Quantum community detection via deterministic elimination

Chukwudubem Umeano,[1] Stefano Scali,[1] and Oleksandr Kyriienko[2]
[1]*Department of Physics and Astronomy, University of Exeter, Stocker Road, Exeter EX4 4QL, United Kingdom*
[2]*School of Mathematical and Physical Sciences, University of Sheffield, Sheffield S3 7RH, United Kingdom*

(Received 3 April 2025; accepted 15 October 2025; published 12 November 2025)

We propose a quantum algorithm for calculating the structural properties of complex networks and graphs. The corresponding protocol—*deteQt*—is designed to perform large-scale community and botnet detection, where a specific subgraph of a larger graph is identified based on its properties. We construct a workflow relying on ground state preparation of the network modularity matrix or graph Laplacian. The corresponding maximum modularity vector is encoded into a $\log_2(N)$-qubit register that contains community information. We develop a strategy for "signing" this vector via quantum signal processing, such that it closely resembles a hypergraph state, and project it onto a suitable linear combination of such states to detect botnets. As part of the workflow, and of potential independent interest, we present a readout technique that allows filtering out the incorrect solutions deterministically. This can reduce the scaling for the number of samples from exponential to polynomial. The approach serves as a building block for graph analysis with quantum speedup and enables the cybersecurity of large-scale networks.

DOI: 10.1103/s7cm-c9qy

I. INTRODUCTION

Quantum computing (QC) has emerged as a computing paradigm that suits problems with a specific structure [1]. Its applications include problems with inherently quantum behavior, with notable examples lying within chemistry [2] and strongly correlated materials [3], or problems with an algebraic structure [4,5]. Here, the prominent example is an exponential speedup via Shor's algorithm for factoring large integers [6]. This is an example of a Bounded-error Quantum Polynomial time (BQP) complexity class algorithm that relies on efficient solution readout where only a few samples are required (logarithmic in the input size). Other domains that can benefit from quantum subroutines include linear equation solvers [7,8], matrix inversion [9–11], and differential equation solvers [12,13]. However, here the advantage largely depends on the ability to read the solution [14], representing a challenge [15]. Finally, quantum computing offers an advantage in sampling quasiprobability distributions [16,17], motivating the development of quantum generative modeling [18–23].

Moving away from natively quantum or algebraic problems, there remains an open question if quantum computers can help solve problems with a combinatorial structure. On one hand, for provably NP-hard instances of optimization problems we do not expect an exponential advantage [24], while there is still room for efficient heuristics [25–27]. However, recent advances show a largely improved ability to approximate certain optimization problems via decoded quantum interferometry [28], triggering further search. Similarly, quantum computers show promise in topological data analysis (TDA) [29–33], where quantum algorithms can provide insights that are difficult to achieve with classical methods [34–39]. In the domain of graph and network theory, there are examples of quantum algorithms for finding shortest paths [40,41] and detecting cliques [42]. Photonic samplers have been successfully applied to study maximal clique finding for molecular docking [43–45]. The exponential improvement for quantum walks on glued trees suggests a further advantage for selected graph-based problems [46]. Interestingly, there are also BQP algorithms with exponential advantage in simulating large networks of coupled oscillators [47].

Motivated by the examples above, we consider a wide area of complex networks [48] as a potential beneficiary of quantum computing. Complex networks (CNs) represent large multinode graphs with connectivity defined by relations between objects that they model. CNs are ubiquitous in various domains, representing intricate systems such as communication networks, biological networks, social networks, and financial systems [49–51]. One of the key tasks in analyzing these networks is community detection [52], where the objective is to identify groups of nodes that are more interconnected relative to the rest of the network. This task is crucial in various applications, including social media analysis, where communities may represent groups with shared interests [53]; biology, where communities might correspond to functional modules in a protein interaction network [54,55]; and cybersecurity, where identifying clusters of compromised nodes can help in the detection of botnets or coordinated cyberattacks [56]. Despite its importance, community detection becomes increasingly challenging as networks grow in size and

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

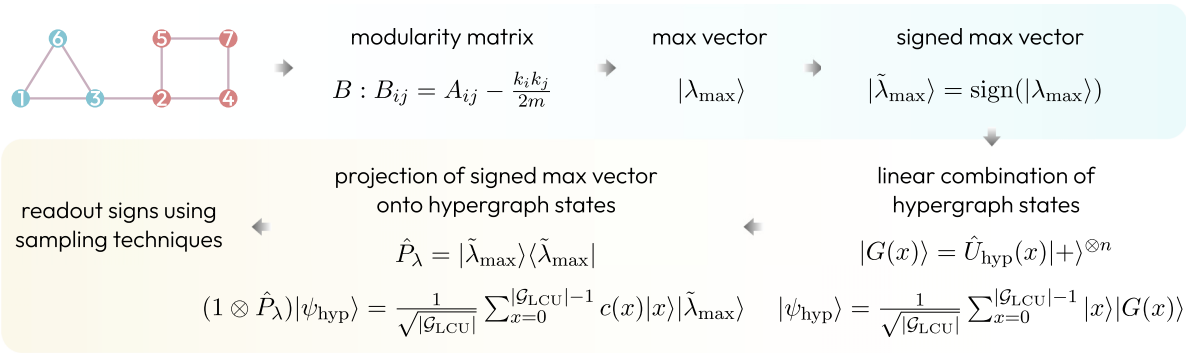

FIG. 1. Flowchart summarizing *deteQt*—the developed quantum algorithms for modularity-based community and botnet detection. The protocol consists of several steps, including finding the leading eigenvector of the modularity matrix, and readout sampling techniques to infer its signs. We detail each step and subroutine in Sec. II.

complexity [57]. To date, the attempts to connect quantum systems and complex networks have mostly remained at a conceptual level [58] and much work is needed to bring full QC benefits to hard community detection and graph partitioning problems.

Classical algorithms for community detection, such as the Girvan-Newman algorithm [59], modularity-based spectral methods [60], and spectral clustering [61], have made significant strides in this area. However, these methods often struggle with scalability and accuracy in large-scale networks. The graph partitioning problem [62,63], closely related to community detection, is also of significant interest, particularly in the context of optimizing the division of a network into smaller, more manageable subgraphs [64]. This problem is critical in parallel computing [65], very Large Scale integration (VLSI) design [66], and load balancing [67]. Current state-of-the-art classical protocols, like the Louvain method [68] and the use of Markov clustering (MCL) [69], attempt to partition large networks efficiently but are limited by computational resources and the exponential growth of problem complexity.

In this paper, we propose a quantum algorithm designed to calculate the structural properties of complex networks and graphs. Our protocol, named *deteQt*, is specifically developed for large-scale community and botnet detection. We introduce the workflow based on ground state preparation for a modularity matrix and quantum spectral analysis. By utilizing tools such as block encodings, linear combinations of unitaries, and quantum singular value transformation, we project the leading eigenvector of the modularity matrix, which encodes the community structure, to identify distinct communities. Furthermore, we introduce an adaptive readout technique that efficiently identifies nodes under specified conditions, which we coin as deterministic elimination (defined by properties of hypergraph states).

Looking toward applying quantum methods to the structural and topological analysis of complex networks, we consider the field of cybersecurity, where early and accurate detection of malicious activity is critical. We connect our protocols with malicious botnet detection, where the structure of such a subgraph is efficiently discovered. The proposed *deteQt* approach not only provides a building block for quantum-accelerated graph analysis but also enhances the cybersecurity of large-scale networks.

II. PROTOCOL: SUBROUTINES

We proceed to describe the quantum approaches to studying complex network properties, in particular concentrating on graph partitioning, community structure learning, and botnet detection. To this end, we have developed a quantum protocol to tackle the problem of modularity-based community detection. We refer to it as the *deteQt* protocol, which can use an exponentially improved number of operations as compared to classical analogs, for specified readout conditions and network types. A schematic of the full *deteQt* protocol is given in Fig. 1, where we describe the required steps from instantiating the network to sampling the partitions.

A. Input

Our procedure is able to work on any network. However, it is most effective when there is a clear community structure; i.e., there exists a subset of nodes that are densely connected to each other, but sparsely connected to the rest of the network. We encode the graph using the modularity matrix B, with elements

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}, \tag{1}$$

where A_{ij} are elements of the adjacency matrix: $A_{ij} = 1$ if node i is connected to node j, otherwise $A_{ij} = 0$. As before, k_i is the degree of node i, and m is the total number of edges in the network.

B. Ground state preparation of max vector

The task of modularity-based community detection is to find the optimal vector s^{opt} that maximizes the modularity Q [52]. Modularity-based methods can be used to divide a network into an arbitrary number of communities. Here, we focus on partitioning into two communities. In this case, we can write the equation for modularity as

$$Q = \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j = \frac{1}{4m} \mathbf{s}^{\mathbf{T}} \mathbf{B} \mathbf{s}, \tag{2}$$

where $s_i = \pm 1$ depending on which community node i belongs to. Note that hierarchical approaches can be used once

we need to detect more communities [52]. An effective way of finding s^{opt} is to find the eigenvector corresponding to the largest eigenvalue of **B**. We refer to this leading eigenvector as the max vector, or $|\lambda_{\max}\rangle$. Taking the sign of the elements of $|\lambda_{\max}\rangle$ gives us the quasioptimal bipartitioning $|s^{\text{opt}}\rangle = \text{sign}(|\lambda_{\max}\rangle)$. From here on, we will refer to the smaller partition as the *botnet*.

