r^2}. \tag{D7}$$

Then

$$\frac{1}{\mathcal{N}\tilde{\mathcal{N}}} \sum_{j=-J/2}^{J/2-1} \delta\, e^{-j^2r^2} \ge \frac{1}{\mathcal{N}^2} \int_{-J\delta/2}^{J\delta/2} dx\, e^{-\frac{x^2}{2\sigma^2}}. \tag{D8}$$

Proof. Let

$$p := 1 + e^{\pi - 2\pi^2}\left(\frac{\pi^{1/4}}{\Gamma(3/4)} - 1\right), \tag{D9}$$

where Γ is the Γ function. We first prove that

$$\tilde{\mathcal{N}} \le \sqrt{p\delta}\,\mathcal{N}, \tag{D10}$$

$$\sqrt{\delta/p} \sum_{j=-J/2}^{J/2-1} e^{-j^2r^2} \ge \int_{-J\delta/2}^{J\delta/2} dx\, e^{-x^2/2\sigma^2}. \tag{D11}$$

Proposition 14 follows from Eqs. (D10) and (D11); Eq. (D10) implies that $\sqrt{p\delta}/(\tilde{\mathcal{N}}\mathcal{N}) \ge 1/\mathcal{N}^2$ and multiplying each side of this inequality by each side of Eq. (D11) yields Eq. (D8). We now prove Eq. (D10). Note that

$$\sum_{j=-J/2}^{J/2-1} e^{-j^2r^2} \le \sum_{j\in\mathbb{Z}} e^{-j^2r^2} = \frac{\sqrt{\pi}}{r} \sum_{k\in\mathbb{Z}} e^{-k^2\pi^2/r^2}, \tag{D12}$$

where the equality follows from the Poisson summation formula [64, p. 385]. We now find an upper bound for the right-hand side of Eq. (D12). Note that $r^2 \le 1/2$ so $e^{-k^2\pi^2/r^2} \le e^{-2k^2\pi^2}$, also $e^{-2k^2\pi^2} \le e^{\pi-2\pi^2}e^{-\pi k^2}$ for each $k \in \mathbb{Z}/\{0\}$, therefore,

$$\begin{aligned}\sum_{k\in\mathbb{Z}} e^{-k^2\pi^2/r^2} &\le \sum_{k\in\mathbb{Z}} e^{-2k^2\pi^2} \le 1 + e^{\pi-2\pi^2} \sum_{k\in\mathbb{Z}/\{0\}} e^{-\pi k^2} \\ &= 1 + e^{\pi-2\pi^2}\left(\frac{\pi^{1/4}}{\Gamma(3/4)} - 1\right) = p,\end{aligned} \tag{D13}$$

where we use $\sum_{k\in\mathbb{Z}} e^{-\pi k^2} = \pi^{1/4}/\Gamma(3/4)$ [65, p. 103]. By Eqs. (D7), (D12), and (D13)

$$\tilde{\mathcal{N}}^2 = \delta^2 \sum_{j=-J/2}^{J/2-1} e^{-j^2r^2} \le p\delta\mathcal{N}^2, \tag{D14}$$

which yields Eq. (D10). We now prove Eq. (D11). Note that

$$\sqrt{\delta/p} \sum_{j=-J/2}^{J/2-1} e^{-j^2r^2} = \sqrt{\delta/p}\left(1 + e^{-J^2r^2/4} + 2\sum_{j=1}^{J/2-1} e^{-j^2r^2}\right), \tag{D15}$$

and, as $e^{-x^2/(2\sigma^2)}$ is a convex function for $|x| \ge \sigma$,

$$\begin{aligned}&\int_{-J\delta/2}^{J\delta/2} dx\, e^{-x^2/2\sigma^2} \\ &= 2\int_0^{\delta} dx\, e^{-x^2/2\sigma^2} + 2\sum_{j=1}^{J/2-1} \int_{j\delta}^{(j+1)\delta} dx\, e^{-x^2/2\sigma^2} \\ &\le 2\delta + 2\delta \sum_{j=1}^{J/2-1} e^{-j^2r^2} - 2(1/2)\delta\left((1/\sqrt{e}) - e^{-J^2r^2/4}\right) \\ &= \delta\left((2 - 1/\sqrt{e}) + e^{-J^2r^2/4} + 2\sum_{j=1}^{J/2-1} e^{-j^2r^2}\right).\end{aligned} \tag{D16}$$

Note that if $\delta \le (2-1/\sqrt{e})^{-2}/p = 0.514\,998$ then $\delta^2(2 - 1/\sqrt{e})^2 \le \delta/p$ and therefore,

$$\delta(2 - 1/\sqrt{e}) \le \sqrt{\delta/p}. \tag{D17}$$

This inequality implies that the right-hand side of Eq. (D15) is greater than or equal to the right-hand side of Eq. (D16). Therefore, Eqs. (D15), (D16), (D17), and $\delta \le 1/2$ yields Eq. (D11). ■

Proposition 15: *Let* $d_{\max} \in \mathbb{R}^+$ *and* $d_{\min} \in \mathbb{R}^+$ *be, respectively, the largest and the smallest diagonal elements of*

the diagonal matrix $\mathbf{D}$ *in either the spectral or the* UDU *decomposition of a symmetric positive-definite matrix* $\mathbf{A} \in \mathbb{R}^{N\times N}$, *with* $N \in \mathbb{Z}^+$. *Also let* $\kappa \in \mathbb{R}^+$ *be the condition number of* $\mathbf{A}$, *quantified as the ratio of the largest to the smallest eigenvalues of* $\mathbf{A}$. *Then*

$$\frac{d_{\max}}{d_{\min}} \in \mathcal{O}(\kappa). \tag{D18}$$

Proof. If $\mathbf{D}$ is the diagonal matrix in the spectral decomposition of $\mathbf{A}$, then $d_{\max}$ and $d_{\max}$ are, respectively, the largest and the smallest eigenvalue of $\mathbf{A}$. Therefore, by definition,

$$\frac{d_{\max}}{d_{\min}} = \kappa. \tag{D19}$$

If $\mathbf{D}$ is the diagonal matrix in the UDU decomposition of $\mathbf{A}$, then

$$\frac{d_{\max}}{d_{\min}} \le \kappa. \tag{D20}$$

The proof of this inequality is as follows. Let $\lambda_{\max}(\mathbf{A})$ and $\lambda_{\min}(\mathbf{A})$ be, respectively, the largest and the smallest eigenvalues of $\mathbf{A}$ and let $\mathbf{e_i}$ be the i^{th} column of the N-by-N identity matrix. Then

$$\begin{aligned}\lambda_{\max}(\mathbf{A}) &= \max_{\|\mathbf{x}\|=1} \mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} \ge \max_i \mathbf{e}_i^{\mathsf{T}}\mathbf{A}\mathbf{e}_i = \max_i \left(\mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}}\right)_{ii} \\ &= \max_i \left(\mathbf{d}_i + \sum_{j>i} \mathbf{d}_j U_{ij}^2\right) \ge \max_i \mathbf{d}_i = \mathbf{d}_{\max},\end{aligned} \tag{D21}$$

$$\begin{aligned}\lambda_{\min}^{-1}(\mathbf{A}) &= \lambda_{\max}\left(\mathbf{A}^{-1}\right) \ge \max_i \left(\mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}}\right)_{ii}^{-1} \\ &= \max_i \left(d_i^{-1} + \sum_{j>i} d_j^{-1}\left(\mathbf{U}^{-1}\right)_{ij}^2\right) \ge \max d_i^{-1} \\ &= d_{\min}^{-1}.\end{aligned} \tag{D22}$$

The second equality in Eq. (D22) follows from the fact that the inverse of an upper unit-triangular matrix is an upper unit-triangular matrix [43, p. 220]. The last inequality in Eqs. (D21) and (D22) holds because $d_i > 0$ for the positive-definite matrix $\mathbf{A}$. Proposition 15 follows from Eqs. (D19) and (D20). ■

Proposition 16: *Let* $\kappa \in \mathbb{R}^+$ *be the condition number of the* ICM *for the ground state of a* N*-mode massive real scalar bosonic free* QFT *with mass* $m_0 \in \mathbb{R}^+$ *in a fixed- or multiscale wavelet basis. Then*

$$\kappa \in \Theta(N/m_0). \tag{D23}$$

Proof. Note that the ICM $\mathbf{A}$ in a multiscale wavelet has the same condition number as the ICM $\mathbf{A}_{ss}$ in a fixed-scale wavelet basis because $\mathbf{A}$ is obtained from $\mathbf{A}_{ss}$ by a wavelet transform. We therefore consider the condition number of $\mathbf{A}_{ss}$. Now using $\mathbf{A}_{ss} = \sqrt{\mathbf{K}_{ss}}$, where $\mathbf{K}_{ss}$, Eq. (23), is the coupling matrix in a fixed-scale wavelet basis, we have

$$\kappa = \sqrt{\frac{\lambda_{\max}(\mathbf{K}_{ss})}{\lambda_{\min}(\mathbf{K}_{ss})}}. \tag{D24}$$