Since the modularity matrix for an N-node network is an $N \times N$ matrix, it can be loaded onto a quantum computer using $n = \lceil \log_2 N \rceil$ qubits. To prepare $|\lambda_{\max}\rangle$, we change $B \to -B$, such that $|\lambda_{\max}\rangle$ is now the ground state. This can be efficiently prepared on a quantum computer for operators (Hamiltonians) with specific structure and nonvanishing spectral gap, such as the Hamiltonians of strongly correlated models [2,10,70]. Modularity matrices of graphs with some distinct community structure also possess nonvanishing spectral gaps; hence GSP protocols in these cases are efficient. There are several quantum techniques for ground state preparation (GSP). The right choice of procedure depends on the resources available and the fidelity required.

C. "Signing" the max vector

Once we have the ground state $|\lambda_{\max}\rangle$, which can always be expanded in the computational basis as $|\lambda_{\max}\rangle = \sum_{x=0}^{2^n-1} c_x |x\rangle$, the next task is to apply the sign function onto the amplitudes of the state. This transforms it into a real equally weighted (REW) state of the form [71]

$$\left|\psi^n_{\text{REW}}\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \text{sign}(c_x)|x\rangle, \tag{3}$$

with $\text{sign}(c_x) = 1$ if $c_x > 0$ and $\text{sign}(c_x) = -1$ otherwise (note that the amplitudes of $|\lambda_{\max}\rangle$ are always real). For example, if we have the max vector $|\lambda_{\max}\rangle = [0.602, 0.372, -0.602, 0.372]$, we aim to transform it into the REW state $|\tilde{\lambda}_{\max}\rangle = \frac{1}{2}[1, 1, -1, 1]$. REW states have a prominent role in several primordial quantum algorithms, such as Simon's [4], Grover's [72], and the Deutsch-Jozsa algorithm [73].

We have two main ingredients for performing the transformation $|\lambda_{\max}\rangle \to |\tilde{\lambda}_{\max}\rangle$. The first step is the block encoding of the amplitudes of $|\lambda_{\max}\rangle$ on the diagonal of a matrix [74]. For simplicity, we give the following details assuming we have a n-qubit unitary state preparation circuit $\hat{U}_{\max}$ such that $\hat{U}_{\max}|0\rangle^{\otimes n} = |\lambda_{\max}\rangle$. This encompasses GSP strategies such as the variational quantum eigensolver [75] or adiabatic state preparation [10], where $\hat{U}_{\max}$ would be our trained variational ansatz. However, it is important to note that this block-encoding protocol can be extended to nonunitary GSP methods. As the authors of [76] demonstrate, any state preparation block encoding (which encompasses many nonunitary GSP protocols) can be converted into a unitary state preparation circuit using fixed point amplitude amplification.

By using a circuit involving calls to controlled $\hat{U}_{\max}$, we can prepare a $(2n+2)$-qubit block-encoding unitary $\hat{U}_{\text{BE}}$ [76] corresponding to

$$\hat{U}_{\text{BE}} = \begin{pmatrix} * & * \\ * & A \end{pmatrix}, \; A = (\langle 0^{\otimes n+2}| \otimes \mathbb{1}_n)\hat{U}_{\text{BE}}(|0^{\otimes n+2}\rangle \otimes \mathbb{1}_n), \tag{4}$$

and the matrix A is a diagonal matrix containing the amplitudes of $|\lambda_{\max}\rangle$,

$$A = \begin{pmatrix} c_0 & 0 & \dots & 0 \\ 0 & c_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_{2^n-1} \end{pmatrix}. \tag{5}$$

We show the circuits to perform the block encoding in the Appendix (Fig. 6).

Now that we have the amplitudes of $|\lambda_{\max}\rangle$ encoded along the diagonal of a matrix, we can use quantum singular value transformation (QSVT) to apply the sign function directly onto the amplitudes. The full QSVT sequence $\hat{U}_{\text{QSVT}}$ is a $(2n+3)$-qubit block encoding of the transformed matrix $\text{sign}(A)$ corresponding to

$$\hat{U}_{\text{QSVT}} = \begin{pmatrix} * & * \\ & \text{sign}(A) \end{pmatrix}, \tag{6}$$

$$\text{sign}(A) = (\langle 0^{\otimes n+3}| \otimes \mathbb{1}_n)\hat{U}_{\text{QSVT}}(|0^{\otimes n+3}\rangle \otimes \mathbb{1}_n), \tag{7}$$

and the matrix $\text{sign}(A)$ is a diagonal matrix containing the signed amplitudes of $|\lambda_{\max}\rangle$,

$$\text{sign}(A) = \begin{pmatrix} \text{sign}(c_0) & 0 & \dots & 0 \\ 0 & \text{sign}(c_1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \text{sign}(c_{2^n-1}) \end{pmatrix}, \tag{8}$$

$$\text{sign}(A)|+\rangle^{\otimes n} = |\tilde{\lambda}_{\max}\rangle. \tag{9}$$

QSVT is primarily used for applying polynomial functions onto matrices; hence we can only approximate the nonpolynomial sign function. We provide more details on the full QSVT sequence, as well as approximation methods for the sign function in the Appendix.

D. Readout

1. Hypergraph state LCU preparation

With access to $|\tilde{\lambda}_{\max}\rangle$ through QSVT, the question is how to extract the signs efficiently to find the botnet. We note that the ability to read out solutions (classical information) from quantum states draws the boundary between high-performing algorithms in the BQP family and generic quantum algorithms. To perform the readout, we develop a technique based on quantum hypergraph states. The hypergraph states represent a generalization of graph states that read as [77]

$$|G\rangle = \prod_{k=1}^{n} \prod_{\{i_1, i_2, \dots, i_k\} \in E} C^k Z_{i_1 i_2 \dots i_k} |+\rangle^{\otimes n}. \tag{10}$$

Essentially, for each k hyperedge $\{i_1, i_2, \dots, i_k\} \in E$, we apply a controlled-$Z$ operation on the connected qubits. These general hypergraph states permit hyperedges of any order k (for $k = 1$, a single-qubit Z gate is applied on the corresponding qubit). Figure 2(d) depicts an example for $n = 7$ qubits.

We can denote the set of all hypergraph states as $\mathcal{G}$. In total, $|\mathcal{G}| = \prod_{k=1}^{n} 2^{nC_k} = 2^{2^n-1}$, and each hypergraph state is a REW state [specified in Eq. (3)]. Since $|\tilde{\lambda}_{\max}\rangle$ is also a REW state (up to some error arising from the QSVT approximation of the sign function), we know that one of our hypergraph states