We prove that

$$\lambda_{\min}(\mathbf{K}_{ss}) = m_0^2, \tag{D25}$$

$$\lambda_{\max}(\mathbf{K}_{ss}) \in \Theta(m_0^2 + N^2). \tag{D26}$$

Proposition 16 follows from Eqs. (D24) to (D26). To prove Eq. (D26), we find a lower and an upper bound for the largest eigenvalue of $\mathbf{K}_{ss}$. We use the Gershgorin circle theorem to find the upper bound [43, p. 388]. This theorem implies that

$$\lambda_{\max}(\mathbf{K}_{ss}) \le \max_i \left(K_{ss;ii} + R_i\right), \quad R_i := \sum_{j\ne i} |K_{ss;ij}|. \tag{D27}$$

Note that for any positive-definite matrix $\mathbf{K}_{ss}$ [43, p. 434]

$$|K_{ss;ij}|^2 \le K_{ss;ii}K_{ss;jj}, \tag{D28}$$

and by Eq. (23)

$$K_{ss;\,ii} = m_0^2 + N^2\Delta_0. \tag{D29}$$

Therefore,

$$|K_{ss;ij}| \le m_0^2 + N^2\Delta_0. \tag{D30}$$

Furthermore, $\mathbf{K}_{ss}$ is a banded circulant matrix with the lower and upper bandwidth $2\mathcal{K}-2$. Thus, using Eq. (D30),

$$R_i \le 2(2\mathcal{K}-2)(m_0^2 + N^2\Delta_0). \tag{D31}$$

Equations (D27), (D29), and (D31) yield

$$\lambda_{\max}(\mathbf{K}_{ss}) \le (4\mathcal{K}-3)(m_0^2 + N^2\Delta_0). \tag{D32}$$

We now find a lower bound for the largest eigenvalue. Note that

$$\lambda_{\max}(\mathbf{K}_{ss}) = \max_{\|\mathbf{x}\|=1} \mathbf{x}^{\mathsf{T}}\mathbf{K}_{ss}\mathbf{x} \ge \max_i \mathbf{e}_i^{\mathsf{T}}\mathbf{K}_{ss}\mathbf{e}_i = m_0^2 + N^2\Delta_0, \tag{D33}$$

where $\mathbf{e_i}$ is the i^{th} column of $\mathbb{1}_{N\times N}$ and in the last equality we use Eq. (D29). Equations (D32) and (D33) yield Eq. (D26). ■

Proposition 17: (Exponentially decaying ICM at a fixed scale.) *Let $m_0 \in \mathbb{R}^+$ be the free mass, $\mathcal{K} \in \mathbb{Z}_{\geq 3}$ be the wavelet index, and $r \in \mathbb{Z}_{\geq 0}$ be the scale index. Let $\mathbf{A}^{(r)}_{ss} \in \mathbb{R}^{L2^r \times L2^r}$, Eq. (24), with $L \in \mathbb{Z}_{\geq 2(2\mathcal{K}-1)}$ be the ground-state ICM at scale r. Then, for any j*

$$|A^{(r)}_{ss;\,0,j}| \leq 4m_0\kappa^{(r)}2^{-|j|/\xi^{(r)}}, \quad \xi^{(r)} := (2\mathcal{K}-1)2^{r+1}/m_0, \tag{D34}$$

where $\kappa^{(r)} > 1$ is the spectral condition number of $\mathbf{K}^{(r)}_{ss}$, Eq. (23).

Proof. We employ the Benzi-Golub theorem [66, Theorem 2.2] to bound off-diagonal entries of ICM $\mathbf{A}^{(r)}_{ss}$; see [67, p. 243] for an alternative statement of this theorem. Let $\alpha > 1$ and $\beta > 0$, with $\alpha > \beta$, be the half axes of an ellipse in the complex plane with foci in ± 1. This ellipse is specified by the sum of its half axes $\chi := \alpha + \beta > 1$, so we denote the ellipse by $\mathcal{E}_\chi$. Let $f(z)$ be an analytic function in the interior of $\mathcal{E}_\chi$ and continuous on $\mathcal{E}_\chi$ for any $1 < \chi < \bar{\chi}$, and let

$$M(\chi) := \max_{z\in\mathcal{E}_\chi} |f(z)| \quad \text{and} \quad K := \frac{2\chi M(\chi)}{\chi - 1}. \tag{D35}$$

Also let $\mathbf{B}$ be a symmetric and banded matrix whose spectrum spec$\mathbf{B}$ is contained in $[-1, 1]$ and let $\mathcal{B}$ the upper bandwidth of $\mathbf{B}$. Then by the Benzi-Golub theorem [66, Theorem 2.2]

$$|[f(\mathbf{B})]_{i,j}| \leq K2^{-\gamma|i-j|}, \quad \gamma := \frac{\log_2 \chi}{\mathcal{B}+1}. \tag{D36}$$

The ICM matrix $\mathbf{A}^{(r)}_{ss}$ is the principal square root of the coupling matrix $\mathbf{K}^{(r)}_{ss}$, Eq. (23), and $\mathbf{K}^{(r)}_{ss}$ is a symmetric and banded matrix with the upper bandwidth $2\mathcal{K}-2$. However, spec$\mathbf{K}^{(r)}_{ss}$ is not contained in $[-1, 1]$, so we cannot directly apply the Benzi-Golub theorem to this matrix. Let $[a, b] \subset \mathbb{R}^+$ be the interval containing spec$\mathbf{K}^{(r)}_{ss}$. To apply the Benzi-Golub theorem, by shifting and scaling the coupling matrix, we construct the matrix

$$\mathbf{B} := \frac{2\mathbf{K}^{(r)}_{ss} - (b+a)\mathbb{1}}{b-a}, \tag{D37}$$

whose spectrum is contained in $[-1, 1]$. Then the function

$$f(z) := \sqrt{\frac{b-a}{2}z + \frac{b+a}{2}}, \tag{D38}$$

maps $\mathbf{B}$ to the ICM; that is $\mathbf{A}^{(r)}_{ss} = f(\mathbf{B})$. By the Benzi-Golub theorem, we then have

$$|A^{(r)}_{ss;\,0,j}| \leq K2^{-\gamma|j|}, \tag{D39}$$

with

$$\gamma = \frac{\log_2 \chi}{2\mathcal{K}-1}. \tag{D40}$$

We now obtain a lower bound for γ. The function $f(z)$ in Eq. (D38) is analytic in the interior of any ellipse $\mathcal{E}_\chi$ with χ less than

$$\bar{\chi} = \frac{b+a}{b-a} + \sqrt{\left(\frac{b+a}{b-a}\right)^2 - 1}. \tag{D41}$$

Using the spectral condition number $\kappa^{(r)} := b/a$ of $\mathbf{K}^{(r)}_{ss}$, we have

$$\bar{\chi} = \frac{\sqrt{\kappa^{(r)}}+1}{\sqrt{\kappa^{(r)}}-1} = 1 + \frac{2}{\sqrt{\kappa^{(r)}}-1} \geq 1 + \frac{2}{\sqrt{\kappa^{(r)}-1}}, \tag{D42}$$

where the inequality holds because $\kappa^{(r)} \geq 1$. Setting

$$\chi = 1 + 2/\sqrt{\kappa^{(r)}-1} \leq \bar{\chi}, \tag{D43}$$

we obtain

$$\log_2(\chi) = \log_2\left(1 + 2/\sqrt{\kappa^{(r)}-1}\right) \geq \sqrt{2/\kappa^{(r)}} \geq m_0/2^{r+1}, \tag{D44}$$

where the first inequality holds because $\kappa^{(r)} \geq 1$ and the second inequality holds because

$$\kappa^{(r)} \leq 1 + 4^{r+2}/m_0^2 \leq 2 \times 4^{r+2}/m_0^2, \tag{D45}$$

as per Proposition 16. The combination of Eq. (D40) and Eq. (D44) yields

$$\gamma \geq 1/\xi^{(r)}, \tag{D46}$$

where $\xi^{(r)}$ is given in Eq. (D34). We now find an upper bound for K in Eq. (D39). By Eq. (D43) and $\kappa^{(r)} \geq 1$

$$\frac{2\chi}{\chi-1} = \chi\sqrt{\kappa^{(r)}-1} = \sqrt{\kappa^{(r)}-1} + 2 \leq 2\sqrt{2\kappa^{(r)}}. \tag{D47}$$