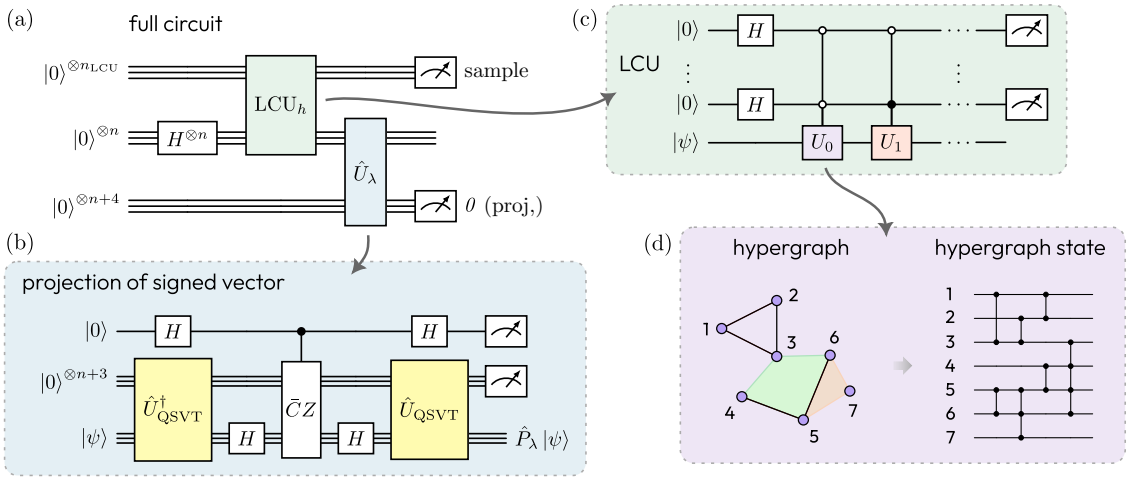

FIG. 2. (a) Full circuit for projecting the signed max vector onto the hypergraph superposition state. The first layer of Hadamards prepares the state $|+\rangle^{\otimes n}$, which the hypergraph circuits act upon [as in Eq. (10)]. (b) Circuit $\hat{U}_\lambda$ for the projection of the signed max vector onto the system register [when $\mathbf{0}$ is measured on the $(n+4)$ ancilla qubits]. Here $\bar{C}Z = -X^{\otimes m}(C^{m-1}Z)X^{\otimes m}$ is the second term of Eq. (13) implemented as a single-ancilla LCU and $H \equiv H^{\otimes n}$. We show the component circuits of $\hat{U}_{\rm QSVT}$ in the Appendix (Figs. 6 and 8). (c) LCU circuit for the preparation of the hypergraph superposition state. The LCU_h block has the standard form of the LCU circuit but without the final PREPARE operation (in this case, a layer of Hadamards). The top register is then sampled to retrieve the bipartition. (d) Mapping between the hypergraphs and the circuits preparing the hypergraph states which make up the LCU. Black lines on the graph (e.g., 1-2 and 4-5) represent edges between the nodes of the hypergraph. These translate into CZ gates (black connected dots) between corresponding qubits in the hypergraph state generation. Shaded regions correspond to hyperedges between multiple qubits, e.g., 5-6-7 and 3-4-5-6 in the figure. These translate into multicontrolled phase gates. The multiqubit CZ gates act on a $|+\rangle^{\otimes 7}$ state prepared from the computational zero state with Hadamards.

corresponds exactly to the bipartition that we are trying to find. In a brute force way, we can find the bipartition by simply evaluating $|\langle\tilde{\lambda}_{\max}|G\rangle|^2$ (using, e.g., a Hadamard test) for all hypergraph states $|G\rangle \in \mathcal{G}$ until we find $|\langle\tilde{\lambda}_{\max}|G\rangle|^2 \approx 1$: the signs of this state give us the bipartition. Of course, measuring each overlap individually is costly, not to mention that we may have to trial an exponential number of hypergraph states before the correct one is found. Instead, we can exploit quantum parallelism to devise a more efficient protocol. For this, we use the linear combination of unitaries (LCU) method to create a superposition of hypergraph states, with the set of unitaries being the controlled-Z operations. Importantly, we can reduce the number of states used in the LCU, $\mathcal{G}_{\rm LCU} \subset \mathcal{G}$ if there is prior knowledge of the size of the botnet. We will provide more rationale for the exact choice of $\mathcal{G}_{\rm LCU}$ in Sec. II D 3. Our hypergraph superposition state has the following form (up to normalization):

$$|\psi_{\rm hyp}\rangle = \frac{1}{\sqrt{|\mathcal{G}_{\rm LCU}|}} \sum_{x=0}^{|\mathcal{G}_{\rm LCU}|-1} |x\rangle|G(x)\rangle. \tag{11}$$

It is important to note that we actually do not use the full LCU protocol to create this state; we include the initial layer of Hadamards and the SELECT operator, but the final layer of Hadamards is omitted, as Fig. 2(c) shows. This circuit gives a superposition of hypergraph states on the second register labeled by a unique bitstring on the first register. This will prove to be crucial to our method, as we will explain in the following sections.

2. *Projection onto signed vector*

We explained in the previous section that evaluating overlaps $|\langle\tilde{\lambda}_{\max}|G\rangle|^2$ is enough to find the bipartition. By projecting onto the hypergraph superposition state, we are able to retrieve the necessary information about all these overlaps without explicitly calculating them one by one. This projector, $\hat{P}_\lambda = |\tilde{\lambda}_{\max}\rangle\langle\tilde{\lambda}_{\max}|$, can be retrieved from the full QSVT circuit, $\hat{U}_{\rm QSVT}$, as we detail in the following.

We know that $\mathrm{sign}(A)|+\rangle^{\otimes n} = |\tilde{\lambda}_{\max}\rangle$. This matrix $\mathrm{sign}(A)$ is encoded within $\hat{U}_{\rm QSVT}$ (6), and we can define a state $|\psi_{\rm QSVT}\rangle = \hat{U}_{\rm QSVT}|0\rangle_a|+\rangle_s$, where we use $a = [1\ldots n+3]$ as the set of $n+3$ ancilla qubits, and $s = [n+4\ldots 2n+3]$ denoting the set of n system qubits. Hence we know if we can project using the projector $\hat{P}_{\rm QSVT} = |\psi_{\rm QSVT}\rangle\langle\psi_{\rm QSVT}|$, then measure $\mathbf{0}$ on the ancilla qubits, we can effectively project onto the system register with $\hat{P}_\lambda$ (since $\mathrm{tr}_a\{\hat{P}_{\rm QSVT}\} = \hat{P}_\lambda$). Expanding further, we have

$$\hat{P}_{\rm QSVT} = |\psi_{\rm QSVT}\rangle\langle\psi_{\rm QSVT}| = \hat{U}_{\rm QSVT}H_s|0\rangle_{as}\langle 0|_{as}H_s\hat{U}^\dagger_{\rm QSVT}, \tag{12}$$

where H_s represents a layer of Hadamards applied on the system register, and $|0\rangle_{as}\langle 0|_{as}$ denotes the zero projector acting on both registers. Hence this sequence for projection can be constructed as a quantum circuit with ancilla measurements. To prepare the zero projector, we note that this can be done efficiently as a simple two-term LCU [78],

$$|0\rangle\langle 0|^{\otimes m} = \frac{\mathbb{1}_m - X^{\otimes m}(C^{m-1}Z)X^{\otimes m}}{2}, \tag{13}$$

which requires a single ancilla. The full circuit to perform the projection is shown in Fig. 2(b).

We now have the means to project onto the hypergraph superposition state, which has the following form (up to normalization):

$$(\mathbb{1} \otimes \hat{P}_\lambda)|\psi_{\text{hyp}}\rangle = \frac{1}{\sqrt{|\mathcal{G}_{\text{LCU}}|}} \sum_{x=0}^{|\mathcal{G}_{\text{LCU}}|-1} c(x)|x\rangle|\tilde{\lambda}_{\max}\rangle, \quad (14)$$

$$c(x) = |\langle\tilde{\lambda}_{\max}|G(x)\rangle|. \quad (15)$$

The full circuit for this projection is shown in Fig. 2(a).

3. Measurement: zero-overlap selection

The final step is to sample the top register of the circuit in Fig. 2(a), which is the ancilla register of the hypergraph LCU. As Eq. (14) shows, the probability of measuring each bitstring x depends on the overlap between its corresponding hypergraph state and the signed max vector: $p(x) \propto c(x)^2 = |\langle\tilde{\lambda}_{\max}|G(x)\rangle|^2$. For a particular $|\tilde{\lambda}_{\max}\rangle$, we can classify bitstrings x into two classes,

$$x \in X_1^\lambda \iff c(x)^2 > 0,$$
$$x \in X_2^\lambda \iff c(x)^2 = 0. \quad (16)$$

Hence if we measure the first register, we are only able to measure bitstrings $x \in X_1^\lambda$. If we choose the hypergraph states used within the LCU sensibly, we will always have zeros in the probability distribution (i.e., $|X_2^\lambda| > 0$). This is the property we can exploit in order to extract the botnet using just a few measurements. As for the choice of hypergraph states for the LCU, $\mathcal{G}_{\text{LCU}} = \{|G(x)\rangle\} \subset \mathcal{G}$, let us assume we have a network with N nodes and a botnet of size k. Therefore, we know that our candidate solutions $\mathcal{G}_{\text{cand}} \subset \mathcal{G}$ correspond to all REW states with a botnet of size k, i.e., $N-k$ positive amplitudes and k negative amplitudes (or vice versa, depending on convention). As for the selection of states $\mathcal{G}_{\text{LCU}}$, we choose the hypergraph states with a botnet of size $k_{\text{LCU}} = \frac{N}{2} - k$. This selection keeps the depth of the LCU as short as possible while ensuring that each candidate solution $|G_c\rangle \in \mathcal{G}_{\text{cand}}$ (including the true solution $|\tilde{\lambda}_{\max}\rangle$) has a unique probability distribution and a unique set of zeros $X_2^{G_c}$,

$$x \in X_1^{G_c} \iff |\langle G_c|G(x)\rangle|^2 > 0,$$
$$x \in X_2^{G_c} \iff |\langle G_c|G(x)\rangle|^2 = 0. \quad (17)$$

For example, if we know our network has $N = 16$ nodes and a botnet of size $k = 3$, we have $\binom{16}{3} = 560$ candidate solutions, $|\mathcal{G}_{\text{cand}}| = 560$. For the hypergraph LCU we choose the states with a botnet of size $k_{\text{LCU}} = \frac{16}{2} - 3 = 5$, giving an LCU with $\binom{16}{5} = 4368$ unitaries, $|\mathcal{G}_{\text{LCU}}| = 4368$, implementable with $\lceil\log_2(4368)\rceil = 13$ ancilla qubits. Each of the 560 candidate states has a unique set of zeros $X_2^{G_c}$, as we visualize in Fig. 3. In the special case where $k = \frac{N}{2}$, we select the hypergraph states with botnet size $k_{\text{LCU}} = 2$. If we are uncertain about the exact size of the botnet, we can extend $\mathcal{G}_{\text{LCU}}$ to include hypergraph states with a range of botnet sizes. Alternatively, we can precalculate the size of the botnet by measuring a single overlap, as $k = \frac{N-(\sqrt{N}\sqrt{2^n}|\langle+^{\otimes n}|\tilde{\lambda}_{\max}\rangle|)}{2}$.