The function $f(z)$ in Eq. (D38) attains its maximum at the point $z_0 := \alpha$, where α is the greater half axis of the ellipse $\mathcal{E}_\chi$. The combination of $\chi = \alpha + \beta$, $\alpha^2 - \beta^2 = 1$

and Eq. (D43) yields

$$z_0 := \frac{\chi^2+1}{2\chi} = 1 + \frac{(\chi-1)^2}{2\chi} \le 1 + \frac{1}{2}(\chi-1)^2 = 1 + \frac{2}{\kappa^{(r)}-1}, \tag{D48}$$

and

$$M(\chi) = \max_{z\in\mathcal{E}_\chi} |f(z)| = f(z_0) = \sqrt{\frac{b-a}{2}\frac{\chi^2+1}{2\chi} + \frac{b+a}{2}} \le \sqrt{a/2}\sqrt{2\kappa^{(r)}+2} \le \sqrt{2a\kappa^{(r)}} = m_0\sqrt{2\kappa^{(r)}}, \tag{D49}$$

where the last equality follows because the smallest eigenvalue of $\mathbf{K}_{ss}^{(r)}$ is $a = m_0^2$. Equations (D35), (D47), and (D49) yield

$$K \le 4m_0\kappa^{(r)}. \tag{D50}$$

Proposition 17 follows from Eqs. (D39), (D46), and (D50). ■

1. Proof of Proposition 8

To prove Proposition 8, first we show that the ε_{th}-approximate ICM $\mathbf{A}_{\varepsilon_{\text{th}}}$, Eq. (50), is a positive-definite matrix. Then we prove that the infidelity between $|G_N(\mathbf{A})\rangle$ and $|G_N(\mathbf{A}_{\varepsilon_{\text{th}}})\rangle$ is no greater than ε.

We use the Bauer-Fike theorem [43, p. 405] to show that $\mathbf{A}_{\varepsilon_{\text{th}}}$ is a positive-definite matrix. Let $\mathbf{Q}$ be a nonsingular matrix such that $\mathbf{Q}^{-1}\mathbf{A}\mathbf{Q} = \mathbf{\Lambda}$ is a diagonal matrix and let $\mathbf{E} := \mathbf{A} - \mathbf{A}_{\varepsilon_{\text{th}}}$. According to the Bauer-Fike theorem, for any eigenvalue $\lambda(\mathbf{A}_{\varepsilon_{\text{th}}})$ of $\mathbf{A}_{\varepsilon_{\text{th}}}$, there is an eigenvalue $\lambda(\mathbf{A})$ of $\mathbf{A}$ such that

$$|\lambda(\mathbf{A}_{\varepsilon_{\text{th}}}) - \lambda(\mathbf{A})| \le \kappa_p(\mathbf{Q})\|\mathbf{E}\|_p\,, \tag{D51}$$

where $\|\bullet\|_p$ is the matrix norm induced by any p-norm on $\mathbb{C}^N$ and $\kappa_p(\bullet)$ is the condition number of a matrix with respect to this norm. As $\mathbf{A}$ is a real-symmetric matrix, $\mathbf{Q}$ can be chosen to be an orthogonal matrix for which $\|\mathbf{Q}\|_2 = 1$ and $\kappa_2(\mathbf{Q}) = 1$. Therefore, using Eq. (D51) with $p = 2$, we obtain

$$|\lambda_{\min}(\mathbf{A}_{\varepsilon_{\text{th}}}) - \lambda(\mathbf{A})| \le \|\mathbf{E}\|_2 \le N\varepsilon_{\text{th}}, \tag{D52}$$

for the smallest eigenvalue $\lambda_{\min}(\boldsymbol{A}_{\varepsilon_{\text{th}}})$ of $\mathbf{A}_{\varepsilon_{\text{th}}}$. The last inequality here comes from $|E_{ij}| \le \varepsilon_{\text{th}}$ by the definition of $\mathbf{E}$ and using the matrix-norm inequalities $\|\mathbf{E}\|_2 \le \|\mathbf{E}\|_F \le N\max_{i,j}|E_{ij}|$, where $\|\bullet\|_F$ is the Frobenius norm of a matrix. If $\lambda_{\min}(\mathbf{A}_{\varepsilon_{\text{th}}}) \ge \lambda_{\min}(\mathbf{A})$, then $\mathbf{A}_{\varepsilon_{\text{th}}}$ is already a positive-definite matrix. Otherwise, using Eq. (D52)

$$|\lambda_{\min}(\mathbf{A}_{\varepsilon_{\text{th}}}) - \lambda_{\min}(\mathbf{A})| \le N\varepsilon_{\text{th}} \le \varepsilon\gamma/\sqrt{N}, \tag{D53}$$

which implies that the $\mathbf{A}_{\varepsilon_{\text{th}}}$ is a positive-definite matrix.

We now prove Eq. (52). Let $\mathbf{T} := \mathbf{E}\mathbf{A}^{-1}$, then

$$\begin{aligned}\langle G_N(\mathbf{A})|G_N(\mathbf{A}_{\varepsilon_{\text{th}}})\rangle &= \left(\frac{\det\mathbf{A}\det\mathbf{A}_{\varepsilon_{\text{th}}}}{(2\pi)^{2N}}\right)^{1/4}\int_{\mathbb{R}^N} d^N\mathbf{x}e^{-\frac{1}{4}\mathbf{x}^{\mathsf{T}}(\mathbf{A}+\mathbf{A}_{\varepsilon_{\text{th}}})\mathbf{x}} \\ &= \frac{[\det(\mathbb{1}-\mathbf{T})]^{1/4}}{[\det(\mathbb{1}-\mathbf{T}/\mathbf{2})]^{1/2}},\end{aligned} \tag{D54}$$

where we use $\det\mathbf{A}_{\varepsilon_{\text{th}}} = \det\mathbf{A}\det(\mathbb{1}-\mathbf{T})$ by employing the multiplicative property of the determinant function [43, p. 11], and the Gaussian integral

$$\int_{\mathbb{R}^N} d^N\mathbf{x}e^{-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x}} = \left(\frac{(2\pi)^N}{\det\mathbf{A}}\right)^{1/2}, \tag{D55}$$

for a symmetric and positive-definite matrix $\mathbf{A}\in\mathbb{R}^{N\times N}$. We now use Proposition 13 to obtain a lower and an upper bound for determinant of the near-identity matrix $\mathbb{1}-\mathbf{T}$. Let us first find an upper bound for $|T_{ij}|$. Note that

$$\begin{aligned}|T_{ij}| &= |\sum_{k=0}^{N-1} E_{ik}\left(\mathbf{A}^{-1}\right)_{kj}| \le \varepsilon_{\text{th}}\max_j\sum_{k=0}^{N-1}|\left(\mathbf{A}^{-1}\right)_{kj}| \\ &= \varepsilon_{\text{th}}\|\mathbf{A}^{-1}\|_1 \le \varepsilon_{\text{th}}\gamma^{-1}\sqrt{N},\end{aligned} \tag{D56}$$

where the last inequality follows from the matrix-norm relations

$$\|\mathbf{A}\|_1 \le \sqrt{N}\|\mathbf{A}\|_2, \quad \|\mathbf{A}\|_2 = \lambda_{\max}(\mathbf{A}), \tag{D57}$$

for a real-symmetric matrix $\mathbf{A}$ [43, pp. 346, 363], and $\lambda_{\max}(\mathbf{A}^{-1}) = \lambda_{\min}^{-1}(\mathbf{A}) \le \gamma^{-1}$; note that $\gamma > 0$ is a lower bound for the eigenvalues of $\mathbf{A}$. By Proposition 13 and Eq. (D56), if $\varepsilon_{\text{th}}\gamma^{-1}N^{3/2} \le \varepsilon$, then

$$\det(\mathbb{1}-\mathbf{T}) \ge 1-\varepsilon, \quad \det(\mathbb{1}-\mathbf{T}/\mathbf{2}) \le (1-\varepsilon/2)^{-1}. \tag{D58}$$

Combination of Eq. (D54) and Eq. (D58) yields

$$\langle G_N(\mathbf{A})|G_N(\mathbf{A}_{\varepsilon_{\text{th}}})\rangle \ge (1-\varepsilon)^{1/4}(1-\varepsilon/2)^{1/2} \ge 1-\varepsilon, \tag{D59}$$

where, in the last inequality, we use $(1-\varepsilon/2)^{1/2} \ge (1-\varepsilon)^{1/2}$ and $(1-\varepsilon)^{3/4} \ge 1-\varepsilon$ for any $\varepsilon\in(0,1)$. Therefore, the infidelity between $|G_N(\mathbf{A})\rangle$ and $|G_N(\mathbf{A}_{\varepsilon_{\text{th}}})\rangle$ is at most ε.