Armed with these sets of zeros $X_2^{G_c}$ (which can be calculated easily beforehand), we use a process of sampling and elimination to find the botnet. We measure the top register of the circuit in Fig. 2(a) in the computational basis, retrieving a bitstring $x \in X_1^\lambda$, labeling a hypergraph state $|G(x)\rangle$. With this single measurement, we can deterministically eliminate each candidate state $|G_c\rangle$ where $x \in X_2^{G_c}$, as it would have been impossible to sample x if $|G_c\rangle$ is our solution state $|\tilde{\lambda}_{\max}\rangle$. Returning to the example in Fig. 3, if bitstring $x = 1000$ was measured, we would eliminate $G_c \in \mathcal{G}_{\text{cand}}$ for $c \in [105, 107, 108, 109, 110, 114, 115, \ldots, 557, 558, 559]$.

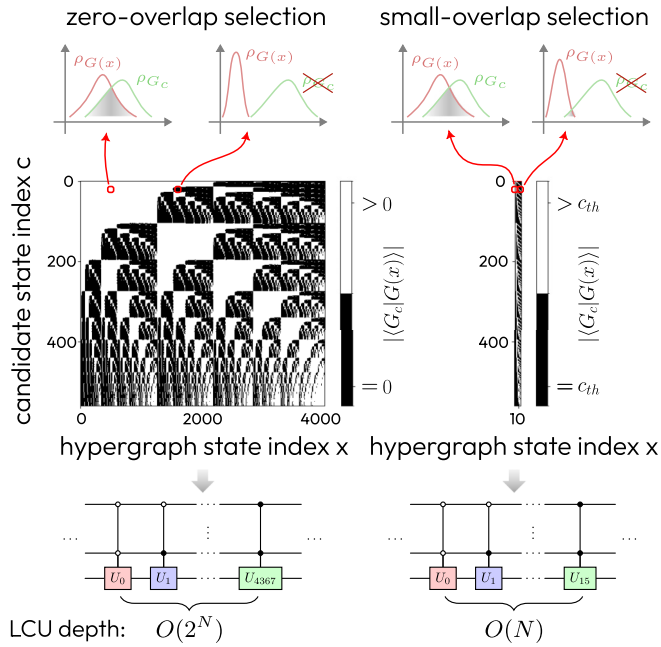

FIG. 3. Comparison of the zero-overlap selection and small-overlap section methods, for $N = 16$ nodes and botnet size $k = 3$. We have $\binom{16}{3} = 560$ candidate states, and we visualize the overlaps of these states with the states in the hypergraph LCU. For both methods, the black rectangles appear at different positions on each row, indicating that each candidate state has a unique set of zero (small) overlaps $X_2^{G_c}$. In the zero-overlap case, we select hypergraph states with botnet size $k_{\text{LCU}} = \frac{16}{2} - 3 = 5$, giving an LCU with $\binom{16}{5} = 4368$ unitaries. In the small-overlap case, we select hypergraph states with botnet size $k_{\text{LCU}} = 1$, giving an LCU with $\binom{16}{1} = 16$ unitaries. Changing from zero-overlap to small-overlap selection gives us an exponential decrease in the LCU depth, which is offset by an increase in the number of circuit runs.

We repeat the process of measurement, then elimination, until one candidate state remains; this $|G_c\rangle$ is our true solution $|\tilde{\lambda}_{\max}\rangle$. We simply take the signs of this state (which we already know) to retrieve the botnet.

4. Measurement: small-overlap selection

In theory, the procedure detailed above allows us to find $|\tilde{\lambda}_{\max}\rangle$ with near certainty within a logarithmic number of measurements with respect to the number of potential solutions. However, this comes with the caveat of a near-exponential depth hypergraph LCU circuit in the case when the botnet size k is small, $\binom{N}{\frac{N}{2}-k} \sim \binom{N}{\frac{N}{2}} \to 2^N$ for $k \to 1$. While LCU circuits for certain groups of REW states can be simplified to have efficient representations [79], this is challenging for arbitrary composition of states.

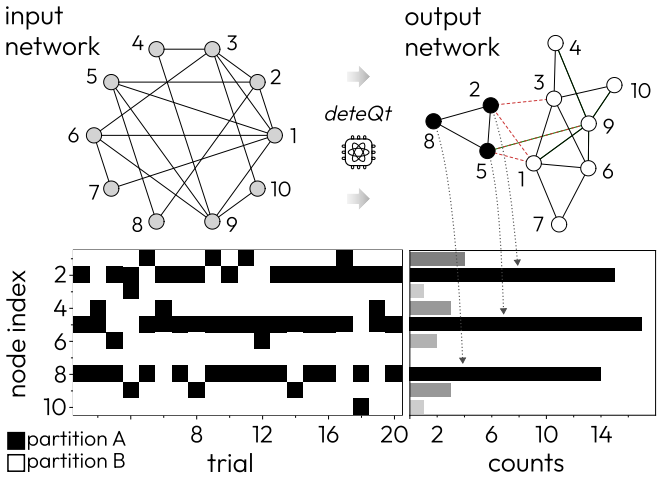

FIG. 4. Example of a *deteQt* run using small-overlap selection. The network considered consists of $N = 10$ nodes (vertices), 18 interactions (edges), and a hidden botnet of size $k = 3$. We run *deteQt* 20 times (trials) reading out the estimated partitions A and B. By counting the occurrences of the single nodes in the partitions, we build the probability distribution to estimate the botnet composition (in our example, 2-5-8).

To counter this, we can venture away from selecting the hypergraph LCU states based on finding zero overlaps, $|\langle G_c|G(x)\rangle|^2 = 0$. To illustrate the point, rather than selecting $k_{\mathrm{LCU}} = \frac{N}{2} - k$, we now select $k_{\mathrm{LCU}} < \frac{N}{2} - k$. Each candidate solution $|G_c\rangle \in \mathcal{G}_{\mathrm{cand}}$, including the true solution $|\tilde{\lambda}_{\max}\rangle$ still has a unique probability distribution, but we now separate bitstrings based on small and large overlaps, instead of zero and nonzero overlaps:

$$x \in X_1^{G_c} \iff |\langle G_c|G(x)\rangle|^2 > c_{\mathrm{th}},$$
$$x \in X_2^{G_c} \iff |\langle G_c|G(x)\rangle|^2 = c_{\mathrm{th}}. \tag{18}$$

This threshold overlap is defined as $c_{\mathrm{th}} = \frac{1}{\sqrt{N2^n}}|2(\frac{N}{2} - k - k_{\mathrm{LCU}})|$. This is the smallest possible overlap between a candidate state $|G_c\rangle$ with botnet size k and a hypergraph LCU state $|G(x)\rangle$ with botnet size k_{LCU}. Again, each set of small overlaps $X_2^{G_c}$, including that of our true solution X_2^{λ}, is unique to each candidate state. This allows for a significant reduction in the LCU circuit depth, as Fig. 3 shows. We apply the same sampling and elimination process to find the botnet, with the caveat that we now have a nonzero probability of sampling $x \in X_2^{\lambda}$. Hence there is a chance of eliminating the true solution $|\tilde{\lambda}_{\max}\rangle$ within this process; we cannot be 100% sure that the last candidate state standing is indeed the true solution.

To cater for this, we repeat the protocol multiple times, before collating statistics to determine what our true botnet is. We show an example of this in Fig. 4. This network has $N = 10$ nodes and a botnet size $k = 3$, but we select $k_{\mathrm{LCU}} = 1$. This gives us a threshold for small overlaps of $c_{\mathrm{th}} = \frac{1}{\sqrt{10\times 16}}|2(\frac{10}{2} - 3 - 1)| \approx 0.158$. Even though the protocol only finds the state corresponding to the true botnet 7 times out of 20 trials, the other trial solutions found have high overlap with our true solution. Counting the frequency at which each node appears in our trial solutions clearly unveils the true botnet.

III. PROTOCOL: SCALING ANALYSIS

Brute-force approaches to community detection involve querying all possible bipartitions, before selecting the one that minimizes the modularity or an alternative community detection metric. These naive methods have a time complexity scaling exponentially with the system size for generic graphs. We also note that state of the art classical algorithms for community detection, such as the Girvan-Newman algorithm and the Louvain method, typically boast a linear or quadratic time complexity in terms of the number of edges [52,59,68]. For dense graphs, the number of edges scales quadratically with the number of nodes N, so, in this regime, the scaling for classical methods ranges between $O(N^2)$ and $O(N^5)$. Other methods such as spectral clustering also rely on exact diagonalization, which quickly becomes infeasible at larger system sizes.