2. Proof of Proposition 9

This Appendix presents a proof for Proposition 9. Following Eq. (59) with $r = c$, we have

$$\mathbf{A}_{ww}^{(r,r)} = \mathbf{G}\mathbf{A}_{ss}^{(r+1)}\mathbf{G}^{\mathsf{T}}, \tag{D60}$$

where $\mathbf{G}$ is the lower half of the wavelet-transform matrix at scale r, see Eq. (A7), with entries $G_{nm} = g_{n-2m}$. Using

Eq. (D60) and circulant property of $\mathbf{A}_{ss}^{(r+1)}$, we have

$$A_{ww;\,0,j}^{(r,r)} = \sum_{m,n=0}^{2\mathcal{K}-1} g_n g_m A_{ss;\,0,(2j+m-n)}^{(r+1)}. \tag{D61}$$

By virtue of Proposition 17, $|A_{ss;\,0,j}^{(r+1)}|$ decreases as j increases. For any m, n, and $j \geq 2\mathcal{K}-1$, we have $2j + m - n \geq j$, so

$$|A_{ss;\,0,(2j+m-n)}^{(r+1)}| \leq |A_{ss;\,0,j}^{(r+1)}| \quad \forall j \geq 2\mathcal{K}-1. \tag{D62}$$

Eq. (D61) and Eq. (D62) yield

$$|A_{ww;\,0,j}^{(r,r)}| \leq |A_{ss;\,0,j}^{(r+1)}| \sum_{m,n=0}^{2\mathcal{K}-1} |g_n g_m|, \tag{D63}$$

and because $\sum g_n^2 = 1$, we have $|g_n| \leq 1$. Therefore,

$$\sum_{m,n=0}^{2\mathcal{K}-1} |g_n g_m| \leq \left(\sum_{m=0}^{2\mathcal{K}-1} |g_m|\right)\left(\sum_{n=0}^{2\mathcal{K}-1} |g_n|\right) < 4\mathcal{K}^2. \tag{D64}$$

Proposition 9 follows from Eqs. (D34), (D63), and (D64).

3. Proof of Corollary 10

Here we prove Corollary 10. By virtue of Proposition 9, the off-diagonal entries in diagonal blocks of the multiscale ICM decay slower as the scale index r increases. Therefore, for a given threshold value ε_{th}, the bottom-right block of the approximate ICM Eq. (50) has the largest bandwidth among the diagonal blocks. Plugging $r = k-1$ into Eq. (53) we obtain

$$|A_{ww;\,0,j}^{(k-1,k-1)}| \leq \varepsilon_{\text{th}}, \tag{D65}$$

for

$$|j| \geq \xi^{(k)} \log_2\left(\frac{16\mathcal{K} m_0 \kappa^{(k)}}{\varepsilon_{\text{th}}}\right). \tag{D66}$$

We now use Proposition 8 and $2^{k+1} \leq \log N$ to express the upper bandwidth in terms of $\mathcal{K}, m_0, \varepsilon_{\text{vac}}$, and N. By Eq. (53)

$$\xi^{(k)} = (2\mathcal{K}-1)2^{k+1}/m_0 \leq (2\mathcal{K}-1)(\log N)/m_0. \tag{D67}$$

The smallest eigenvalue of the ICM is m_0, so by Proposition 8 we take $\varepsilon_{\text{th}} = m_0 \varepsilon_{\text{vac}} N^{-3/2}$. Therefore,

$$\begin{aligned}\frac{16\mathcal{K}^2 m_0 \kappa^{(k)}}{\varepsilon_{\text{th}}} &= \frac{16\mathcal{K}^2 N^{3/2} \kappa^{(k)}}{\varepsilon_{\text{vac}}} \leq \frac{32\mathcal{K}^2 N^{3/2} 4^{k+2}}{m_0^2 \varepsilon_{\text{vac}}} \\ &\leq \frac{32\mathcal{K}^2 N^{3/2} \log^2 N}{m_0^2 \varepsilon_{\text{vac}}} \leq \frac{16\mathcal{K}^2 N^2}{m_0^2 \varepsilon_{\text{vac}}^2},\end{aligned} \tag{D68}$$

where we use Eq. (D45) in the first inequality. In the last two inequalities we use

$$2^{k+1} \leq \log N, \quad \varepsilon_{\text{vac}} \geq \varepsilon_{\text{vac}}^2, \quad \text{and} \quad 32N^{3/2}\log^2 N \leq 16N^2, \tag{D69}$$

for $\varepsilon_{\text{vac}} \in (0,1)$ and $N \in \mathbb{Z}^+$. Equation (54) follows from Eqs. (D66)–(D68).

APPENDIX E: COMPUTING ROTATION ANGLE FOR 1DG-STATE GENERATION

In this Appendix, we analyze the time complexity for computing the rotation angle θ_ℓ in Eq. (71) on a quantum computer. We first simplify computing θ_ℓ using the double-angle identity as

$$\theta_\ell = \frac{1}{2}\arccos(u_\ell) = \frac{\pi}{4} - \frac{1}{4}\arcsin(u_\ell), \tag{E1}$$

where

$$u_\ell := 2\frac{f(\tilde{\sigma}_\ell/2,\ \mu_\ell/2,\ m_\ell - 1)}{f(\tilde{\sigma}_\ell,\ \mu_\ell,\ m_\ell)} - 1, \tag{E2}$$

and $f(\tilde{\sigma}_\ell,\ \mu_\ell,\ m_\ell)$ is defined in Eq. (69). Using Eq. (E1), computing θ_ℓ requires computing u_ℓ and $\arcsin(u_\ell)$, and performing one multiplication and one addition. Therefore, time complexity for computing the rotation angle θ_ℓ, denoted by T_{ANGLE}, is

$$T_{\text{ANGLE}} = T_u + T_{\arcsin} + 2, \tag{E3}$$

where T_u and $T_{\arcsin}$ are time complexities for computing u_ℓ and $\arcsin(u_\ell)$, respectively. By Eq. (E2), computing u_ℓ requires computing $f(\tilde{\sigma}_\ell/2,\ \mu_\ell/2,\ m_\ell - 1)$ and $f(\tilde{\sigma}_\ell,\ \mu_\ell,\ m_\ell)$, and performing one division, one multiplication, and one addition. Therefore,

$$T_u = 2T_f + 3, \tag{E4}$$

where T_f is time complexity for computing $f(\tilde{\sigma}_\ell,\ \mu_\ell,\ m_\ell)$. By Eq. (E3) and Eq. (E4), we need only T_f and $T_{\arcsin}$ to obtain T_{ANGLE}. First we provide a high-level description of how to obtain T_f and $T_{\arcsin}$. We show, in Proposition 18, that computing $f(\tilde{\sigma}_\ell,\ \mu_\ell,\ m_\ell)$ to t bit of precision requires computing at most $4t-1$ exponentials and adding them all. The argument of each exponential needs performing one addition, one multiplication, one division and calculating the square of two numbers. Calculating the square of a number can be performed by one multiplication [53, p. 7]. Altogether, time complexity for computing $f(\tilde{\sigma}_\ell,\ \mu_\ell,\ m_\ell)$ is at most $(4t-1)T_{\exp} + (4t-2) + 5$, and therefore

$$T_f \in \mathcal{O}(tT_{\exp}), \tag{E5}$$

where $T_{\exp}$ is time complexity for computing an exponential. We use Proposition 19 to show $T_{\exp}$ scales linearly

with t. By this proposition, to compute e^{-x} with t bits of precision, first the polynomial

$$P(x) := a_0 + a_1 x + a_2 x^2, \tag{E6}$$

with $a_0 = 1, a_1 = 1/4^t$ and $a_2 = 1/4^{2t}$ is evaluated at x and then the result is iteratively squared $2t$ times. Computing square of a number needs one multiplication, so $T_{\exp} = T_P + 2t$, where T_P is time complexity for computing $P(x)$ in Eq. (E6).

We use Horner's method to compute a polynomial on a quantum computer [53], and to obtain T_P. To evaluate the polynomial in Eq. (E6) at x, we store the coefficients a_0, a_1, and a_2 on a ancillary quantum register and implement the operation $a_2 x + a_1 \mapsto (a_2 x + a_1)x + a_0$. Implementing this operation needs two multiplications and two additions before uncomputing the intermediate results, so T_P is a constant and

$$T_{\exp} \in \Theta(t). \tag{E7}$$

We use Proposition 20 and Proposition 21 to obtain $T_{\arcsin}$. In Proposition 20, we show that the argument x of $\arcsin(x)$ is a positive number less than $1/2$ and, in Proposition 21, we show a polynomial of degree t suffices to approximate $\arcsin(x)$ with t bits of precision. As described, computing a polynomial of degree t by Horner's method requires $\Theta(t)$ multiplications and additions, so

$$T_{\arcsin} = \Theta(t), \tag{E8}$$

The combination of Eqs. (E3)–(E5) and (E7)–(E8) yields

$$T_{\text{ANGLE}} \in \mathcal{O}(t^2). \tag{E9}$$

We now state and prove the propositions that we use to obtain T_{ANGLE}.

Proposition 18: *Let $m \in \mathbb{Z}^+, t \in \mathbb{Z}^+$, $\ell \in \{0, \ldots, m-1\}$, $\mu_\ell \in [0, 1)$, $m_\ell := m - \ell$, $\tilde{\sigma}_\ell \in (0, 2^{m_\ell/2})$ and*

$$f(\tilde{\sigma}_\ell, \mu_\ell, m_\ell) := \sum_{j=-2^{m_\ell-1}}^{2^{m_\ell-1}-1} \exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right), \tag{E10}$$

$$g(\tilde{\sigma}_\ell, \mu_\ell) := \sum_{j=-2t-1}^{2t} \exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right). \tag{E11}$$

For $2^{m_\ell} > 8(t+3)$, if $\tilde{\sigma}_\ell^2 > t$ then

$$|f(\tilde{\sigma}_\ell, \mu_\ell, m_\ell) - \tilde{\sigma}_\ell\sqrt{2\pi}| \le \frac{1}{2^{t+1}}, \tag{E12}$$

else

$$|f(\tilde{\sigma}_\ell, \mu_\ell, m_\ell) - g(\tilde{\sigma}, \mu_\ell)| \le \frac{1}{2^{t+1}}. \tag{E13}$$

Proof. We first prove the first part of Proposition 18. Using the triangle inequality,

$$\begin{aligned}
&\left|f(\tilde{\sigma}_\ell, \mu_\ell, m_\ell) - \sigma_\ell\sqrt{2\pi}\right| \\
&\quad\le \left|f(\tilde{\sigma}_\ell, \mu_\ell, m_\ell) - \sum_{j\in\mathbb{Z}} \exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right)\right| \\
&\qquad + \left|\sum_{j\in\mathbb{Z}} \exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right) - \tilde{\sigma}_\ell\sqrt{2\pi}\right|.
\end{aligned} \tag{E14}$$

Let us now find and upper bound for the first term in the right-hand side of Eq. (E14). By Eq. (E10),

$$\begin{aligned}
&\left|f(\tilde{\sigma}_\ell, \mu_\ell, m_\ell) - \sum_{j\in\mathbb{Z}} \exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right)\right| \\
&= \sum_{2^{m_\ell-1}}^{\infty} \left(\exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right) + \exp\left(-\tfrac{(j+1-\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right)\right) \\
&\le 2\sum_{2^{m_\ell-1}}^{\infty} \exp\left(-\tfrac{j^2}{2\tilde{\sigma}_\ell^2}\right) \le 2\sum_{2^{m_\ell-1}}^{\infty} e^{-j/4} = \frac{2e^{-2^{m_\ell-3}}}{1-e^{-1/4}} \\
&\le \frac{1}{2}e^{-t}.
\end{aligned} \tag{E15}$$

The first inequality here holds because $\mu_\ell \in [0, 1)$. The second inequality is obtained using

$$\sum_{2^{m_\ell-1}}^{\infty} \exp\left(-\tfrac{j^2}{2\tilde{\sigma}_\ell^2}\right) \le \sum_{2^{m_\ell-1}}^{\infty} \exp\left(-\tfrac{2^{m_\ell-1}}{2\tilde{\sigma}_\ell^2} j\right), \tag{E16}$$

and $\tilde{\sigma}_\ell^2 \le 2^{m_\ell}$. The last inequality is obtained using $2^{m_\ell} > 8(t+3)$ and $2e^{-3}/(1-e^{-1/4}) < 1/2$.

We now establish an upper bound for the second term in the right-hand side of Eq. (E14). By Poisson's summation formula [64, p. 385],

$$\begin{aligned}
&\sum_{j\in\mathbb{Z}} \exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right) \\
&\quad= \tilde{\sigma}_\ell\sqrt{2\pi} \sum_{j\in\mathbb{Z}} e^{-2\pi^2\tilde{\sigma}_\ell^2 j^2 - 2\pi i \mu_\ell j} \\
&\quad= \tilde{\sigma}_\ell\sqrt{2\pi} \left(1 + 2\sum_{j=1}^{\infty} e^{-2\pi^2\tilde{\sigma}_\ell^2 j^2} \cos(2\pi\mu_\ell j)\right).
\end{aligned} \tag{E17}$$

Therefore,

$$\begin{aligned}
&\left|\sum_{j\in\mathbb{Z}}\exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right)-\tilde{\sigma}_\ell\sqrt{2\pi}\right| \\
&\quad=2\sqrt{2\pi}\tilde{\sigma}_\ell\left|\sum_{j=1}^{\infty}e^{-2\pi^2\tilde{\sigma}_\ell^2 j^2}\cos(2\pi\mu_\ell j)\right| \\
&\quad\le 2\sqrt{2\pi}\tilde{\sigma}_\ell\sum_{j=1}^{\infty}e^{-2\pi^2\tilde{\sigma}_\ell^2 j^2} \\
&\quad\le 2\sqrt{2\pi}\tilde{\sigma}_\ell\sum_{j=1}^{\infty}e^{-2\pi^2\tilde{\sigma}_\ell^2 j}=2\sqrt{2\pi}\tilde{\sigma}_\ell\frac{e^{-2\pi^2\tilde{\sigma}_\ell^2}}{1-e^{-2\pi^2\tilde{\sigma}_\ell^2}} \\
&\quad\le e^{-\pi^2\tilde{\sigma}_\ell^2}\le e^{-\pi^2 t}. \qquad \text{(E18)}
\end{aligned}$$

Here we use the triangle inequality and $|\cos(2\pi\mu_\ell j)|\le 1$ to obtain the first inequality. The second inequality is obtained by

$$\sum_{j=1}^{\infty}e^{-2\pi^2\tilde{\sigma}_\ell^2 j^2}\le\sum_{j=1}^{\infty}e^{-2\pi^2\tilde{\sigma}_\ell^2 j}. \qquad \text{(E19)}$$

The third inequality is obtained using $1-e^{-2\pi^2\tilde{\sigma}_\ell^2}\ge e^{-\tilde{\sigma}_\ell^2}$ for $\tilde{\sigma}_\ell^2\ge t\ge 1$, and $2\sqrt{2\pi}xe^{-(\pi^2-1)x^2}\le 1$ for any $x\in\mathbb{R}^+$. The last inequality holds because $\tilde{\sigma}_\ell^2\ge t$. The first part of Proposition 18 follows from Eqs. (E14), (E15), (E18), and

$$\frac{1}{2}e^{-t}+e^{-\pi^2 t}\le\frac{1}{2^{t+1}}\quad\forall t\in\mathbb{Z}^+. \qquad \text{(E20)}$$

We now prove the second part of Proposition 18. By Eqs. (E10) and (E11),

$$\begin{aligned}
&|f(\tilde{\sigma}_\ell,\mu_\ell,m_\ell)-g(\tilde{\sigma},\mu_\ell)| \\
&\quad=\sum_{j=2t+1}^{2^{m_\ell-1}-1}\left(\exp\left(-\tfrac{(j+\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right)+\exp\left(-\tfrac{(j+1-\mu_\ell)^2}{2\tilde{\sigma}_\ell^2}\right)\right) \\
&\quad\le 2\sum_{j=2t+1}^{2^{m_\ell-1}-1}\exp\left(-\tfrac{j^2}{2\tilde{\sigma}_\ell^2}\right)\le 2\sum_{j=2t+1}^{\infty}e^{-j}=2\frac{e^{-(2t+1)}}{1-e^{-1}} \\
&\quad\le\frac{1}{2^{t+1}}. \qquad \text{(E21)}
\end{aligned}$$

The first inequality here is obtained using $\mu_\ell\in[0,1)$; the second inequality is obtained using

$$\sum_{j=2t+1}^{2^{m_\ell-1}-1}\exp\left(-\tfrac{j^2}{2\tilde{\sigma}_\ell^2}\right)\le\sum_{j=2t+1}^{\infty}\exp\left(-\tfrac{2t+1}{2\tilde{\sigma}_\ell^2}j\right), \qquad \text{(E22)}$$

and $\tilde{\sigma}_\ell^2\le t$. The last inequality is valid for any $t\in\mathbb{Z}^+$. ■