We now present some prospective scalings for the end-to-end protocol, extremely important to analyze as we test its ability to outperform classical analysis. We give these scalings as a function of (1) number of nodes in the network, N; (2) size of the botnet, k; (3) degree of the polynomial approximation of the sign function, d. Here we assume that we are dealing with a network with a known botnet of size k, such that we can restrict $|\mathcal{G}_{\mathrm{LCU}}|$ as detailed previously. During this report we have written qubit counts as a function of $n = \lceil \log_2 N \rceil$, so any quantity that has a linear scaling with the number of qubits actually has a logarithmic scaling with the number of nodes, $O(n) = O(\log N)$.

A. Number of qubits

Analyzing Fig. 2(a), we can see that the full circuit for the protocol uses $2n + 4 + \log_2 |\mathcal{G}_{\mathrm{LCU}}|$ qubits; $2n + 4 = O(\log N)$ and $|\mathcal{G}_{\mathrm{LCU}}|$ depends on whether we choose to perform readout using zero-overlap selection or small-overlap selection.

If we elect for the small-overlap method, we choose k_{LCU} as a small integer a independent of botnet size k. We have $|\mathcal{G}_{\mathrm{LCU}}| = \binom{N}{k_{\mathrm{LCU}}}$, and since $a \ll N$, it follows that $|\mathcal{G}_{\mathrm{LCU}}| = \binom{N}{a} \sim O(N^a)$. We are generally free to choose $k_{\mathrm{LCU}} = 1$; this gives us an exact linear scaling, $|\mathcal{G}_{\mathrm{LCU}}| = \binom{N}{1} = N$.

On the other hand, if we are searching for a botnet of size k and we choose the zero-overlap method, we prepare the hypergraph LCU using circuits corresponding to states with a botnet of size $k_{\mathrm{LCU}} = \frac{N}{2} - k$, and $|\mathcal{G}_{\mathrm{LCU}}| = \binom{N}{k_{\mathrm{LCU}}}$. We acknowledge that in the extreme worst case, when $k \simeq 1$ ($k_{\mathrm{LCU}} \simeq \frac{N}{2}$), $|\mathcal{G}_{\mathrm{LCU}}| \sim O(c^N)$, with $1 < c < 2$. However a more generic network tends to have larger communities; hence we give the scaling for these more common cases. Specifically, we take the case when $k = \frac{N}{2} - a$ ($k_{\mathrm{LCU}} = a$), where a is some integer and $a \ll N$. As in the small-overlap case, the number of states in the LCU in this situation roughly scales polynomially: $|\mathcal{G}_{\mathrm{LCU}}| = \binom{N}{a} \sim O(N^a)$.

Overall, both approaches give us a total qubit count of $O(\log N) + O(\log N^a) \simeq O(a \log N)$. This compact, logarithmic encoding of the problem is key for the scalability of our protocol to large graphs, where exact diagonalization-based protocols become classically intractable.

B. Gate counts

For the gate counts (circuit depth), we give scalings in terms of the number of single- and two-qubit gates required to implement the circuit.

Looking again at the circuit in Fig. 2(a), we have two main blocks to analyze: the hypergraph LCU and the unitary $\hat{U}_\lambda$ projecting onto the signed max vector. Starting with the former, the depth of the hypergraph LCU circuit scales with the number of unitaries in the LCU, $|\mathcal{G}_{\mathrm{LCU}}| \sim O(N^a)$. Each hypergraph state is made up of a constant number of multi-qubit-controlled gates [exemplified in Fig. 2(d)], and an $n = \log N$-qubit controlled operator can be broken down using $O(\log N)$ single- and two-qubit gates [76]. Therefore, the circuit depth of the hypergraph LCU is $O(N^a \log N)$.

As for $\hat{U}_\lambda$, the circuit depth is dominated by the two copies of $\hat{U}_{\mathrm{QSVT}}$ [Fig. 2(b)], which in turn is dominated by the d repetitions of the block-encoding circuit $\hat{U}_{\mathrm{BE}}$ (Fig. 8). $\hat{U}_{\mathrm{BE}}$ contains six queries to controlled $\hat{U}_{\max}$ (Fig. 6; also see [76]), so the scaling depends on the method of ground state preparation. As an example, we take $\hat{U}_{\max}$ to be a variational quantum eigensolver (VQE) ansatz with a depth equal to the number of qubits. This circuit depth is sufficient for general Hamiltonians with a nonvanishing spectral gap, and we note that even an approximate GSP protocol is enough, as long as the signs of the amplitudes are correct. This gives us a $\log N$-qubit circuit with $O(\log N)$ depth: $\hat{U}_{\max}$ has a total of $O(\log^2 N)$ gates. As for controlled $\hat{U}_{\max}$, each of the $\log^2 N$ gates can be decomposed using $O(\log N)$ single- or two-qubit gates, giving a circuit depth of $O(\log^3 N)$ gates for $\hat{U}_{\mathrm{BE}}$. The QSVT circuit contains d repetitions of $\hat{U}_{\mathrm{BE}}$, giving $\hat{U}_{\mathrm{QSVT}}$, and hence $\hat{U}_\lambda$ a circuit depth of $O(d \log^3 N)$.

Therefore, the circuit depth for the full protocol is $O(N^a \log N) + O(d \log^3 N)$. The term that dominates is determined by the size of the botnet, the method of ground state preparation, and the desired error of the sign function approximation.

C. Circuit repetitions

Figure 2(a) showcases one reason that circuit repetitions are required: only when the $n + 4$ ancilla qubits are measured as $\mathbf{0}$ is the signed max vector projected onto the hypergraph superposition state. The number of repetitions required depends on the probability of measuring $\mathbf{0}$. In general we require $O(N)$ repetitions to perform the projection, but we note here that we can boost the probability of measuring all zeros on the ancilla register by using amplitude amplification. This would reduce the number of circuit repetitions to $O(1)$, at the expense of an increase in $O(\sqrt{N})$ to the circuit depth.

As for the sampling, as we see in Fig. 3, just a single sampled bitstring x is enough to eliminate roughly half of the set of candidate solutions. This is where we see a trade-off appearing: networks with larger botnets require fewer states in the hypergraph LCU $\mathcal{G}_{\mathrm{LCU}}$, hence smaller circuit depths, but have a larger number of candidate states $\mathcal{G}_{\mathrm{cand}}$, which requires a larger number of measurements to complete the filtration. Dealing again with the case where $k = \frac{N}{2} - a$, we have $\mathcal{G}_{\mathrm{cand}} \sim O(c^N)$, with $1 < c < 2$. Taking the assumption that roughly half the set of candidate solutions is eliminated after each measurement, we require $O(\log c^N) \approx O(N)$ samples to go from the full set of candidates to a trial solution.

For both readout methods, some repetitions of the full protocol are required to ensure the accuracy of the final solution. In the small-overlap selection case, as mentioned in Sec. II D 4, we repeat the protocol several times to cater for the significant chance of eliminating the true solution within a single trial. The number of trials required depends on the size of these small overlaps, $c_{\mathrm{th}} = \frac{1}{\sqrt{N2^n}}|2(\frac{N}{2} - k - k_{\mathrm{LCU}})|$. This threshold overlap is proportional to the probability of measuring a bitstring $x \in X_2^\lambda$, leading to incorrect elimination of the true solution. Since we take $O(N)$ samples within a single trial, the probability of reaching the end of the trial without eliminating the correct solution can be low, particularly at larger system sizes. Heuristically, we find that taking statistics from $O[N(\frac{N}{2} - k - k_{\mathrm{LCU}})] \lesssim O(N^2)$ trials is enough to mitigate for these issues.

Even in the zero-overlap selection case, we note that there is a very small error probability arising from the fact that we can only approximately apply the sign function. The signed max vector is not exactly a REW state, so there is a very small chance that we measure bitstrings that we do not expect to measure if the sign function was applied perfectly. In other words, there is a nonzero probability of measuring bitstrings $x \in X_2^\lambda$, leading to incorrect elimination of candidate states. The probability of this occurring varies from case to case, as the sign function is applied more accurately when the amplitudes of the max vector are larger; this becomes more problematic at larger system sizes when the amplitudes become smaller on average. With the $1/\sqrt{N}$ scaling in the amplitudes of a uniform state with N components, we propose repeating the whole procedure $O(\sqrt{N})$ times to ensure the accuracy of our solution.

Therefore in the best case scenario, with very high quality sign function application and amplitude amplification, our protocol requires just $O(N)$ samples. In the more general case with minimal circuit depth, we take $O(N)$ repetitions to perform the zero projection. Along with the $O(N)$ samples required to find a solution from a single trial, and the number of trials needed to mitigate for erroneous candidate state elimination ranging from $O(\sqrt{N})$ to $O(N^2)$, the total number of circuit repetitions is $O(N^2)$ in the general case and $O(N^4)$ in the worst case.

IV. APPLICATIONS

Community detection, and more precisely botnet detection, are real-world use cases of what more largely can be referred to as the graph partitioning problem.

One of the critical applications of community detection is anomaly detection in intrusion detection systems (IDSs). Network traffic can be represented as a graph, where nodes represent devices or addresses, and edges represent communication between them. Under normal conditions, the network exhibits a particular community structure, where clusters of nodes represent typical communication patterns within the network. Community detection algorithms can help identify these normal patterns by detecting dense subgraphs corresponding to routine network activity. However, when an

anomaly occurs—such as a cyberattack, a new worm propagation, or unauthorized access—the network community structure can significantly change. When integrated into an IDS, community detection methods can identify sophisticated attacks that traditional signature-based methods might miss. Analyzing the evolution of community structures can reveal stealthy threats like advanced persistent threats, which gradually expand their influence across the network over time.

Similarly, botnets—networks of compromised devices controlled by an attacker—often communicate with a central command-and-control (C2) server. The nodes in the botnet tend to form a community within the larger network, characterized by frequent and coordinated communication. Community detection can identify these clusters of compromised devices, enabling cybersecurity teams to isolate the botnet and cut off communication with the C2 server.

Finally, insider threats, where individuals within an organization misuse their access to harm the organization, pose a significant security challenge. These threats can be difficult to detect because of the legitimate access insiders have to the network and its resources. These covert networks can hide within large communication networks. An insider threat might begin to interact with communities or access resources that are outside their typical community. Community detection algorithms can continuously monitor the network to identify such changes in behavior, flagging potential insider threats.

In Fig. 5, we show example applications of *deteQt* to networks with nontrivial dimensionalities and structures. First, we show an example of a botnet hidden within the global network that behaves like healthy nodes, that is, having similar connection patterns. In this case, the botnet size is $k = 5$; the number of nodes is $N = 50$ with 1213 interactions. Using the small-overlap selection method described in Sec. II D 4, *deteQt* is able to find the solution to the botnet out of a total of more than 2×10^6 possible partitions. To do this, we needed to run the protocol approximately 10^3 times. As a second example, we show a case of an isolated network. In the case of the isolated botnet ($k = 4$), the network studied is composed of $N = 100$ nodes and 4624 interactions, with a total of approximately 4×10^6 possible partitions. To resolve the botnet probability distribution in this case, we needed approximately 10^4 trials.

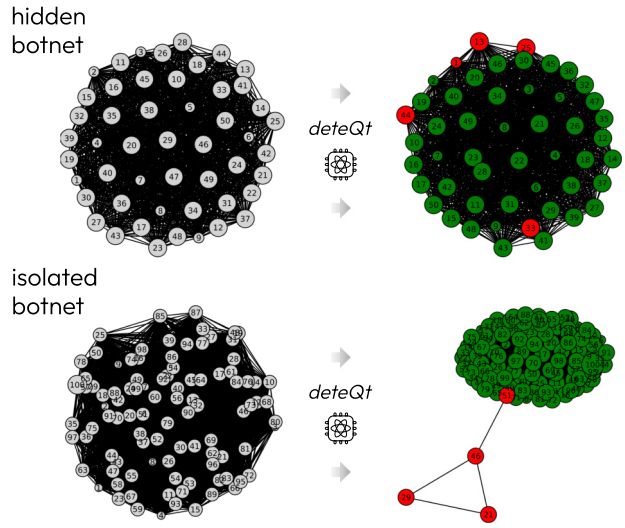

FIG. 5. Example of a *deteQt* run to detect botnets of different types and sizes. We show botnets that mix with the network and behave similarly to healthy nodes (hidden botnets) or botnets whose intranet is isolated from the rest of the network, limiting their exposure to the healthy nodes (isolated botnets). In the case of the hidden botnet ($k = 5$), the network studied is composed of $N = 50$ nodes (vertices) and 1213 interactions (edges) with a total of more than 2×10^6 possible partitions. In the case of the isolated botnet ($k = 4$), the network studied is composed of $N = 100$ nodes and 4624 interactions, with a total of approximately 4×10^6 possible partitions.

V. CONCLUSIONS

In this paper, we have developed quantum computing algorithms for solving problems in complex network analysis, relevant for applications in cyber defense. As such, we introduced *deteQt*—a quantum protocol for resolving the community structure of networks and graphs, offering memory and processing advantages. Motivated by spectral methods, our approach is based on analyzing the leading eigenvector of a modularity matrix (or graph Laplacian) which contains crucial information about communities. Using the tools of ground state preparation and quantum signal processing, we prepared the corresponding real equally weighted quantum state, and proposed techniques for efficient readout of community indices based on deterministic elimination with hypergraph states. Specifically, we apply our approach to the problem of botnet detection, that is, the identification of malicious subnetworks within larger networks. As the search of suitable use cases for early fault-tolerant quantum computers continues, we suggest that complex networks analysis represents a highly promising yet largely overlooked application area. Here, quantum detection can enhance response times to emerging threats and provide powerful tools for addressing complex attacks.

ACKNOWLEDGMENTS

Research is funded by Frazer-Nash Consultancy Ltd. on behalf of the Defence Science and Technology Laboratory (Dstl) which is an executive agency of the U.K. Ministry of Defence. The research supports the Autonomous Resilient Cyber Defence (ARCD) project within the Dstl Cyber Defence Enhancement programme. O.K. was partially supported by UK EPSRC under Award No. EP/Z53318X/1 (QCi3 Hub).

DATA AVAILABILITY

The data that support the findings of this article are openly available [80].

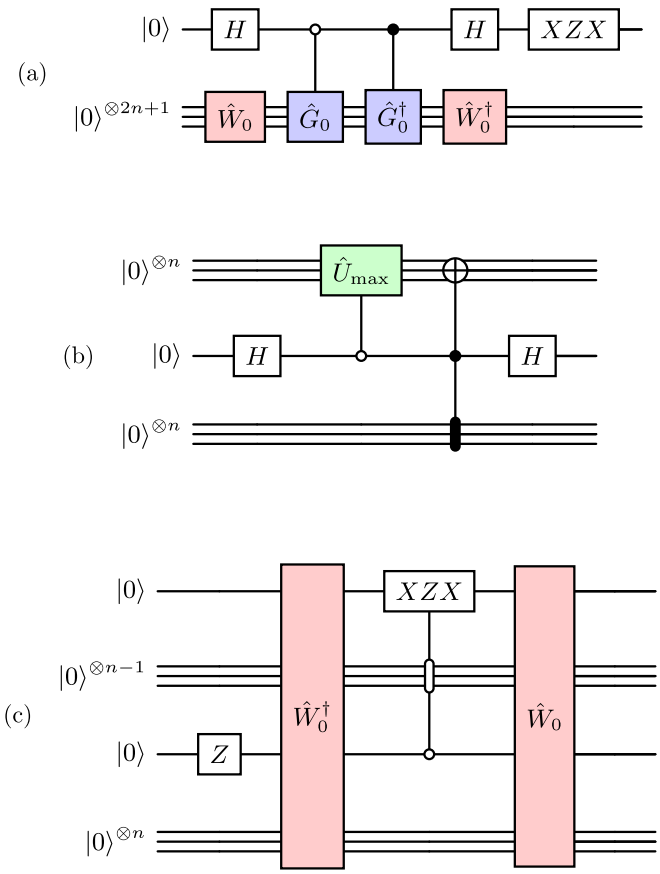

FIG. 6. (a) Full circuit for $\hat{U}_{\mathrm{BE}}$, the diagonal block encoding of the amplitudes of $|\lambda_{\max}\rangle$. $\hat{U}_{\max}$ is the unitary state preparation circuit, and the block XZX represents the sequences of single-qubit gates X, Z, X. The circuits for the components $\hat{W}_0$ and $\hat{G}_0$ are given in (b,c), respectively. See [76] for the full details of this block-encoding procedure.

APPENDIX: QUANTUM SINGULAR VALUE TRANSFORMATION

Block encoding is an approach for representing a complex operator A via embedding it into a unitary matrix U [81]. As explained in the main text, we can prepare a $(2n+2)$-qubit block-encoding unitary $\hat{U}_{\mathrm{BE}}$ which encodes a diagonal matrix A containing the amplitudes of our max vector $|\lambda_{\max}\rangle$. This block-encoding circuit involves six calls to the controlled unitary preparation circuit $\hat{U}_{\max}$ [76] (see Fig. 6).

Once we have represented operators in the block-encoded form [Eq. (4)], we can proceed with information processing based on the unitary operators. The beauty of such a representation is in the ability to manipulate eigenvalues of associated operators by controlled actions conditioned on just one qubit. This corresponds to quantum signal processing (QSP) and quantum singular value transformation (QSVT) techniques. In their essence, these techniques represent polynomial transformations of eigenvalues $\lambda \mapsto P(\lambda)$, where $P(x)$ is a polynomial approximation of some selected function $f(x) \approx P(x)$. QSVT is the approach used when transformation is applied to arbitrary operators (matrices) [70,82].