Proposition 19: *Let $t\in\mathbb{Z}^+, x\in[0,t]$ and $P(x/\ell):=1-x/\ell+x^2/\ell^2$. Then, for any $\ell\ge 4^t$,*

$$|e^{-x}-P^\ell(x/\ell)|\le 1/4^t. \qquad \text{(E23)}$$

Proof. Note that

$$\begin{aligned}
e^{-x}&=\left(e^{-x/\ell}\right)^\ell=\left[P(x/\ell)+E(x/\ell)\right]^\ell, \\
E(x/\ell)&:=\sum_{n\ge 3}\frac{(-)^n}{n!}(x/\ell)^n. \qquad \text{(E24)}
\end{aligned}$$

Using binomial expansion

$$e^{-x}=P^\ell(x/\ell)+\sum_{n=1}^{\ell}\binom{\ell}{n}P^{\ell-n}(x/\ell)E^n(x/\ell). \qquad \text{(E25)}$$

Therefore,

$$\begin{aligned}
|e^{-x}-P^\ell(x/\ell)|&\le\sum_{n=1}^{\ell}\binom{\ell}{n}|P(x/\ell)|^{\ell-n}|E(x/\ell)|^n \\
&\le\sum_{n=1}^{\ell}\binom{\ell}{n}|P(x/\ell)|^{\ell-n}|E(x/\ell)|^n, \qquad \text{(E26)}
\end{aligned}$$

where, in the last inequality, we use $P(x/\ell)\le 1$ for $x/\ell\le 1$. Let us now find an upper bound for $|E(x/\ell)|$. Using Eq. (E24) and the triangle inequality,

$$\begin{aligned}
|E(x/\ell)|&\le\sum_{n\ge 3}\frac{1}{n!}(x/\ell)^n=(x/\ell)^3\sum_{n\ge 0}\frac{1}{(n+3)!}(x/\ell)^n \\
&\le\frac{1}{6}(x/\ell)^3e^{x/\ell}\le\frac{1}{2}(x/\ell)^3, \qquad \text{(E27)}
\end{aligned}$$

the last inequality holds because $e^{x/\ell}/3\le 1$ for $x/\ell\le 1$. Using Eqs. (E26) and (E27)

$$\begin{aligned}
|e^{-x}-P^\ell(x/\ell)|&\le\sum_{n=1}^{\ell}\binom{\ell}{n}\frac{1}{2^n}(x/\ell)^{3n}\le\sum_{n=1}^{\ell}\frac{\ell^n}{2^n}(x/\ell)^{3n} \\
&=\frac{x^3}{2\ell^2}+\sum_{n=2}^{\ell}\frac{x^{3n}}{2^n\ell^{2n}}\le\frac{x^3}{2\ell^2}\left(1+\frac{x^3}{2\ell^2}\right)\le\frac{1}{\ell}\le\frac{1}{4^t}. \qquad \text{(E28)}
\end{aligned}$$

Here we use $x^3/\ell\le 1/2$ for $x\in[0,t]$ and $\ell>4^t$. ■

Proposition 20: *Let* $\tilde{\sigma} \in [1,\infty), \mu \in [0,1), m \in \mathbb{Z}^+$ *and*

$$f(\tilde{\sigma},\mu,m) = \sum_{j=-2^{m-1}}^{2^{m-1}-1} e^{-(j+\mu)^2/2\tilde{\sigma}^2}. \tag{E29}$$

Then

$$\left| 2\frac{f(\tilde{\sigma}/2,\mu/2,m-1)}{f(\tilde{\sigma},\mu,m)} - 1 \right| \le 1/2. \tag{E30}$$

Proof. Let $f_{\rm even} := f(\tilde{\sigma}/2,\mu/2,m-1)$ and $f_{\rm odd} = f(\tilde{\sigma}/2,(\mu+1)/2,m-1)$. Then $f(\tilde{\sigma},\mu,m) = f_{\rm even} + f_{\rm odd}$, and Eq. (E30) becomes

$$\left| \frac{f_{\rm even} - f_{\rm odd}}{f_{\rm even} + f_{\rm odd}} \right| \le 1/2. \tag{E31}$$

We show $3f_{\rm even} \ge f_{\rm odd}$ and $3f_{\rm odd} \ge f_{\rm even}$. These two inequalities yield Eq. (E31), which proves Proposition 20. Notice that

$$\begin{aligned} f_{\rm even} &= \sum_{j=0}^{2^{m-2}-1} e^{-(2j+\mu)^2/2\tilde{\sigma}^2} + \sum_{j=1}^{2^{m-2}} e^{-(2j-\mu)^2/2\tilde{\sigma}^2} \\ &= \sum_{j=0}^{2^{m-2}-1} e^{-(2j+\mu)^2/2\tilde{\sigma}^2} + \sum_{j=2}^{2^{m-2}+1} e^{-(2j-2-\mu)^2/2\tilde{\sigma}^2}, \end{aligned} \tag{E32}$$

$$\begin{aligned} f_{\rm odd} &= \sum_{j=0}^{2^{m-2}-1} e^{-(2j+1+\mu)^2/2\tilde{\sigma}^2} + \sum_{j=1}^{2^{m-2}} e^{-(2j-1-\mu)^2/2\tilde{\sigma}^2} \\ &= \sum_{j=1}^{2^{m-2}} \left(e^{-(2j-1+\mu)^2/2\tilde{\sigma}^2} + e^{-(2j-1-\mu)^2/2\tilde{\sigma}^2} \right). \end{aligned} \tag{E33}$$

Using these equations

$$\begin{aligned} f_{\rm even} - f_{\rm odd} = &\sum_{j=0}^{2^{m-2}-1} \left(e^{-(2j+\mu)^2/2\tilde{\sigma}^2} - e^{-(2j+\mu+1)^2/2\tilde{\sigma}^2} \right) \\ &+ \sum_{j=2}^{2^{m-2}} \left(e^{-(2j-2-\mu)^2/2\tilde{\sigma}^2} - e^{-(2j-1-\mu)^2/2\tilde{\sigma}^2} \right) \\ &+ e^{-(2^{m-1}-\mu)^2/2\tilde{\sigma}^2} - e^{-(1-\mu)^2/2\tilde{\sigma}^2} \end{aligned} \tag{E34}$$

$$\begin{aligned} f_{\rm odd} - f_{\rm even} = &\sum_{j=1}^{2^{m-2}} \left(e^{-(2j-\mu-1)^2/2\tilde{\sigma}^2} - e^{-(2j-\mu)^2/2\tilde{\sigma}^2} \right) \\ &+ \sum_{j=1}^{2^{m-2}-1} \left(e^{-(2j-1+\mu)^2/2\tilde{\sigma}^2} - e^{-(2j+\mu)^2/2\tilde{\sigma}^2} \right) \\ &+ e^{-(2^{m-1}-1+\mu)^2/2\tilde{\sigma}^2} - e^{-\mu^2/2\tilde{\sigma}^2}. \end{aligned} \tag{E35}$$

The first three terms in the right-hand side of these equations are non-negative, so we have

$$f_{\rm even} - f_{\rm odd} \ge -e^{-(1-\mu)^2/2\tilde{\sigma}^2}, \quad f_{\rm odd} - f_{\rm even} \ge -e^{-\mu^2/2\tilde{\sigma}^2}. \tag{E36}$$

By these inequalities, Eq. (E32) and Eq. (E33)

$$\begin{aligned} 3f_{\rm even} - f_{\rm odd} &\ge 2f_{\rm even} - e^{-(1-\mu)^2/2\tilde{\sigma}^2} \\ &\ge 2\left(e^{-\mu^2/2\tilde{\sigma}^2} + e^{-(2-\mu)^2/2\tilde{\sigma}^2} \right) \\ &\quad - e^{-(1-\mu)^2/2\tilde{\sigma}^2} \ge 0, \end{aligned} \tag{E37}$$

$$\begin{aligned} 3f_{\rm odd} - f_{\rm even} &\ge 2f_{\rm odd} - e^{-\mu^2/2\tilde{\sigma}^2} \\ &\ge 2\left(e^{-(1+\mu)^2/2\tilde{\sigma}^2} + e^{-(1-\mu)^2/2\tilde{\sigma}^2} \right) \\ &\quad - e^{-\mu^2/2\tilde{\sigma}^2} \ge 0, \end{aligned} \tag{E38}$$

for all $\tilde{\sigma} \ge 1$ and $\mu \in [0,1)$. These inequalities yield $3f_{\rm even} \ge f_{\rm odd}$ and $3f_{\rm odd} \ge f_{\rm even}$, and hence Eq. (E31). ■

Proposition 21: *Let* $t \in \mathbb{Z}^+, |x| \le 1/2$ *and*

$$P(x) := \sum_{\ell=0}^{t-1} a_{2\ell+1} x^{2\ell+1}, \quad a_{2\ell+1} := \frac{(2\ell-1)!!}{(2\ell)!!}\frac{1}{2\ell+1}. \tag{E39}$$

Then $|\arcsin(x) - P(x)| \le 1/2^{2t+1}$.