Generally, quantum signal processing designs polynomials of any variable x by embedding it with single-qubit rotations,

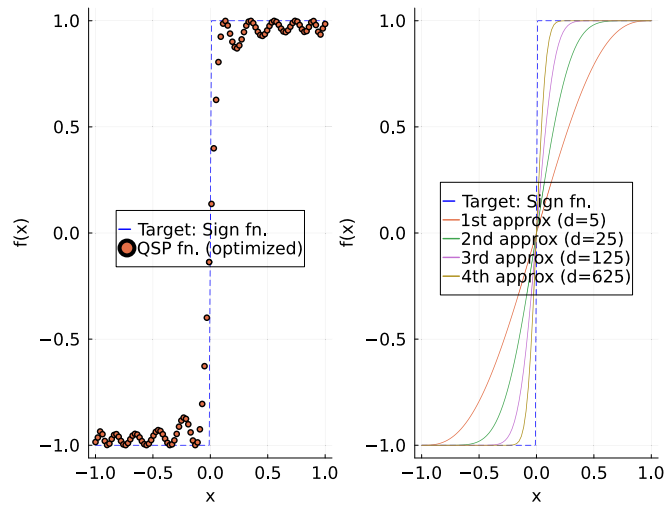

FIG. 7. QSP approximation of the sign function using angles generated by optimization (left) and recursive methods (right). For the optimization, $d = 29$ angles were used to achieve a decent approximation, while for the recursive method we plot the first-, second-, third-, and fourth-order approximations, corresponding to $d = 5, 25, 125, 625$.

based on the additional phases $\boldsymbol{\phi}$ that define the type of polynomial [83]. This polynomial transformation is achieved through a sequence of unitary operations [82]:

$$U(\boldsymbol{\phi}, x) = e^{i\phi_0 Z} \prod_{k=1}^{d} (W(x) e^{i\phi_k Z} W(x)^\dagger). \tag{A1}$$

$W(x)$ is called the signal rotation operator which encodes the scalar x, and $\boldsymbol{\phi}$ is a vector of phase angles. This sequence with the correct sequence of angles performs the transformation $x \mapsto P(x)$, where $P(x) = \langle 0|U(\boldsymbol{\phi}, x)|0\rangle$.

As noted in the main text, QSP is optimized for polynomial functions of fixed degree, yet we can still find angles that can approximate the nonpolynomial sign function. For example, we can opt for brute force optimization, or alternatively, recursive methods using an analytically found set of angles is also effective [84]. We show a comparison between the two methods in Fig. 7. QSP with optimization reaches a good approximation with fewer angles (and hence shorter circuit depth), but recursive methods provide smoother functions and give better convergence to the exact sign function at higher orders.

QSVT, a generalization of QSP, specifically targets the singular values of the matrix A. It allows the implementation of a desired polynomial transformation $P(A)$ on the singular values of A using a sequence of block-encoded operations and phase rotations. The QSVT sequence is made up of block-encoding unitaries $\hat{U}_{\mathrm{BE}}$ interleaved with projector-controlled phase (PCP) gates Π_ϕ. Since our block encoding uses $2n+2$ qubits, we require the PCP gates to apply $e^{i\phi}$ onto the subspace corresponding to the n system qubits, and $e^{-i\phi}$ to the $(n+2)$-dimensional ancilla subspace. We can implement this PCP gate using an additional ancilla, and we show part of the full QSVT sequence in Fig. 8.

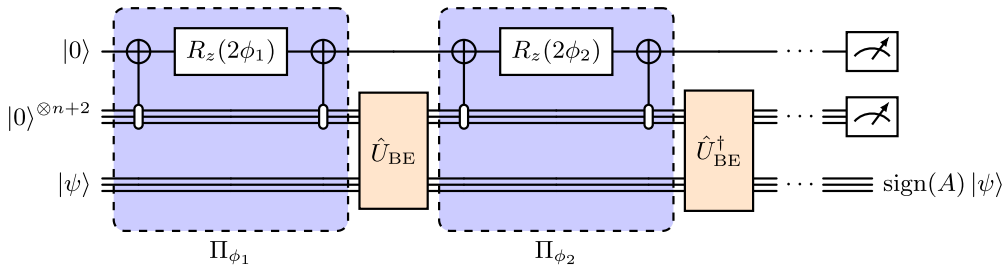

FIG. 8. Snippet of the QSVT sequence $\hat{U}_{\mathrm{QSVT}}$ for the transformation $A \to \mathrm{sign}(A)$, with the circuit for $\hat{U}_{\mathrm{BE}}$ given in Fig. 6. If 0 is measured on the ancilla qubits, the transformed matrix $\mathrm{sign}(A)$ is applied onto the system register $|\psi\rangle$. The accuracy of the application of the sign function depends on the choice of angles ϕ.

The full QSVT sequence acts on the singular values of A such that

$$P(A) = UP(\Sigma)V^{\dagger}, \tag{A2}$$

where $A = U\Sigma V^{\dagger}$ is the singular value decomposition of A, and Σ is the diagonal matrix of singular values.

[1] S. Aaronson, *Quantum Computing Since Democritus* (Cambridge University Press, Cambridge, 2013).

[2] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Quantum computational chemistry, Rev. Mod. Phys. **92**, 015003 (2020).

[3] Y. Alexeev *et al.*, Quantum-centric supercomputing for materials science: A perspective on challenges and future directions, Future Gener. Comput. Syst. **160**, 666 (2024).

[4] D. R. Simon, On the power of quantum computation, SIAM J. Comput. **26**, 1474 (1997),.

[5] S. Aaronson, BQP and the polynomial hierarchy, in *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10 (Association for Computing Machinery, New York, 2010), p. 141–150.

[6] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).

[7] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. **103**, 150502 (2009).

[8] A. M. Childs, R. Kothari, and R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, SIAM J. Comput. **46**, 1920 (2017).

[9] L. Lin and Y. Tong, Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems, Quantum **4**, 361 (2020).

[10] Y. Subaşı, R. D. Somma, and D. Orsucci, Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing, Phys. Rev. Lett. **122**, 060504 (2019).

[11] S. Chakraborty, Implementing any linear combination of unitaries on intermediate-term quantum computers, Quantum **8**, 1496 (2024).

[12] P. C. S. Costa, D. An, Y. R. Sanders, Y. Su, R. Babbush, and D. W. Berry, Optimal scaling quantum linear-systems solver via discrete adiabatic theorem, PRX Quantum **3**, 040303 (2022).

[13] C. A. Williams, A. A. Gentile, V. E. Elfving, D. Berger, and O. Kyriienko, Quantum iterative methods for solving differential equations with application to computational fluid dynamics arXiv:2404.08605.

[14] N. Linden, A. Montanaro, and C. Shao, Quantum vs classical algorithms for solving the heat equation arXiv:2004.06516.

[15] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).

[16] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, Nature (London) **574**, 505 (2019).

[17] D. Hangleiter and J. Eisert, Computational advantage of quantum random sampling, Rev. Mod. Phys. **95**, 035001 (2023).

[18] C. Zoufal, A. Lucchi, and S. Woerner, Quantum generative adversarial networks for learning and loading random distributions, npj Quantum Inf. **5**, 103 (2019).

[19] B. Coyle, D. Mills, V. Danos, and E. Kashefi, The born supremacy: Quantum advantage and training of an Ising born machine, npj Quantum Inf. **6**, 60 (2020).

[20] A. E. Paine, V. E. Elfving, and O. Kyriienko, Quantum quantile mechanics: Solving stochastic differential equations for generating time-series, Adv. Quantum Technol. **6**, 2300065 (2023).

[21] O. Kyriienko, A. E. Paine, and V. E. Elfving, Protocols for trainable and differentiable quantum generative modeling, Phys. Rev. Res. **6**, 033291 (2024).

[22] S. Kasture, O. Kyriienko, and V. E. Elfving, Protocols for classically training quantum generative models on probability distributions, Phys. Rev. A **108**, 042406 (2023).

[23] H.-Y. Wu, V. E. Elfving, and O. Kyriienko, Multidimensional quantum generative modeling by quantum Hartley transform, Adv. Quantum Technol. **8**, 2400337 (2025).

[24] A. Abbas, A. Ambainis, B. Augustino, A. Bärtschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, N. Franco, F. Fratini, B. Fuller, J. Gacon, C. Gonciulea, S. Gribling, S. Gupta, S. Hadfield, R. Heese, G. Kircher *et al.*, Challenges and opportunities in quantum optimization, Nat. Rev. Phys. **6**, 718 (2024).

[25] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, Phys. Rev. X **10**, 021067 (2020).

[26] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar *et al.*, Quantum optimization of maximum independent set using Rydberg atom arrays, Science **376**, 1209 (2022).

[27] L. Zhu, H. L. Tang, G. S. Barron, F. A. Calderon-Vargas, N. J. Mayhall, E. Barnes, and S. E. Economou, Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, Phys. Rev. Res. **4**, 033029 (2022).