Proof. Using the binomial series of $(1-x^2)^{-1/2}$,

$$\begin{aligned} \arcsin(x) &= \int_0^x \frac{dx}{\sqrt{1-x^2}} \\ &= P(x) + \int_0^x dx \sum_{\ell=t}^{\infty} \frac{(2\ell-1)!!}{(2\ell)!!} x^{2\ell}. \end{aligned} \tag{E40}$$

By this equation and $(2\ell-1)!!/(2\ell)!! < 1$ we have

$$\begin{aligned} |\arcsin(x) - P(x)| &< \int_0^x dx \sum_{\ell=t}^{\infty} x^{2\ell} = \int_0^x dx x^{2t} \sum_{\ell=0}^{\infty} x^{2\ell} \\ &\le \frac{4}{3}\frac{x^{2t+1}}{2t+1} \le \frac{1}{2^{2t+1}}, \end{aligned} \tag{E41}$$

where we use $|x| \le 1/2$ and $t \ge 1$ in the last two inequalities. ■

Input:
$N \in \mathbb{Z}^+$ ▷ order of a square matrix
$\mathbf{A} \in \mathbb{R}^{N\times N}$ ▷ a real-symmetric matrix of order N
Output:
$\mathbf{U} \in \mathbb{R}^{N\times N}$ ▷ upper unit-triangular matrix in UDU decomposition of $\mathbf{A}$
$\mathbf{d} \in \mathbb{R}^{N}$ ▷ diagonals of the diagonal matrix $\mathbf{D}$ in UDU decomposition of $\mathbf{A}$
function UDU($N, \mathbf{A}$)
  $\mathbf{U} \leftarrow \mathbb{1}$ ▷ initializes $\mathbf{U}$ as identity matrix
  for $i \leftarrow N-1$ to 0 **do** ▷ iterates over columns/rows of $\mathbf{A}$ from right/bottom to left/top
    $\mathbf{v}_{i,i+1:N-1} \leftarrow \mathbf{u}_{i,i+1:N-1} \odot \mathbf{d}_{i+1:N-1}$ ▷ $\odot$ denotes the Hadamard product
    $d_i \leftarrow a_{ii} - \mathbf{u}_{i,i+1:N-1} \cdot \mathbf{v}_{i,i+1:N-1}$
    if $i > 0$ **then**
      $\mathbf{u}_{0:i-1,i} \leftarrow \left[\mathbf{a}_{0:i-1,i} - \mathbf{u}_{0:i-1,i+1:N-1} \cdot \mathbf{v}_{i,i+1:N-1}\right]/d_i$
return $\{\mathbf{d}, \mathbf{U}\}$

Algorithm 12. Classical algorithm for UDU decomposition of a dense real-symmetric matrix

APPENDIX F: UDU DECOMPOSITION

In this Appendix, we present a classical algorithm for computing the UDU decomposition of a dense real-symmetric matrix. The algorithm presented here elucidates our classical UDU-decomposition algorithm for a matrix with a fingerlike sparse structure. First we describe the UDU matrix decomposition and then present the algorithm for a dense matrix as pseudocode.

In the UDU matrix decomposition, a symmetric matrix $\mathbf{A}$ is decomposed into the product of an upper unit-triangular matrix $\mathbf{U}$, a diagonal matrix $\mathbf{D}$ and transpose of the upper unit-triangular matrix. The UDU decomposition is closely related to the LDL decomposition, where a symmetric matrix is decomposed into the product of a lower unit-triangular matrix $\mathbf{L}$, a diagonal matrix, and transpose of the lower unit-triangular matrix. The LDL decomposition algorithm starts from the top-left corner of the matrix $\mathbf{L}$ and proceeds to compute entries of this matrix row by row [68]. The UDU-decomposition algorithm, however, starts from the top-right corner of $\mathbf{U}$ and proceeds to compute its entries column by column.

To elucidate the algorithm, we write the UDU decomposition of a real-symmetric matrix $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}} = \mathbf{U}\mathbf{V}^{\mathsf{T}}, \quad \mathbf{V} := \mathbf{U}\mathbf{D}, \tag{F1}$$

where $\mathbf{V}$ is an upper triangular matrix. By definition, the nonzero elements in the i^{th} row of $\mathbf{V}$ are

$$\mathbf{v}_{i,\,i:N-1} = \mathbf{u}_{i,\,i:N-1} \odot \mathbf{d}_{i:N-1}, \tag{F2}$$

where $\odot$ denotes the Hadamard product. By Eq. (F1), elements of the i^{th} column in the upper-triangular part of $\mathbf{A}$ are

$$\mathbf{A}_{0:i,\,i} = \mathbf{u}_{0:i,\,i:N-1} \cdot \mathbf{v}_{i,\,i:N-1}. \tag{F3}$$

This equation along with $u_{ii} = 1$ and $v_{ii} = d_i$ yield

$$d_i = a_{ii} - \mathbf{u}_{i,\,i+1:N-1} \cdot \mathbf{v}_{i,\,i+1:N-1}, \tag{F4}$$

$$\mathbf{u}_{0:i-1,\,i} = \left[\mathbf{A}_{0:i-1,i} - \mathbf{u}_{0:i-1,\,i+1:N-1} \cdot \mathbf{v}_{i,\,i+1:N-1}\right]/d_i \quad \forall\, i \neq 0, \tag{F5}$$

for the diagonal d_i and shear elements in the i^{th} column of $\mathbf{U}$, respectively.

The procedure of the UDU-decomposition algorithm is as follows. We start from the last column $i = N-1$ and proceed to the first column $i = 0$. For each index i, first we compute $\mathbf{v}_{i,\,i:N-1}$, i.e., the nonzero elements in the i^{th} row of $\mathbf{V}$ by Eq. (F3). Then we compute the i^{th} diagonal d_i of $\mathbf{D}$ by Eq. (F4) and the shear elements in the i^{th} column of $\mathbf{U}$ by Eq. (F5). The inputs, outputs, and explicit procedure of the algorithm is presented in Algorithm 12.

[1] S. P. Jordan, K. S. M. Lee, and J. Preskill, Quantum algorithms for quantum field theories, Science **336**, 1130 (2012).
[2] S. P. Jordan, K. S. M. Lee, and J. Preskill, Quantum computation of scattering in scalar quantum field theories, Quantum Inf. Comput. **14**, 1014 (2014).
[3] S. P. Jordan, K. S. M. Lee, and J. Preskill, Quantum algorithms for fermionic quantum field theories, (2014), arXiv:1404.7115.
[4] A. Hamed Moosavian and S. Jordan, Faster quantum algorithm to simulate fermionic quantum field theory, Phys. Rev. A **98**, 012332 (2018).
[5] A. Roggero, A. C. Y. Li, J. Carlson, R. Gupta, and G. N. Perdue, Quantum computing for neutrino-nucleus scattering, Phys. Rev. D **101**, 074038 (2020).
[6] E. F. Dumitrescu, A. J. McCaskey, G. Hagen, G. R. Jansen, T. D. Morris, T. Papenbrock, R. C. Pooser, D. J. Dean, and P. Lougovski, Cloud Quantum Computing of an Atomic Nucleus, Phys. Rev. Lett. **120**, 210501 (2018).

[7] S. Harmalkar, H. Lamm, and S. Lawrence, Quantum simulation of field theories without state preparation, (2020), arXiv:2001.11490.

[8] E. J. Gustafson and H. Lamm, Toward quantum simulations of $\mathbb{Z}_2$ gauge theory without state preparation, Phys. Rev. D **103**, 054507 (2021).

[9] K. Choi, D. Lee, J. Bonitati, Z. Qian, and J. Watkins, Rodeo Algorithm for Quantum Computing, Phys. Rev. Lett. **127**, 040505 (2021).

[10] D. Lee, J. Bonitati, G. Given, C. Hicks, N. Li, B.-N. Lu, A. Rai, A. Sarkar, and J. Watkins, Projected cooling algorithm for quantum computation, Phys. Lett. B **807**, 135536 (2020).

[11] E. J. Gustafson, Projective cooling for the transverse Ising model, Phys. Rev. D **101**, 071504 (2020).

[12] M. Blasone, P. Jizba, and G. Vitiello, *Quantum Field Theory and Its Macroscopic Manifestations* (Imperial College Press, London, 2011).

[13] C. Zalka, Simulating quantum systems on a quantum computer, Proc. Math. Phys. Eng. Sci. **454**, 313 (1998).

[14] L. Grover and T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, (2002), arXiv:quant-ph/0208112.

[15] Y. R. Sanders, G. H. Low, A. Scherer, and D. W. Berry, Black-Box Quantum State Preparation without Arithmetic, Phys. Rev. Lett. **122**, 020502 (2019).

[16] S. Lloyd, Universal quantum simulators, Science **273**, 1073 (1996).

[17] T. Byrnes and Y. Yamamoto, Simulating lattice gauge theories on a quantum computer, Phys. Rev. A **73**, 022328 (2006).

[18] G. K. Brennen, P. Rohde, B. C. Sanders, and S. Singh, Multiscale quantum simulation of quantum field theory using wavelets, Phys. Rev. A **92**, 032315 (2015).

[19] A. Kitaev and W. A. Webb, Wavefunction preparation and resampling using a quantum computer, (2009), arXiv:0801.0342v2.

[20] J. Alman and V. V. Williams, in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)* (SIAM, Alexandria, 2021), p. 522.

[21] R. Asaka, K. Sakai, and R. Yahagi, Quantum circuit for the fast Fourier transform, Quantum Inf. Process. **19**, 277 (2020).

[22] D. Bini and P. Favati, On a matrix algebra related to the discrete Hartley transform, SIAM J. Matrix Anal. Appl. **14**, 500 (1993).

[23] G. Beylkin, R. Coifman, and V. Rokhlin, Fast wavelet transforms and numerical algorithms I, Commun. Pure Appl. Math. **44**, 141 (1991).

[24] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, in *Quantum Computation and Information*, Contemporary Mathematics, Vol. 305 (American Mathematical Society, Washington DC, 2002).

[25] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum Random Access Memory, Phys. Rev. Lett. **100**, 160501 (2008).

[26] M. Bagherimehrab, *Algorithmic quantum-state generation for simulating quantum field theories on a quantum computer*, Ph.D. thesis, University of Calgary, Calgary, AB (2022), http://hdl.handle.net/1880/114333.

[27] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way* (Academic Press, Orlando, 2009), 3rd ed.

[28] I. Daubechies, *Ten Lectures on Wavelets* (Society for Industrial and Applied Mathematics, Philadelphia, 1992).

[29] F. Bulut and W. N. Polyzou, Wavelets in field theory, Phys. Rev. D **87**, 116011 (2013).

[30] T. Farrelly and J. Streich, Discretizing quantum field theories for quantum simulation, (2020), arXiv:2002.02643.

[31] H. J. Rothe, *Lattice Gauge Theories: An Introduction*, 3rd ed., World Scientific Lecture Notes in Physics, Vol. 74 (World Scientific, Singapore, 2005).

[32] N. Klco and M. J. Savage, Digitization of scalar fields for quantum computing, Phys. Rev. A **99**, 052335 (2019).

[33] S. Singh and G. K. Brennen, Holographic construction of quantum field theory using wavelets, (2016), arXiv:1606.05068.

[34] E. Knill, *Conventions for Quantum Pseudocode*, Technical Report LA-UR-96-2724 (Los Alamos National Lab, 1996).

[35] M. Lanzagorta and J. Uhlmann, *Quantum Computer Science*, Synthesis Lectures on Quantum Computing, Vol. 2 (Morgan & Claypool, 2008). p. 28.

[36] J. A. Miszczak, *High-level Structures for Quantum Computing*, Synthesis Lectures on Quantum Computing, Vol. 6 (Morgan & Claypool, 2012), p. 45.

[37] Q. Wang and M. Ying, Quantum random access stored-program machines, (2020), arXiv:2003.03514.

[38] M. T. Goodrich and R. Tamassia, *Algorithm Design and Applications* (Wiley, Hoboken, 2014), 1st ed, https://dl.acm.org/doi/10.5555/2755032.

[39] R. P. Brent, Multiple-precision zero-finding methods and the complexity of elementary function evaluation, (2010), arXiv:004.3412v2.

[40] A. Montanaro, Quantum algorithms: An overview, npj Quantum Inf. **2**, 15023 (2016).

[41] Similar to the procedure of constructing creation and annihilation operators from the position and momentum operators of a quantum harmonic oscillator, a set of discrete creation and annihilation operators can be constructed from the discrete field operators and their conjugate momenta in Eq. (9); see Ref. [29, p. 7].

[42] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, New York, 2011).

[43] R. A. Horn and C. Johnson, *Matrix Analysis* (Cambridge University Press, New York, 2012), 2nd ed.

[44] G. Beylkin, On the representation of operators in bases of compactly supported wavelets, SIAM J. Numer. Anal. **29**, 1716 (1992).

[45] S. Harris and D. Harris, *Digital Design and Computer Architecture* (Morgan Kaufmann, Boston, 2016).

[46] R. Babbush, D. W. Berry, J. R. McClean, and H. Neven, Quantum simulation of chemistry with sublinear scaling in basis size, npj Quantum Inf. **5**, 92 (2019).

[47] R. Babbush, D. W. Berry, I. D. Kivlichan, A. Y. Wei, P. J. Love, and A. Aspuru-Guzik, Exponentially more precise quantum simulation of fermions in second quantization, New J. Phys. **18**, 033032 (2016).

[48] R. Bracewell, The fast Hartley transform, Proc. IEEE **72**, 1010 (1984).

[49] A. Klappenecker and M. Roetteler, Quantum software reusability, Int. J. Found. Comput. Sci. **14**, 777 (2003).
[50] C.-C. Tseng and T.-M. Hwang, in *2005 IEEE Int. Symp. Circuits Syst.*, Vol. 1 (IEEE, Kobe, 2005), p. 824.
[51] J. Arndt, *Matters Computational: Ideas, Algorithms, Source Code* (Springer-Verlag, Berlin, 2011).
[52] S. P. Jordan, H. Krovi, K. S. M. Lee, and J. Preskill, BQP-completeness of scattering in scalar quantum field theory, Quantum **2**, 44 (2018).
[53] T. Häner, M. Roetteler, and K. M. Svore, Optimizing quantum circuits for arithmetic, (2018), arXiv:1805.12445.
[54] C. Gidney, Asymptotically efficient quantum Karatsuba multiplication, (2019), arXiv:1904.07356.
[55] D. E. Knuth, *The Art of Computer Programming* Vol. 2 (Addison-Wesley, Boston, 1997), 3rd ed, https://dl.acm.org/doi/10.5555/270146.
[56] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization, Quantum **3**, 208 (2019).
[57] A. M. Childs, J.-P. Liu, and A. Ostrander, High-precision quantum algorithms for partial differential equations, (2020), arXiv:2002.07868.
[58] P. C. S. Costa, S. Jordan, and A. Ostrander, Quantum algorithm for simulating the wave equation, Phys. Rev. A **99**, 012323 (2019).
[59] A. Montanaro and S. Pallister, Quantum algorithms and the finite element method, Phys. Rev. A **93**, 032324 (2016).
[60] S. Arfaoui, A. B. Mabrouk, and C. Cattani, *Wavelet Analysis: Basic Concepts and Applications* (CRC Press, Boca Raton, 2021).
[61] C. K. Chui, *Wavelets: A Mathematical Tool for Signal Analysis* (Society for Industrial and Applied Mathematics, Philadelphia, 1997).
[62] C. A. Neff and J. H. Reif, An efficient algorithm for the complex roots problem, J. Complex. **12**, 81 (1996).
[63] R. P. Brent, J.-a. H. Osborn, and W. D. Smith, Note on best possible bounds for determinants of matrices close to the identity matrix, Linear Algebra Appl. **466**, 21 (2015).
[64] A. Altland and J. von Delft, *Mathematics for Physicists: Introductory Concepts and Methods* (Cambridge University Press, Cambridge, 2019).
[65] B. C. Berndt, *Ramanujan's Notebooks. Part III* (Springer-Verlag, New York, 1991).
[66] M. Benzi and G. H. Golub, Bounds for the entries of matrix functions with applications to preconditioning, BIT Numer. Math. **39**, 417 (1999).
[67] M. Benzi, in *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications* (Springer, Cham, 2016), p. 211.
[68] N. J. Higham, Cholesky factorization, WIREs Comput. Stat. **1**, 251 (2009).