[28] S. P. Jordan, N. Shutty, M. Wootters, A. Zalcman, A. Schmidhuber, R. King, S. V. Isakov, and R. Babbush, Optimization by decoded quantum interferometry, Nature (London) **646**, 831 (2025).

[29] Edelsbrunner, Letscher, and Zomorodian, Topological persistence and simplification, Discrete Comput. Geom. **28**, 511 (2002).

[30] A. Zomorodian and G. Carlsson, Computing persistent homology, Discrete Comput. Geom. **33**, 249 (2005).

[31] G. Carlsson, Topology and data, Bull. Am. Math. Soc. **46**, 255 (2009).

[32] L. Wasserman, Topological data analysis, Annu. Rev. Stat. Appl. **5**, 501 (2018).

[33] F. Chazal and B. Michel, An introduction to topological data analysis: Fundamental and practical aspects for data scientists, Front. Artif. Intell. **4**, 667963 (2021).

[34] S. Lloyd, S. Garnerone, and P. Zanardi, Quantum algorithms for topological and geometric analysis of data, Nat. Commun. **7**, 10138 (2016).

[35] C. Gyurik, C. Cade, and V. Dunjko, Towards quantum advantage via topological data analysis, Quantum **6**, 855 (2022).

[36] D. W. Berry, Y. Su, C. Gyurik, R. King, J. Basso, A. Del Toro Barba, A. Rajput, N. Wiebe, V. Dunjko, and R. Babbush, Analyzing prospects for quantum advantage in topological data analysis, PRX Quantum **5**, 010319 (2024).

[37] S. Scali, C. Umeano, and O. Kyriienko, Quantum topological data analysis via the estimation of the density of states, Phys. Rev. A **110**, 042616 (2024).

[38] S. Scali, C. Umeano, and O. Kyriienko, The topology of data hides in quantum thermal states, APL Quantum **1**, 036106 (2024).

[39] I. Y. Akhalwaya, S. Ubaru, K. L. Clarkson, M. S. Squillante, V. Jejjala, Y.-H. He, K. Naidoo, V. Kalantzis, and L. Horesh, Topological data analysis on noisy quantum computers, arXiv:2209.09371.

[40] C. Dürr, M. Heiligman, P. HOyer, and M. Mhalla, Quantum query complexity of some graph problems, SIAM J. Comput. **35**, 1310 (2006).

[41] T. Krauss and J. McCollum, Solving the network shortest path problem on a quantum annealer, IEEE Trans. Quantum Eng. **1**, 1 (2020).

[42] G. Chapuis, H. Djidjev, G. Hahn, and G. Rizk, Finding maximum cliques on a quantum annealer, in *Proceedings of the Computing Frontiers Conference*, CF '17 (ACM, New York, 2017), pp. 63–70.

[43] J. M. Arrazola and T. R. Bromley, Using Gaussian boson sampling to find dense subgraphs, Phys. Rev. Lett. **121**, 030503 (2018).

[44] L. Banchi, M. Fingerhuth, T. Babej, C. Ing, and J. M. Arrazola, Molecular docking with Gaussian boson sampling, Sci. Adv. **6**, eaax1950 (2020).

[45] S. Yu, Z.-P. Zhong, Y. Fang, R. B. Patel, Q.-P. Li, W. Liu, Z. Li, L. Xu, S. Sagona-Stophel, E. Mer, S. E. Thomas, Y. Meng, Z.-P. Li, Y.-Z. Yang, Z.-A. Wang, N.-J. Guo, W.-H. Zhang, G. K. Tranmer, Y. Dong, Y.-T. Wang *et al.*, A universal programmable Gaussian boson sampler for drug discovery, Nat. Comput. Sci. **3**, 839 (2023).

[46] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, Exponential algorithmic speedup by a quantum walk, in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03 (Association for Computing Machinery, New York, 2003), pp. 59–68.

[47] R. Babbush, D. W. Berry, R. Kothari, R. D. Somma, and N. Wiebe, Exponential quantum speedup in simulating coupled classical oscillators, Phys. Rev. X **13**, 041041 (2023).

[48] S. H. Strogatz, Exploring complex networks, Nature (London) **410**, 268 (2001).

[49] R. Albert and A.-L. Barabási, Statistical mechanics of complex networks, Rev. Mod. Phys. **74**, 47 (2002).

[50] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Complex networks: Structure and dynamics, Phys. Rep. **424**, 175 (2006).

[51] M. Bardoscia, P. Barucca, S. Battiston, F. Caccioli, G. Cimini, D. Garlaschelli, F. Saracco, T. Squartini, and G. Caldarelli, The physics of financial networks, Nat. Rev. Phys. **3**, 490 (2021).

[52] S. Fortunato, Community detection in graphs, Phys. Rep. **486**, 75 (2010).

[53] H. Liao, M. S. Mariani, M. Medo, Y.-C. Zhang, and M.-Y. Zhou, Ranking in evolving complex networks, Phys. Rep. **689**, 1 (2017).

[54] E. de Silva and M. P. Stumpf, Complex networks and simple models in biology, J. R. Soc. Interface. **2**, 419 (2005).

[55] E. J. Deeds, J. Krivine, J. Feret, V. Danos, and W. Fontana, Combinatorial complexity and compositional drift in protein interaction networks, PLoS ONE **7**, e32032 (2012).

[56] W. He, W. Xu, X. Ge, Q.-L. Han, W. Du, and F. Qian, Secure control of multiagent systems against malicious attacks: A brief survey, IEEE Trans. Indust. Inf. **18**, 3595 (2022).

[57] S. Klamt and J. Stelling, Combinatorial complexity of pathway analysis in metabolic networks, Mol. Biol. Rep. **29**, 233 (2002).

[58] J. Biamonte, M. Faccin, and M. De Domenico, Complex networks from classical to quantum, Commun. Phys. **2**, 53 (2019).

[59] M. E. J. Newman and M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E **69**, 026113 (2004).

[60] M. E. J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. USA **103**, 8577 (2006).

[61] U. von Luxburg, A tutorial on spectral clustering, Stat. Comput. **17**, 395 (2007).

[62] L. Kučera, Expected complexity of graph partitioning problems, Discrete Appl. Math. **57**, 193 (1995).

[63] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, Recent advances in graph partitioning, in *Algorithm Engineering* (Springer International Publishing, Berlin, 2016), pp. 117–158.

[64] K. Andreev and H. Räcke, Balanced graph partitioning, in *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA04 (ACM, New York, 2004), pp. 120–124.

[65] B. Hendrickson and T. G. Kolda, Graph partitioning models for parallel computing, Parallel Comput. **26**, 1519 (2000).

[66] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure* (Springer International Publishing, Berlin, 2022).

[67] S. V. Patil and D. B. Kulkarni, K-way spectral graph partitioning for load balancing in parallel computing, Int. J. Inf. Technol. **13**, 1893 (2021).

[68] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, Generalized louvain method for community detection in large networks, in *2011 11th International Conference on Intelligent Systems Design and Applications* (IEEE, New York, 2011), pp. 88–93.

[69] S. Van Dongen, Graph clustering via a discrete uncoupling process, SIAM J. Matrix Anal. Appl. **30**, 121 (2008).

[70] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC '19 (ACM, New York, 2019).

[71] R. Qu and Y. Bao, Quantum entanglement in n-qubit real equally weighted states, arXiv:1207.2518.

[72] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (Association for Computing Machinery, New York, 1996), pp. 212–219.

[73] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proc. R. Soc. Lond. A **439**, 553 (1992).

[74] G. H. Low and I. L. Chuang, Hamiltonian simulation by qubitization, Quantum **3**, 163 (2019).

[75] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[76] A. G. Rattew and P. Rebentrost, Nonlinear transformations of quantum amplitudes: Exponential improvement, generalization, and applications, arXiv:2309.09839.

[77] M. Rossi, M. Huber, D. Bruß, and C. Macchiavello, Quantum hypergraph states, New J. Phys. **15**, 113022 (2013).

[78] H. Alghassi, A. Deshmukh, N. Ibrahim, N. Robles, S. Woerner, and C. Zoufal, A variational quantum algorithm for the Feynman-Kac formula, Quantum **6**, 730 (2022).

[79] C. Umeano, V. E. Elfving, and O. Kyriienko, Geometric quantum machine learning of $\mathrm{BQP}^{\mathcal{A}}$ protocols and latent graph classifiers, arXiv:2402.03871.

[80] S. Scali and C. Umeano, Qu-dos/dataset-communitydetection: Zenodo (2025), https://zenodo.org/records/17370413.

[81] Y. Dong, X. Meng, K. B. Whaley, and L. Lin, Efficient phase-factor evaluation in quantum signal processing, Phys. Rev. A **103**, 042419 (2021).

[82] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, Grand unification of quantum algorithms, PRX Quantum **2**, 040203 (2021).

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

[84] K. Mizuta and K. Fujii, Recursive quantum eigenvalue and singular-value transformation: Analytic construction of matrix sign function by Newton iteration, Phys. Rev. Res. **6**, L012007 (2024).